7.5

Tworzenie aplikacji dla produktu 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 1155.

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.

Spis treści

Projektowanie aplikacji

Produkt IBM WebSphere MQ udostępnia kilka sposobów tworzenia aplikacji w celu wysyłania i odbierania komunikatów, które są potrzebne do obsługi procesów biznesowych. Istnieje również możliwość tworzenia aplikacji do zarządzania menedżerami kolejek i powiązanymi zasobami.

Przed opracowaniu aplikacji dla produktu IBM WebSphere MQnależy zapoznać się z pojęciami w sekcji Przegląd techniczny produktuIBM WebSphere MQIBM WebSphere MQ -przegląd techniczny.

Istnieje możliwość tworzenia aplikacji dla produktu IBM WebSphere MQ w różnych językach programowania. Aby uzyskać informacje na temat obsługiwanych języków programowania i ich funkcji, patrz “Wybór języka programowania do użycia” na stronie 80.

Informacje na temat typów aplikacji, które można napisać dla produktu IBM WebSphere MQ na różnych platformach, można znaleźć w następujących sekcjach.

Typy aplikacji, które można napisać dla platform IBM WebSphere MQ

Te informacje są związane z typami aplikacji, które można zapisać na serwerze IBM WebSphere MQ.

Produkty IBM WebSphere MQ to menedżery kolejek i aktywatory aplikacji. Obsługują one interfejs MQI produktu IBM Message Queue Interface (MQI), przez który programy mogą umieszczać komunikaty w kolejce i uzyskać komunikaty z kolejki.

Produkt IBM WebSphere MQ dla platform innych niżz/OS umożliwia pisanie aplikacji, które:

- Wysyłanie komunikatów do innych aplikacji działających w ramach tych samych systemów operacyjnych. Aplikacje mogą znajdować się na tym samym lub innym systemie.
- Wysyłanie komunikatów do aplikacji, które działają na innych platformach IBM WebSphere MQ .
- Kolejkowanie komunikatów z programu CICS for TXSeries for AIX, TXSeries for HP-UX, TXSeries for Solaris oraz TXSeries for Windows systems.
- Użyj kolejkowania komunikatów z poziomu Encina dla systemów AIX, HP-UX, Solaris i Windows .
- Użyj kolejkowania komunikatów z systemu Tuxedo dla systemów AIX, AT & T, HP-UX, Solaris i Windows .
- Produkt IBM WebSphere MQ należy używać jako menedżera transakcji, koordynując aktualizacje wykonywane przez zewnętrznych menedżerów zasobów w ramach jednostek pracy IBM WebSphere MQ . Następujące zewnętrzne menedżery zasobów są obsługiwane i są zgodne z interfejsem XA X/OPEN.
  - DB2
  - Informix
  - Oracle
  - Sybase
- Przetwarzanie kilku komunikatów razem jako pojedyncza jednostka pracy, która może zostać zatwierdzona lub wycofana.
- Uruchom w pełnym środowisku produktu IBM WebSphere MQ lub uruchom go z poziomu środowiska klienta MQI produktu IBM WebSphere MQ na następujących platformach:
  - UNIX and Linux® systemy
  - Windows

Pojęcia pokrewne

Zabezpieczenia

Pojęcia związane z projektowaniem aplikacji

Do pisania aplikacji IBM WebSphere MQ można użyć wyboru języków proceduralnych lub obiektowych. Odsyłacze w tym temacie można znaleźć w informacjach dotyczących pojęć związanych z produktem IBM WebSphere MQ , które są przydatne dla programistów aplikacji.

Przed rozpoczęciem projektowania i pisania aplikacji produktu IBM WebSphere MQ należy zapoznać się z podstawowymi pojęciami dotyczącymi produktu IBM WebSphere MQ , a także zapoznać się z tematami w sekcji Przegląd techniczny. Więcej informacji na temat typów aplikacji, które można napisać dla produktu IBM WebSphere MQ, zawiera sekcja “Projektowanie aplikacji” na stronie 7.

Użyj poniższych odsyłaczy, aby dowiedzieć się więcej na temat pojęć IBM WebSphere MQ z programowaniu aplikacji:

- “Komunikaty produktu IBM WebSphere MQ” na stronie 10
- Przesyłanie komunikatów w modelu punkt-punkt
- Wprowadzenie do przesyłania komunikatów w trybie publikowania/subskrypcji produktu WebSphere MQ
- “Korzystanie z interfejsu kolejki komunikatów (MQI) w aplikacji klienckiej” na stronie 361
- “Korzystanie z usług Web Service w produkcie WebSphere MQ” na stronie 972
- “Programy obsługi wyjścia kanału dla kanałów przesyłania komunikatów” na stronie 406
- “Scenariusze obsługi transakcyjnej” na stronie 42

Przed uruchomieniem aplikacji, które korzystają z interfejsu MQI, należy utworzyć określone obiekty produktu IBM WebSphere MQ . Więcej informacji na ten temat zawiera sekcja “Aplikacje korzystające z interfejsu MQI” na stronie 9.

Pojęcia pokrewne

“Projektowanie aplikacji produktu IBM WebSphere MQ” na stronie 91
Po zdecydowaniu się, w jaki sposób aplikacje mogą korzystać z platform i środowisk, które są dostępne dla użytkownika, należy zdecydować, w jaki sposób korzystać z funkcji oferowanych przez produkt WebSphere MQ.

“Przykładowe programy produktu WebSphere MQ” na stronie 98
Ta kolekcja tematów zawiera informacje na temat przykładowych programów WebSphere MQ na różnych platformach.

“Pisanie aplikacji kolejkowania” na stronie 199
Ta sekcja zawiera informacje na temat pisania aplikacji kolejkowania, łączenia się i rozłączania z menedżerem kolejek, publikowania/subskrypcji oraz obiektów otwierających i zamykających.

“Pisanie aplikacji klienckich” na stronie 360
Co należy wiedzieć, aby pisać aplikacje klienckie w produkcie WebSphere MQ.

“Wybór języka programowania do użycia” na stronie 80
W tej sekcji znajdują się informacje na temat języków programowania i środowisk obsługiwanych przez produkt IBM WebSphere MQ, a także niektóre zagadnienia dotyczące ich używania.

“Korzystanie z klas produktu WebSphere MQ dla usługi JMS” na stronie 734
Klasy WebSphere MQ classes for Java Message Service (klasy WebSphere MQ classes for JMS) są dostawcą JMS dostarczonym z produktem WebSphere MQ. Oprócz implementowania interfejsów zdefiniowanych w pakiecie javax.jms , klasy WebSphere MQ classes for JMS udostępniają dwa zestawy rozszerzeń do interfejsu API JMS.

“Korzystanie z interfejsu modelu obiektu komponentu (klasy automatyzacji produktu WebSphere MQ dla elementu ActiveX)” na stronie 1062
Klasy automatyzacji produktu WebSphere MQ dla ActiveX (MQAX) to komponenty ActiveX , które udostępniają klasy, których można użyć w aplikacji w celu uzyskania dostępu do produktu WebSphere MQ.

“Korzystanie z klas produktu WebSphere MQ dla języka Java” na stronie 669

Klasy produktu WebSphere MQ dla języka Java umożliwiają korzystanie z produktu WebSphere MQ w środowisku Java. Aplikacja Java może korzystać z klas produktu WebSphere MQ dla języka Java lub klas WebSphere MQ dla usługi JMS w celu uzyskania dostępu do zasobów produktu WebSphere MQ .

“Używanie środowiska .NET” na stronie 573
Klasy produktu WebSphere MQ dla środowiska .NET umożliwiają programowi napisanego w środowisku programowania .NET nawiązanie połączenia z produktem WebSphere MQ jako klienta MQI produktu WebSphere MQ lub nawiązanie bezpośredniego połączenia z serwerem WebSphere MQ .

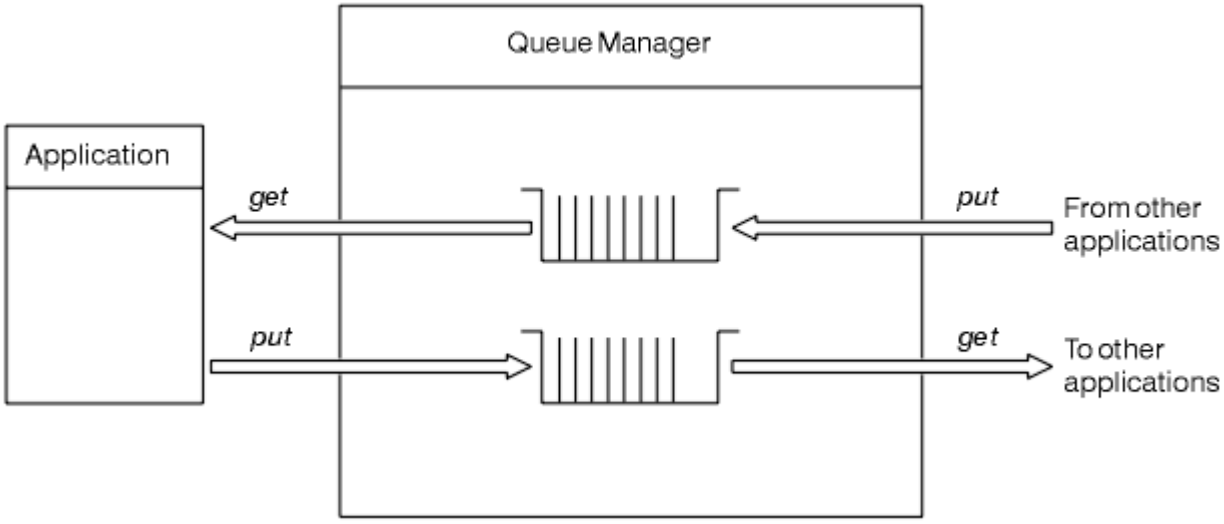
“Używanie języka C++” na stronie 644
ProduktWebSphere MQ udostępnia klasy języka C++ odpowiadające obiektom WebSphere MQ oraz niektóre dodatkowe klasy równoważne z typami danych tablicowych. Udostępnia ona wiele funkcji, które nie są dostępne w interfejsie MQI.

“Budowanie aplikacji IBM WebSphere MQ” na stronie 439
Ta sekcja zawiera informacje na temat budowania aplikacji produktu IBM WebSphere MQ na różnych platformach.

Aplikacje korzystające z interfejsu MQI

Programy aplikacji IBM WebSphere MQ wymagają pewnych obiektów, zanim będą mogły działać poprawnie.

Rysunek 1 na stronie 9 przedstawia aplikację, która usuwa komunikaty z kolejki, przetwarza je, a następnie wysyła wyniki do innej kolejki w tym samym menedżerze kolejek.



Rysunek 1. Kolejki, komunikaty i aplikacje

Aplikacje mogą umieszczać komunikaty w kolejkach lokalnych lub zdalnych (przy użyciu programu MQPUT), mogą one tylko pobrać komunikaty bezpośrednio z kolejek lokalnych (za pomocą MQGET).

Zanim będzie można uruchomić tę aplikację, muszą zostać spełnione następujące warunki:

- Menedżer kolejek musi istnieć i być uruchomiony.
- Pierwsza kolejka aplikacji, z której komunikaty mają zostać usunięte, musi być zdefiniowana.
- Druga kolejka, na której aplikacja umieszcza komunikaty, musi być również zdefiniowana.
- Aplikacja musi być w stanie połączyć się z menedżerem kolejek. Aby to zrobić, musi być on połączony z produktem IBM WebSphere MQ. Patrz sekcja “Budowanie aplikacji IBM WebSphere MQ” na stronie 439.
- Aplikacje, które umieszczają komunikaty w pierwszej kolejce, muszą również łączyć się z menedżerem kolejek. Jeśli są one zdalne, muszą być również skonfigurowane z kolejkami transmisji i kanałami. Ta część systemu nie jest wyświetlana w programie Rysunek 1 na stronie 9.

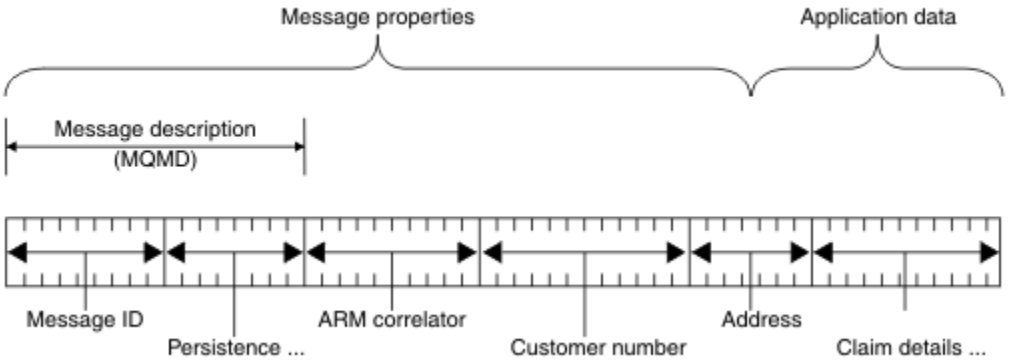
Komunikaty produktu IBM WebSphere MQ

W tej sekcji przedstawiono pojęcie komunikatu produktu IBM WebSphere MQ , części komunikatu i deskryptor komunikatu.

Komunikaty produktu IBM WebSphere MQ składają się z dwóch części:

- Właściwości komunikatu
- Dane aplikacji

Rysunek 2 na stronie 10 reprezentuje komunikat i pokazuje, w jaki sposób jest logicznie podzielony na właściwości komunikatu i dane aplikacji.



Rysunek 2. Reprezentacja komunikatu

Dane aplikacji przenoszone w komunikacie WebSphere MQ nie są zmieniane przez menedżer kolejek, chyba że wykonywana jest na nim konwersja danych. Ponadto produkt WebSphere MQ nie ma żadnych ograniczeń dotyczących treści tych danych. Długość danych w każdym komunikacie nie może przekraczać wartości atrybutu *MaxMsgLength* zarówno w menedżerze kolejek, jak i w menedżerze kolejek.

On WebSphere MQ for AIX, WebSphere MQ for HP-UX, WebSphere MQ for Linux, WebSphere MQ for Solaris, and WebSphere MQ for Okna, the *MaxMsgLength* defaults to 100 MB (104 857 600 bytes).

W niektórych okolicznościach komunikaty są nieco krótsze niż wartość atrybutu *MaxMsgLength* . Więcej informacji zawiera sekcja “Dane w komunikacie” na stronie 236.

Komunikat tworzy się podczas korzystania z wywołań MQI MQPUT lub MQPUT1 . Jako dane wejściowe dla tych wywołań należy podać informacje sterujące (takie jak priorytet komunikatu i nazwę kolejki odpowiedzi) oraz dane użytkownika, a następnie wywołanie powoduje umieszczenie komunikatu w kolejce. Więcej informacji na temat tych wywołań można znaleźć w sekcji MQPUT i MQPUT1 .

deskryptor komunikatu
Dostęp do informacji sterujących komunikatów można uzyskać za pomocą struktury MQMD, która definiuje *deskryptor komunikatu*.

Pełny opis struktury MQMD znajduje się w sekcji MQMD-deskryptor komunikatu.

Opis sposobu użycia pól w strukturze MQMD, które zawierają informacje o pochodzeniu komunikatu, zawiera sekcja “Kontekst komunikatu” na stronie 39 .

Istnieją różne wersje deskryptora komunikatu. Dodatkowe informacje na temat grupowania i segmentacji komunikatów (patrz “Grupy komunikatów” na stronie 36) są dostępne w wersji 2 deskryptora komunikatu (lub MQMDE). Jest on taki sam, jak deskryptor komunikatu w wersji 1, ale zawiera dodatkowe pola. Są one opisane w sekcji MQMDE-Rozszerzenie deskryptora komunikatu.

Typy komunikatów

Istnieją cztery typy komunikatów zdefiniowane przez produkt IBM WebSphere MQ.

Te cztery komunikaty są następujące:

- Datagram

- Komunikaty żądań
- Komunikaty odpowiedzi
- Komunikaty raportów
  - Typy komunikatów raportu
  - Opcje komunikatu raportu

Aplikacje mogą korzystać z pierwszych trzech typów komunikatów w celu przekazywania informacji między sobą. Czwarty typ raportu jest przeznaczony dla aplikacji i menedżerów kolejek, które mają być używane do raportowania informacji o zdarzeniach, takich jak wystąpienie błędu.

Każdy typ komunikatu jest identyfikowany przez wartość MQMT_ *. Można również zdefiniować własne typy komunikatów. Informacje na temat zakresu wartości, których można użyć, zawiera sekcja MsgType.

Datagramy

datagramu należy używać wtedy, gdy nie jest wymagana odpowiedź z aplikacji, która odbiera komunikat (to znaczy pobiera komunikat z kolejki).

Przykład aplikacji, która może używać datagramów, to taka, która wyświetla informacje o locie w salonie lotniskowym. Komunikat może zawierać dane dla całego ekranu informacji o locie. Taka aplikacja prawdopodobnie nie zażądała potwierdzenia dla komunikatu, ponieważ prawdopodobnie nie ma znaczenia, czy komunikat nie został dostarczony. Po krótkim czasie aplikacja wysyła komunikat aktualizacji.

Komunikaty żądań

Użyj *komunikatu żądania* , jeśli chcesz uzyskać odpowiedź z aplikacji, która odbierze komunikat.

Przykład aplikacji, która może używać komunikatów żądań, to taka, która wyświetla saldo konta sprawdzanego. Komunikat żądania może zawierać numer konta, a komunikat odpowiedzi będzie zawierał saldo konta.

Jeśli chcesz powiązać komunikat odpowiedzi z komunikatem żądania, istnieją dwie opcje:

- Utwórz aplikację obsługując komunikat żądania, który jest odpowiedzialny za zapewnienie, że umieszcza informacje w komunikacie odpowiedzi, który odnosi się do komunikatu żądania.
- Użyj pola raportu w deskryptorze komunikatu żądania, aby określić treść pól *MsgId* i *CorrelId* w komunikacie odpowiedzi:
  - Można zażądać, aby *MsgId* lub `CorrelId` oryginalnego komunikatu był kopiowany do pola *CorrelId* komunikatu odpowiedzi (domyślnym działaniem jest kopiowanie elementu *MsgId*).
  - Można zażądać, aby dla komunikatu odpowiedzi został wygenerowany nowy `MsgId` , albo że element *MsgId* oryginalnego komunikatu ma zostać skopiowany do pola *MsgId* komunikatu odpowiedzi (domyślnym działaniem jest wygenerowanie nowego identyfikatora komunikatu).

Komunikaty odpowiedzi

W przypadku odpowiedzi na inny komunikat należy użyć *komunikatu odpowiedzi* .

Podczas tworzenia komunikatu odpowiedzi należy przestrzegać wszystkich opcji, które zostały ustawione w deskryptorze komunikatu, do którego tworzona jest odpowiedź. Opcje raportu określają treść pól identyfikatora komunikatu (`MsgId`) oraz identyfikatora korelacji (`CorrelId`). Te pola umożliwiają aplikacji, która odbiera odpowiedź, w celu skorelowania odpowiedzi z jej oryginalnym żądaniem.

Komunikaty raportów

Komunikaty raportów informują aplikacje o zdarzeniach, takich jak wystąpienie błędu podczas przetwarzania komunikatu.

Mogą one być generowane przez:

- Menedżer kolejek,
- Agent kanału komunikatów (na przykład, jeśli nie mogą dostarczyć komunikatu), lub
- Aplikacja (na przykład, jeśli nie może korzystać z danych w komunikacie).

Komunikaty raportów mogą być generowane w dowolnym momencie i mogą pojawić się w kolejce, gdy aplikacja nie oczekuje na nie.

Typy komunikatów raportu

Po umieszczniu komunikatu w kolejce można wybrać opcję odbierania:

- *Komunikat raportu o wyjątku*. Ta opcja jest wysyłana w odpowiedzi na komunikat z ustawioną flagą wyjątków. Jest on generowany przez agenta kanału komunikatów (MCA) lub aplikację.
- *Komunikat raportu utraty ważności*. Oznacza to, że aplikacja podjęła próbę pobrania komunikatu, który osiągnął próg utraty ważności. Komunikat jest oznaczany do usunięcia. Ten typ raportu jest generowany przez menedżer kolejek.
- *Komunikat o potwierdzeniu przybycia (COA)*. Oznacza to, że komunikat osiągnął swoją kolejkę docelową. Jest on generowany przez menedżer kolejek.
- Komunikat raportu *Potwierdzenie dostarczenia (COD)*. Oznacza to, że komunikat został pobrany przez aplikację odbierającą. Jest on generowany przez menedżer kolejek.
- *Komunikat raportu powiadomienia o działaniu pozytywnym (PAN)*. Oznacza to, że żądanie zostało pomyślnie obsłużone (co oznacza, że działanie żądane w komunikacie zostało wykonane pomyślnie). Ten typ raportu jest generowany przez aplikację.
- *Komunikat raportu powiadomienia o działaniu negatywnym (NAN)*. Oznacza to, że żądanie nie zostało pomyślnie obsłużone (co oznacza, że działanie żądane w komunikacie nie zostało wykonane pomyślnie). Ten typ raportu jest generowany przez aplikację.

Uwaga: Każdy typ komunikatu raportu zawiera jedną z następujących wartości:

- Cały oryginalny komunikat
- Pierwsze 100 bajtów danych w oryginalnym komunikacie
- Brak danych z oryginalnego komunikatu

W przypadku umieszczenia komunikatu w kolejce można zażądać więcej niż jednego typu komunikatu raportu. Jeśli zostanie wybrany komunikat o potwierdzeniu dostarczenia i opcje komunikatu raportu o wyjątku, jeśli komunikat nie zostanie dostarczony, zostanie wyświetlony komunikat o wyjątku. Jeśli jednak zostanie wybrana tylko opcja komunikatu z potwierdzeniem dostarczenia, a komunikat nie zostanie dostarczony, nie zostanie wyświetlony komunikat o raporcie o wyjątku.

Komunikaty raportu, które są wysyłane, gdy spełnione są kryteria generowania konkretnego komunikatu, są jedynymi otrzymanymi.

Opcje komunikatów raportu

Po wystąpieniu wyjątku można *odrzucić* komunikat. Jeśli wybrano opcję odrzucenia i zażądano komunikatu raportu o wyjątku, komunikat raportu będzie kierowany do serwerów *ReplyToQ* i *ReplyToQMgr*, a oryginalny komunikat zostanie usunięty.

Uwaga: Korzyścią z tego jest to, że można zmniejszyć liczbę komunikatów, które trafiają do kolejki niedostarczonych komunikatów. Oznacza to jednak, że aplikacja, o ile nie wysyła tylko komunikatów datagramu, ma do czynienia z zwrócanych wiadomości. Po wygenerowaniu komunikatu raportu o wyjątku dziedziczy on trwałość oryginalnego komunikatu.

Jeśli komunikat raportu nie może zostać dostarczony (jeśli kolejka jest pełna, na przykład), komunikat raportu jest umieszczany w kolejce niedostarczonych komunikatów.

Jeśli chcesz otrzymać komunikat raportu, podaj nazwę kolejki odpowiedzi w polu *ReplyToQ* . W przeciwnym razie wywołanie MQPUT lub MQPUT1 oryginalnego komunikatu nie powiedzie się i zostanie wyświetlony komunikat MQRC_MISSING_REPLY_TO_Q.

W deskryptorze komunikatu (MQMD) komunikatu można użyć innych opcji raportu, aby określić treść pól *MsgId* i *CorrelId* wszystkich komunikatów raportu utworzonych dla tego komunikatu:

- Użytkownik może zażądać skopiowania *MsgId* lub *CorrelId* oryginalnego komunikatu do pola *CorrelId* komunikatu raportu. Domyślne działanie polega na skopiowaniu identyfikatora komunikatu. Użyj identyfikatora MQRO_COPY_MSG_ID_TO_CORRELID, ponieważ umożliwia nadawcę komunikatu korelowanie komunikatu odpowiedzi lub raportu z oryginalnym komunikatem. Identyfikator korelacji komunikatu odpowiedzi lub komunikatu raportu jest identyczny z identyfikatorem komunikatu oryginalnego komunikatu.
- Użytkownik może zażądać, aby dla komunikatu raportu został wygenerowany nowy *MsgId* albo że *MsgId* oryginalnego komunikatu ma zostać skopiowany do pola *MsgId* komunikatu raportu. Domyślne działanie polega na wygenerowaniu nowego identyfikatora komunikatu. Użyj komendy MQRO_NEW_MSG_ID, ponieważ zapewnia on, że każdy komunikat w systemie ma inny identyfikator komunikatu i może zostać jednoznacznie odróżniony od wszystkich innych komunikatów w systemie.
- W przypadku aplikacji specjalizowanych może być konieczne użycie identyfikatora MQRO_PASS_MSG_ID lub MQRO_PASS_CORREL_ID. Należy jednak zaprojektować aplikację, która odczytuje komunikaty z kolejki, aby upewnić się, że działa poprawnie, gdy na przykład kolejka zawiera wiele komunikatów o tym samym identyfikatorze komunikatu.

  Aplikacje serwera muszą sprawdzić ustawienia tych flag w komunikacie żądania i odpowiednio ustawić pola *MsgId* i *CorrelId* w komunikacie odpowiedzi lub komunikacie raportu.

  Aplikacje, które działają jako pośrednicy między aplikacją requestera a aplikacją serwera, nie muszą sprawdzać ustawień tych flag. Jest to spowodowane tym, że zwykle te aplikacje przesyłają komunikat do aplikacji serwera w niezmienionych polach *MsgId*, *CorrelId*i *Report* . Dzięki temu aplikacja serwera może skopiować *MsgId* z oryginalnego komunikatu w polu *CorrelId* komunikatu odpowiedzi.

Podczas generowania raportu na temat komunikatu, aplikacje serwera muszą sprawdzić, czy którekolwiek z tych opcji zostały ustawione.

Więcej informacji na temat sposobu korzystania z komunikatów raportu zawiera sekcja Raport.

Aby wskazać rodzaj raportu, menedżery kolejek używają zakresu kodów sprzężenia zwrotnego. Te kody umieszczają w polu *Feedback* deskryptora komunikatu komunikatu raportu. Menedżery kolejek mogą również zwracać kody przyczyny MQI w polu *Feedback* . IBM WebSphere MQ Definiuje zakres kodów sprzężenia zwrotnego dla aplikacji, które mają być używane.

Więcej informacji na temat informacji zwrotnych i kodów przyczyny można znaleźć w sekcji Opinia.

Przykładem programu, który może użyć kodu sprzężenia zwrotnego, jest monitorowanie obciążeń innych programów obsługujących kolejkę. Jeśli istnieje więcej niż jedna instancja programu obsługującego kolejkę, a liczba komunikatów przybywających do kolejki nie uzasadnia tego, taki program może wysłać komunikat raportu (z kodem sprzężenia zwrotnego MQFB_QUIT) do jednego z programów obsługujących, aby wskazać, że program powinien zakończyć działanie. (Program monitorujący może użyć wywołania MQINQ, aby dowiedzieć się, ile programów serwuje kolejkę).

Raporty i segmentowane komunikaty

Nieobsługiwane w produkcie WebSphere MQ for z/OS .

Jeśli komunikat jest segmentowany (opis komunikatów segmentowanych zawiera sekcja “Segmentacja komunikatów” na stronie 270), a użytkownik prosi o wygenerowanie raportów, może otrzymać więcej raportów niż to, co zrobiłby, gdyby komunikat nie był posegmentowany.

Dla raportów wygenerowanych przez produkt WebSphere MQ

Jeśli komunikaty zostaną segmentowane lub zezwolono na to menedżer kolejek, istnieje tylko jedna obserwacja, w której można oczekiwać otrzymania pojedynczego raportu dla całego komunikatu. Jest to sytuacja, w której zażądano tylko raportów COD i określono MQGMO_COMPLETE_MSG dla aplikacji pobierających.

W innych przypadkach aplikacja musi być przygotowana do obsługi kilku raportów, zwykle po jednym dla każdego segmentu.

Uwaga: Jeśli komunikaty są segmentowane i wymagane jest tylko pierwsze 100 bajtów oryginalnych danych komunikatu, które mają zostać zwrócone, należy zmienić ustawienie opcji raportu, aby poprosić o raporty bez danych dla segmentów, które mają przesunięcie 100 lub więcej. Jeśli tego nie zrobisz, pozostaw to ustawienie tak, aby każdy segment żądał 100 bajtów danych, a użytkownik wczytał komunikaty raportu za pomocą pojedynczej komendy MQGET, podając MQGMO_COMPLETE_MSG, do dużego komunikatu zawierającego 100 bajtów danych odczytu na każdym odpowiednim przesuniętym przesunięciom. W takim przypadku potrzebny jest duży bufor lub konieczne jest podanie wartości MQGMO_ACCEPT_TRUNCATED_MSG.

Dla raportów generowanych przez aplikacje

Jeśli aplikacja generuje raporty, zawsze kopiuj nagłówki WebSphere MQ , które są obecne na początku oryginalnych danych komunikatu do danych komunikatu raportu.

Następnie do danych komunikatu raportu należy dodać wartość none (brak), 100 bajtów (lub wszystkie oryginalne dane komunikatu) (lub inną wartość, która zwykle jest dołączana).

Nagłówki WebSphere MQ , które muszą zostać skopiowane, można rozpoznać po wyszukaniu kolejnych nazw formatów, rozpoczynając od deskryptora MQMD i kontynuując wszystkie obecne nagłówki. Następujące nazwy `Format` wskazują następujące nagłówki produktu WebSphere MQ :

- MQMDE
- MQDLH
- MQXQH
- MQIIH.
- MQH*

MQH* oznacza dowolną nazwę rozpoczynający się od znaków MQH.

Nazwa `Format` występuje na określonych pozycjach dla MQDLH i MQXQH, ale dla innych nagłówków WebSphere MQ występuje w tej samej pozycji. Długość nagłówka jest zawarta w polu, które występuje również w tej samej pozycji dla nagłówków MQMDE, MQIMS i wszystkich nagłówków MQH*.

Jeśli używany jest deskryptor MQMD w wersji 1, a użytkownik zgłasza segment lub komunikat w grupie lub komunikat, dla którego dozwolony jest segmentację, dane raportu muszą rozpoczynać się od wywołania MQMDE. W polu *OriginalLength* należy ustawić długość oryginalnych danych komunikatu, z wyłączeniem długości wszystkich znalezionych nagłówków WebSphere MQ .

Pobieranie raportów

Jeśli zapytasz o raporty COA lub COD, możesz poprosić o ich ponowne zmontowania dla Ciebie z MQGMO_COMPLETE_MSG.

Wywołanie MQGET z MQGMO_COMPLETE_MSG jest spełnione, gdy w kolejce znajdują się wystarczające komunikaty raportów (jednego typu, na przykład COA, i z tym samym *GroupId*), które reprezentują jeden kompletny oryginalny komunikat. Jest to prawda, nawet jeśli same komunikaty raportu nie zawierają pełnych danych pierwotnych; pole *OriginalLength* w każdym komunikacie raportu podaje długość pierwotnych danych reprezentowanych przez ten komunikat raportu, nawet jeśli same dane nie są obecne.

Tej techniki można użyć nawet wtedy, gdy w kolejce znajduje się kilka różnych typów raportów (na przykład COA i COD), ponieważ MQGET z MQGMO_COMPLETE_MSG zawiera komunikaty raportów tylko wtedy, gdy mają ten sam kod produktu *Feedback* . Nie można jednak używać tej techniki w przypadku raportów o wyjątkach, ponieważ w ogóle mają one różne kody *Feedback* .

Tej techniki można użyć do uzyskania pozytywnego wskazania, że cały komunikat dotarł. Jednak w większości przypadków konieczne jest zastosowanie niektórych segmentów w celu uzyskania pewnych segmentów, podczas gdy inne mogą wygenerować wyjątek (lub wygaśnięcie, o ile to możliwe). W tym przypadku nie można użyć komendy MQGMO_COMPLETE_MSG, ponieważ w ogólnym przypadku mogą być różne kody *Feedback* dla różnych segmentów, a w przypadku segmentu może być więcej niż jeden raport. Można jednak użyć komendy MQGMO_ALL_SEGMENTS_AVAILABLE.

Aby zezwolić na to, konieczne może być pobranie raportów w miarę ich przybycia, a następnie zbudowanie obrazu w aplikacji, co stało się z oryginalnym komunikatem. Można użyć pola *GroupId* w komunikacie raportu, aby skorelować raporty z*GroupId* oryginalnego komunikatu oraz pole *Feedback*, aby określić typ każdego komunikatu raportu. Sposób, w jaki to robisz, zależy od wymagań aplikacji.

Jedno podejście jest następujące:

- Zapytaj o raporty dotyczące COD i raporty o wyjątkach.
- Po określonym czasie sprawdź, czy kompletny zestaw raportów COD został odebrany za pomocą komendy MQGMO_COMPLETE_MSG. Jeśli tak, aplikacja wie, że cały komunikat został przetworzony.
- Jeśli nie, a raporty o wyjątkach odnoszące się do tego komunikatu są obecne, rozwiążesz problem w odniesieniu do nieposegmentowanych komunikatów, ale upewnij się, że w pewnym momencie wyczyszczasz segmenty osierote.
- Jeśli istnieją segmenty, dla których nie ma żadnych raportów, oryginalne segmenty (lub raporty) mogą oczekiwać na ponowne podłączenie kanału lub przeciążenia sieci w pewnym momencie. Jeśli w ogóle nie otrzymano żadnych raportów o wyjątkach (lub jeśli uważasz, że te, które mogą być tylko tymczasowe), możesz zdecydować się na to, aby aplikacja czekała nieco dłużej.

  Tak jak poprzednio, jest to podobne do rozważań, jakie masz przy kontaktach z nieposegmentowanymi wiadomościami, poza tym, że trzeba również rozważyć możliwość czyszczenia segmentów osierotnych.

Jeśli oryginalny komunikat nie jest krytyczny (na przykład jeśli jest to zapytanie lub komunikat, który może zostać powtórzony później), należy ustawić czas utraty ważności, aby upewnić się, że osierote segmenty są usuwane.

Menedżery kolejek na poziomie zaplecza

Gdy raport jest generowany przez menedżer kolejek, który obsługuje segmentację, ale jest odbierany w menedżerze kolejek, który *nie* obsługuje segmentacji, struktura MQMDE (identyfikująca *Offset* i *OriginalLength* reprezentowane przez raport) jest zawsze uwzględniana w danych raportu, oprócz zera, 100 bajtów lub wszystkich oryginalnych danych w komunikacie.

Jeśli jednak segment komunikatu przechodzi przez menedżer kolejek, który nie obsługuje segmentacji, jeśli raport jest tam generowany, struktura MQMDE w pierwotnym komunikacie jest traktowana wyłącznie jako dane. W związku z tym nie jest ona uwzględniana w danych raportu, jeśli zażądano zerowej liczby bajtów danych pierwotnych. Bez wywołania MQMDE komunikat raportu może nie być przydatny.

Zażądaj co najmniej 100 bajtów danych w raportach, jeśli istnieje możliwość, że komunikat może przemieszczać się przez menedżer kolejek z powrotem na poziomie zaplecza.

Format informacji sterujących komunikatów i danych komunikatu

Menedżer kolejek jest zainteresowany tylko formatem informacji sterujących w komunikacie, podczas gdy aplikacje, które obsługują ten komunikat, są zainteresowane formatem zarówno informacji sterujących, jak i danych.

Format informacji o sterowaniu komunikatami

Informacje sterujące w polach łańcucha znaków w deskryptorze komunikatu muszą znajdować się w zestawie znaków używanym przez menedżer kolejek.

Ten zestaw znaków jest definiowany przez atrybut *CodedCharSetId* obiektu menedżera kolejek. Informacje sterujące muszą znajdować się w tym zestawie znaków, ponieważ gdy aplikacje przekazują komunikaty z jednego menedżera kolejek do innego, agenty kanałów komunikatów, które przesyłają komunikaty, używają wartości tego atrybutu w celu określenia, jakie dane mają być wykonywane.

Format danych komunikatu

Można określić dowolną z następujących czynności:

- Format danych aplikacji
- Zestaw znaków danych znakowych
- Format danych liczbowych

W tym celu należy użyć następujących pól:

Format
Oznacza to, że odbiorca komunikatu ma format danych aplikacji w komunikacie.

Gdy menedżer kolejek tworzy komunikat, w niektórych okolicznościach używa pola *Format* w celu zidentyfikowania formatu tego komunikatu. Na przykład, gdy menedżer kolejek nie może dostarczyć komunikatu, umieszcza komunikat w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów). Dodaje nagłówek (zawierający więcej informacji sterujących) do komunikatu, a następnie zmienia pole *Format* , aby to pokazać.

Menedżer kolejek ma *wbudowane formaty* o nazwach rozpoczynających się od MQ, na przykład MQFMT_STRING. Jeśli nie są one zgodne z potrzebami użytkownika, można zdefiniować własne formaty (*formaty zdefiniowane przez użytkownika*), ale nie wolno używać nazw rozpoczynających się od MQ .

Podczas tworzenia własnych formatów i korzystania z nich należy napisać wyjście konwersji danych w celu obsługi programu pobieranego za pomocą komendy MQGMO_CONVERT.

CodedCharSetId
Definiuje zestaw znaków danych znakowych w komunikacie. Aby ustawić ten zestaw znaków na wartość tego menedżera kolejek, można ustawić to pole na stałe MQCCSI_Q_MGR lub MQCCSI_INHERIT.

Po dostaniu komunikatu z kolejki należy porównać wartość pola *CodedCharSetId* z wartością oczekiwaną przez aplikację. Jeśli te dwie wartości różnią się, może być konieczne przekształcenie dowolnych danych znakowych w komunikacie lub użycie wyjścia komunikatu konwersji danych, jeśli jest ono dostępne.

Encoding
Opisuje format danych liczbowych komunikatu, który zawiera binarne liczby całkowite, liczby całkowite z upakowanymi liczbami całkowitymi i liczby zmiennopozycyjne. Zwykle jest ona zakodowana zgodnie z określoną maszyną, na której jest uruchomiony menedżer kolejek.

Po umieszczniu komunikatu w kolejce zwykle należy podać stałą MQENC_NATIVE w polu *Encoding* . Oznacza to, że kodowanie danych komunikatu jest takie samo, jak kodowanie komputera, na którym działa aplikacja.

Po dostaniu komunikatu z kolejki należy porównać wartość pola *Encoding* w deskryptorze komunikatu z wartością stałej MQENC_NATIVE na komputerze. Jeśli te dwie wartości różnią się, może być konieczne przekształcenie dowolnych danych liczbowych w komunikacie lub użycie wyjścia komunikatu konwersji danych, jeśli jest ono dostępne.

Konwersja danych aplikacji
Dane aplikacji mogą wymagać konwersji na zestaw znaków i kodowanie wymagane przez inną aplikację, w której występują różne platformy.

Może ona zostać przekształcona w wysyłającym menedżerze kolejek lub w odbierającym menedżerze kolejek. Jeśli biblioteka wbudowanych formatów nie spełnia Twoich potrzeb, możesz zdefiniować swoje własne. Typ konwersji zależy od formatu komunikatu określonego w polu formatu deskryptora komunikatu, MQMD.

Uwaga: Komunikaty z podanym parametrem MQFMT_NONE nie są przekształcane.

Konwersja w wysyłającym menedżerze kolejek

Ustaw atrybut kanału CONVERT na wartość YES, jeśli chcesz, aby wysyłający agent kanału komunikatów (MCA) przekształcił dane aplikacji.

Konwersja jest wykonywana w wysyłającym menedżerze kolejek dla niektórych wbudowanych formatów i dla formatów zdefiniowanych przez użytkownika, jeśli zostanie dostarczone odpowiednie wyjście użytkownika.

Wbudowane formaty

Są to:

- Komunikaty, które są znakami wszystkich znaków (przy użyciu nazwy formatu MQFMT_STRING)
- Komunikaty zdefiniowane w produkcie WebSphere MQ , na przykład formaty komend programowalnych

  Produkt WebSphere MQ korzysta z komunikatów w formacie komend programowalnych dla komunikatów administracyjnych i zdarzeń (w tym przypadku używana jest nazwa formatu MQFMT_ADMIN). Można użyć tego samego formatu (przy użyciu nazwy formatu MQFMT_PCF) dla własnych komunikatów, a także skorzystać z wbudowanej konwersji danych.

Wszystkie wbudowane formaty menedżera kolejek mają nazwy rozpoczynające się od wywołania MQFMT. Są one wymienione i opisane w sekcji Format.

Formaty zdefiniowane przez aplikację

W przypadku formatów zdefiniowanych przez użytkownika, konwersja danych aplikacji musi być wykonywana przez program obsługi wyjścia konwersji danych (więcej informacji na ten temat zawiera sekcja “Pisanie wyjść konwersji danych” na stronie 426). W środowisku klient-serwer wyjście jest ładowane na serwerze, a konwersja odbywa się w tym miejscu.

Konwersja w odbierającym menedżerze kolejek

Dane komunikatu aplikacji mogą być przekształcane przez otrzymujący menedżer kolejek zarówno dla formatów wbudowanych, jak i zdefiniowanych przez użytkownika.

Konwersja jest wykonywana w trakcie przetwarzania wywołania MQGET, jeśli zostanie określona opcja MQGMO_CONVERT. Szczegółowe informacje na ten temat zawiera sekcja Opcje .

Kodowane zestawy znaków

Produkty WebSphere MQ obsługują kodowane zestawy znaków, które są udostępniane przez bazowy system operacyjny.

Podczas tworzenia menedżera kolejek używany jest identyfikator kodowanego zestawu znaków (CCSID) menedżera kolejek, który jest oparty na środowisku bazowym. Jeśli jest to mieszana strona kodowa, produkt WebSphere MQ używa części SBCS strony kodowej mieszanej jako identyfikatora CCSID menedżera kolejek.

W przypadku ogólnego przekształcenia danych, jeśli bazowy system operacyjny obsługuje strony kodowe DBCS, produkt WebSphere MQ może z niego korzystać.

Szczegółowe informacje na temat obsługiwanych kodowanych zestawów znaków można znaleźć w dokumentacji systemu operacyjnego.

Podczas pisania aplikacji, które obejmują wiele platform, należy wziąć pod uwagę konwersję danych aplikacji, nazwy formatów i wyjścia użytkownika. Informacje na temat wywoływania i zapisywania wyjść konwersji danych można znaleźć w sekcji “Pisanie wyjść konwersji danych” na stronie 426 .

Priorytety komunikatów

Priorytet komunikatu (w polu *Priority* struktury MQMD) ustawia się podczas umieszczania komunikatu w kolejce. Dla priorytetu można ustawić wartość liczbową lub pozwolić na to, aby komunikat był domyślnym priorytetem kolejki.

Atrybut *MsgDeliverySequence* kolejki określa, czy komunikaty w kolejce są przechowywane w sekwencji FIFO (najpierw w pierwszej kolejności), czy w FIFO w kolejności priorytetów. Jeśli ten atrybut jest ustawiony na wartość MQMDS_PRIORITY, komunikaty są umieszczane w kolejce z priorytetem

określonym w polu *Priority* ich deskryptorów komunikatów; ale jeśli jest ustawiony na wartość MQMDS_FIFO, komunikaty są umieszczane w kolejce z domyślnym priorytetem kolejki. Komunikaty o równym priorytecie są zapisywane w kolejce w celu ich przybycia.

Atrybut *DefPriority* kolejki ustawia domyślną wartość priorytetu dla komunikatów umieszczanych w tej kolejce. Ta wartość jest ustawiana podczas tworzenia kolejki, ale może zostać zmieniona później. Kolejki aliasowe i lokalne definicje kolejek zdalnych mogą mieć różne priorytety domyślne z kolejek podstawowych, do których są one rozstrzygane. Jeśli w ścieżce rozstrzygania znajduje się więcej niż jedna definicja kolejki (patrz “Rozdzielczość nazwy” na stronie 222), domyślny priorytet jest przyjmowany z wartości (w momencie operacji put) atrybutu *DefPriority* kolejki określonej w komendzie open.

Wartość atrybutu *MaxPriority* menedżera kolejek to maksymalny priorytet, który można przypisać do komunikatu przetwarzanego przez ten menedżer kolejek. Nie można zmienić wartości tego atrybutu. W produkcie WebSphere MQatrybut ma wartość 9; można tworzyć komunikaty o priorytetach między 0 (najniższy) i 9 (najwyższy).

Właściwości komunikatu

Za pomocą właściwości komunikatu można zezwolić aplikacji na wybór komunikatów do przetwarzania lub pobieranie informacji o komunikacie bez uzyskiwania dostępu do nagłówków MQMD lub MQRFH2 . Ułatwiają one także komunikację między aplikacjami WebSphere MQ i JMS.

Właściwość komunikatu to dane powiązane z komunikatem, składające się z nazwy tekstowej i wartości określonego typu. Właściwości komunikatu są używane przez selektory komunikatów do filtrowania publikacji na tematy lub do selektywnego pobierania komunikatów z kolejek. Właściwości komunikatu mogą być używane w celu uwzględnienia danych biznesowych lub informacji o stanie bez konieczności zapisywania ich w danych aplikacji. Aplikacje nie muszą uzyskiwać dostępu do danych w nagłówkach deskryptora komunikatu produktu MQ (MQMD) lub MQRFH2 , ponieważ pola w tych strukturach danych mogą być dostępne jako właściwości komunikatu przy użyciu wywołań funkcji interfejsu kolejki komunikatów (Message Queue Interface-MQI).

Użycie właściwości komunikatu w produkcie WebSphere MQ mimika użycie właściwości w usłudze JMS. Oznacza to, że można ustawiać właściwości w aplikacji JMS i pobierać je w proceduralnej aplikacji WebSphere MQ lub w innej rundce. Aby udostępnić właściwość dla aplikacji JMS, należy ją przypisać przedrostkiem "usr"; jest ona dostępna (bez przedrostka) jako właściwość użytkownika komunikatu JMS. Na przykład właściwość WebSphere MQ *usr.myproperty* (łańcuch znaków) jest dostępna dla aplikacji JMS przy użyciu wywołania JMS `message.getStringProperty('myproperty')`. Należy pamiętać, że aplikacje JMS nie mają dostępu do właściwości z przedrostkiem "usr", jeśli zawierają one dwa lub więcej U+002E (".") . Właściwość bez przedrostka i nie ma wartości U+002E (".") Znak jest traktowany tak, jakby miał przedrostek "usr". I odwrotnie, można uzyskać dostęp do właściwości użytkownika ustawionej w aplikacji JMS w aplikacji WebSphere MQ , dodając "usr." w wywołaniu MQINQMP z przedrostkiem nazwy właściwości.

Właściwości komunikatu i długość komunikatu

Użyj atrybutu menedżera kolejek *MaxPropertiesLength* , aby kontrolować wielkość właściwości, które mogą przepływać przez dowolny komunikat w menedżerze kolejek produktu WebSphere MQ .

Ogólnie rzecz biorąc, jeśli właściwości MQSETMP są używane do ustawiania właściwości, wielkość właściwości to długość nazwy właściwości w bajtach, plus długość wartości właściwości w bajtach, która została przekazana do wywołania MQSETMP. Istnieje możliwość zmiany zestawu znaków nazwy właściwości i wartości właściwości podczas przesyłania komunikatu do miejsca docelowego, ponieważ mogą one zostać przekształcone w kod Unicode. W takim przypadku wielkość właściwości może ulec zmianie.

W wywołaniu MQPUT lub MQPUT1 właściwości komunikatu nie są wliczane do długości komunikatu dla kolejki i menedżera kolejek, ale są one wliczane do długości właściwości, które są postrzegane przez menedżer kolejek (niezależnie od tego, czy zostały one ustawione przy użyciu wywołań MQI właściwości komunikatu).

Jeśli wielkość właściwości przekracza maksymalną długość właściwości, komunikat jest odrzucany za pomocą komendy MQRC_PROPERTIES_TOO_BIG. Ponieważ wielkość właściwości jest zależna od jego reprezentacji, należy ustawić maksymalną długość właściwości na poziomie brutto.

Aplikacja może pomyślnie umieścić komunikat w buforze, który jest większy niż wartość parametru *MaxMsgLength*, jeśli bufor zawiera właściwości. Jest to spowodowane tym, że nawet jeśli są reprezentowane jako elementy MQRFH2 , właściwości komunikatu nie są wliczane do długości komunikatu. Pola nagłówka MQRFH2 są dodawane do długości właściwości tylko wtedy, gdy jeden lub więcej folderów jest zawartych, a każdy folder w nagłówku zawiera właściwości. Jeśli jeden lub więcej folderów znajduje się w nagłówku MQRFH2 , a dowolny folder nie zawiera właściwości, wówczas pola nagłówka MQRFH2 są liczone w kierunku długości komunikatu.

W wywołaniu MQGET właściwości komunikatu nie są wliczane do długości komunikatu w odniesieniu do kolejki i menedżera kolejek. Ponieważ jednak właściwości są zliczane oddzielnie, możliwe jest, że bufor zwrócony przez wywołanie MQGET jest większy niż wartość atrybutu *MaxMsgLength* .

Nie należy używać zapytań o wartości *MaxMsgLength* , a następnie przydzielać bufor o takiej wielkości przed wywołaniem komendy MQGET. Zamiast tego należy przydzielić bufor, który jest wystarczająco duży. Jeśli wywołanie MQGET nie powiedzie się, należy przydzielić bufor z przewodnikiem o wielkości parametru *DataLength* .

Parametr *DataLength* wywołania MQGET zwraca długość (w bajtach) danych aplikacji i wszystkie właściwości zwrócone w udostępnionym buforze, jeśli uchwyt komunikatu nie jest określony w strukturze MQGMO.

Parametr *Buffer* wywołania MQPUT zawiera dane komunikatu aplikacji, które mają zostać wysłane, oraz wszystkie właściwości reprezentowane w danych komunikatu.

W przypadku przepływu do menedżera kolejek, który jest wcześniejszy niż wersja 7.0 produktu, właściwości komunikatu, z wyjątkiem tych, które znajdują się w deskryptorze komunikatu, są wliczane do długości komunikatu. Dlatego należy podnieść wartość atrybutu *MaxMsgLength* kanałów przechodnich do systemu w wersji wcześniejszej niż 7.0 w razie potrzeby, aby zrekompensować fakt, że dla każdego komunikatu może być wysłanych więcej danych. Alternatywnie można obniżyć kolejkę lub menedżer kolejek *MaxMsgLength*, tak aby ogólny poziom przesyłanych danych w całym systemie pozostał taki sam.

Dla właściwości komunikatu istnieje limit długości 100 MB, wyłączając deskryptor komunikatu lub rozszerzenie dla każdego komunikatu.

Wielkość właściwości w wewnętrznej reprezentacji to długość nazwy, powiększona o jej wartość, a także niektóre dane sterujące dla właściwości. Istnieje również kilka danych sterujących dla zestawu właściwości po dodaniu jednej właściwości do komunikatu.

Nazwy właściwości

Nazwa właściwości to łańcuch znaków. Niektóre ograniczenia mają zastosowanie do jego długości i zestawu znaków, które mogą być używane.

Nazwa właściwości to łańcuch znaków, w którym rozróżniana jest wielkość liter, ograniczona do +4095 znaków, chyba że kontekst został ograniczony przez kontekst. Ten limit jest zawarty w stałej wartości MQ_MAX_PROPERTY_NAME_LENGTH.

W przypadku przekroczenia tej maksymalnej długości podczas korzystania z wywołania MQI właściwości komunikatu wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_PROPERTY_NAME_LENGTH_ERR.

Ponieważ w usłudze JMS nie ma maksymalnej długości nazwy właściwości, aplikacja JMS może ustawić poprawną nazwę właściwości JMS, która nie jest poprawną nazwą właściwości produktu WebSphere MQ , jeśli jest ona przechowywana w strukturze MQRFH2 .

W tym przypadku podczas analizy używane są tylko pierwsze 4095 znaków nazwy właściwości. Następujące znaki są obcinane. Może to spowodować, że aplikacja używała selektorów, aby nie były one zgodne z łańcuchem wyboru, lub aby były zgodne z łańcuchem, który nie jest oczekiwany, ponieważ więcej niż jedna właściwość może być obcięta do tej samej nazwy. Gdy nazwa właściwości zostanie obcięta, produkt WebSphereMQ wysyła komunikat dziennika błędów.

Wszystkie nazwy właściwości muszą być zgodne z regułami zdefiniowanymi przez specyfikację języka Java dla identyfikatorów Java, z wyjątkiem tego, że znak Unicode U+002E (.) jest dozwolony jako część nazwy-ale nie jest to początek. Reguły dla identyfikatorów Java są zgodne z regułami zawartymi w specyfikacji JMS dla nazw właściwości.

Znaki białych znaków i operatory porównania są zabronione. Osadzone wartości NULL są dozwolone w nazwie właściwości, ale nie są zalecane. W przypadku użycia wbudowanych wartości null zapobiega to stosowaniu stałej MQVS_NULL_TERMINATED, gdy jest ona używana z strukturą MQCHARV w celu określenia łańcuchów długości zmiennych.

Nazwy właściwości należy przechowywać w prosty sposób, ponieważ aplikacje mogą wybierać komunikaty w oparciu o nazwy właściwości, a konwersja między zestawem znaków nazwy i selektora może spowodować niespodziewanie przełączenie.

Nazwy właściwości produktu WebSphere MQ używają znaku U+002E (.) w celu logicznego grupowania właściwości. Spowoduje to rozdziel przestrzeni nazw dla właściwości. Właściwości z następującymi przedrostkami, w dowolnej mieszance małych lub wielkich liter są zarezerwowane do użycia przez produkt:

- `mcd`
- `jms`
- `usr`
- `mq`
- `sib`
- `wmq`
- `Root`
- `Body`
- `Properties`

Dobrym sposobem uniknięcia starć nazw jest zapewnienie, aby wszystkie aplikacje prefiksował ich właściwości komunikatów z ich nazwą domeny internetowej. Na przykład w przypadku tworzenia aplikacji korzystającej z nazwy domeny "`ourcompany.com`" można nazwać wszystkie właściwości przedrostkiem "`com.ourcompany`". Ta konwencja nazewnictwa umożliwia także łatwy wybór właściwości. Na przykład aplikacja może zapytać o wszystkie właściwości komunikatu, które zaczynają się od "`com.ourcompany.%`".

Więcej informacji na temat korzystania z nazw właściwości zawiera sekcja Ograniczenia dotyczące nazw właściwości .

Ograniczenia dotyczące nazw właściwości
Podczas tworzenia nazwy właściwości należy przestrzegać określonych reguł.

W przypadku nazw właściwości obowiązują następujące ograniczenia:

1. Właściwość nie może rozpoczynać się od następujących łańcuchów:
   - "JMS"-zarezerwowane do użycia przez klasy produktu WebSphere MQ dla usługi JMS.
   - "usr.JMS"-niepoprawne.

   Jedynymi wyjątkami są następujące właściwości udostępniające synonimy dla właściwości JMS:

Właściwość	**Synonim dla**
JMSCorrelationID	Root .MQMD.CorrelId or jms.Cid
JMSDeliveryMode	Root .MQMD.Persistence or jms.Dlv
JMSDestination	jms.Dst
JMSExpiration	Root .MQMD.Expiry or jms.Exp
JMSMessageID	Root .MQMD.MsgId

Właściwość	Synonim dla
JMSPriority	Root .MQMD.Priority or jms.Pri
JMSRedelivered	Root .MQMD.BackoutCount
JMSReplyTo (łańcuch zakodowany jako identyfikator URI)	Root .MQMD.ReplyToQ or Root .MQMD.ReplyToQMgr or jms.Rto
JMSTimestamp	Root .MQMD.PutDate or Root .MQMD.PutTime or jms.Tms
JMSType	mcd.Type lub mcd.Set lub mcd.Fmt
JMSXAppID	Root .MQMD.PutApplName
JMSXDeliveryCount	Root .MQMD.BackoutCount
JMSXGroupID	Root .MQMD.GroupId or jms.Gid
JMSXGroupSeq	Root .MQMD.MsgSeqNumber or jms.Seq
JMSXUserID	Root .MQMD.UserIdentifier

Te synonimy umożliwiają aplikacji MQI uzyskiwanie dostępu do właściwości JMS w podobny sposób do klas produktu WebSphere MQ dla aplikacji klienckiej JMS. Z tych właściwości można ustawić tylko atrybuty JMSCorrelationID, JMSReplyTo, JMSType, JMSXGroupIDi JMSXGroupSeq za pomocą interfejsu MQI.

Należy zauważyć, że właściwości JMS_IBM_ * dostępne w klasach WebSphere MQ dla usługi JMS nie są dostępne przy użyciu interfejsu MQI. Pola, do których odwołują się odwołanie do właściwości JMS_IBM_ *, mogą być dostępne w inny sposób przez aplikacje MQI.

2. Właściwość nie może być wywoływana w żadnej mieszance mniejszej lub wielkiej litery, "NULL", "TRUE", "FALSE", "NOT", "AND", "OR", "BETWEEN", "LIKE", "IN", "IS" i "ESCAPE". These are the names of SQL keywords used in selection strings.
3. Nazwa właściwości rozpoczynający się od "mq " w dowolnej mieszance pisanej małymi lub wielkimi literami, a nie zaczynając "mq_usr" może zawierać tylko jeden znak "." znak (U+002E). Wiele "." znaki nie są dozwolone we właściwościach z tymi przedrostkami.
4. Dwa "." Znaki muszą zawierać inne znaki między; nie można mieć pustego miejsca w hierarchii. Podobnie nazwa właściwości nie może kończyć się na "." na końcu.
5. Jeśli aplikacja ustawi właściwość "a.b", a następnie właściwość "a.b.c", nie jest jasne, czy w hierarchii "b" znajduje się wartość, czy inna grupa logiczna. Such a hierarchy is "mixed content" and this is not supported. Ustawienie właściwości, która powoduje, że treść mieszana nie jest dozwolona.

Te ograniczenia są wymuszane przez mechanizm sprawdzania poprawności w następujący sposób:

- Poprawność nazw właściwości jest sprawdzana podczas ustawiania właściwości za pomocą wywołania MQSETMP – ustawianie właściwości komunikatu , jeśli sprawdzenie poprawności zostało zażądane, gdy uchwyt komunikatu został utworzony. If an attempt to validate a property is undertaken and fails due to an error in the specification of the property name, the completion code is MQCC_FAILED with reason:
  - Błąd MQRC_PROPERTY_NAME_ERROR dla powodów 1-4.
  - MQRC_MIXED_CONTENT_NOT_ALLOWED dla przyczyny 5.
- Nazwy właściwości określonych bezpośrednio jako elementy MQRFH2 nie są gwarantowane w celu sprawdzenia poprawności przez wywołanie MQPUT.

Pola deskryptora komunikatu jako właściwości
Większość pól deskryptora komunikatu może być traktowana jako właściwości. Nazwa właściwości jest tworzona przez dodanie przedrostka do nazwy pola deskryptora komunikatu.

Jeśli aplikacja MQI chce zidentyfikować właściwość komunikatu zawartą w polu deskryptora komunikatu (na przykład w łańcuchu selektora lub przy użyciu funkcji API właściwości komunikatu), należy użyć następującej składni:

Nazwa właściwości	Pole deskryptora komunikatu
Root.MQMD. < Pola>	< Pole >

Określ <Field> z tą samą sprawą co w przypadku pól struktury MQMD w deklaracji języka C. Na przykład nazwa właściwości `Root.MQMD.AccountingToken` uzyskuje dostęp do pola `AccountingToken` deskryptora komunikatu.

Pola `StrucId` i `Version` deskryptora komunikatu nie są dostępne z użyciem przedstawionej składni.

Pola deskryptora komunikatu nie są nigdy reprezentowane w nagłówku MQRFH2 , tak jak w przypadku innych właściwości.

Jeśli dane komunikatu są uruchamiane za pomocą wywołania MQMDE, który został uhonorowany przez menedżer kolejek, można uzyskać dostęp do pól MQMDE za pomocą opisanego w nich notacji `Root.MQMD.<Field>` . W tym przypadku pola MQMDE są traktowane jako logicznie część deskryptora MQMD z perspektywy właściwości. Patrz sekcja "MQMDE określona w wywołaniach MQPUT i MQPUT1 " w sekcji Przegląd produktu MQMDE.

Typy i wartości danych właściwości

Właściwością może być wartość boolowska, łańcuch bajtowy, łańcuch znaków lub liczba zmiennopozycyjna lub liczba całkowita. Ta właściwość może przechowywać dowolną poprawną wartość w zakresie typu danych, chyba że kontekst został ograniczony przez kontekst.

Typ danych wartości właściwości musi mieć jedną z następujących wartości:

- MQBOOL
- MQBYTE []
- MQCHAR []
- MQFLOAT32
- MQFLOAT64
- MQINT8
- MQINT16
- MQINT32
- MQINT64

Właściwość może istnieć, ale nie ma zdefiniowanej wartości. Jest to właściwość o wartości NULL. Właściwość o wartości NULL różni się od właściwości bajtowej (MQBYTE []) lub właściwości łańcucha znaków (MQCHAR []), ponieważ ma zdefiniowaną, ale pustą wartość, to znaczy jedną z wartością zerową.

Łańcuch bajtów nie jest poprawnym typem danych właściwości w usłudze JMS lub XMS. Zaleca się, aby nie używać właściwości łańcucha bajtów w folderze < usr>.

Wybieranie komunikatów z kolejek

Komunikaty z kolejek można wybierać przy użyciu pól MsgId i CorrelId w wywołaniu MQGET lub za pomocą komendy SelectionString w wywołaniu MQOPEN lub MQSUB.

Selektory

Selektor komunikatów to łańcuch o zmiennej długości używany przez aplikację do rejestrowania zainteresowania tylko tymi komunikatami, które mają właściwości, które spełniają zapytanie SQL (Structured Query Language), które reprezentuje łańcuch wyboru.

Wybór przy użyciu wywołań funkcji MQSUB i MQOPEN

Za pomocą *SelectionString*, która jest strukturą typu MQCHARV, można dokonywać wyborów przy użyciu wywołań MQSUB i MQOPEN.

Struktura *SelectionString* jest używana do przekazywania łańcucha wyboru o zmiennej długości do menedżera kolejek.

Identyfikator CCSID powiązany z łańcuchem selektora jest ustawiany za pomocą pola VSCCSID w strukturze MQCHARV. Użyta wartość musi być identyfikatorem CCSID, który jest obsługiwany dla łańcuchów selektora. Listę obsługiwanych stron kodowych zawiera sekcja Konwersja stron kodowych .

Określenie identyfikatora CCSID, dla którego nie ma obsługiwanej konwersji Unicode WebSphere MQ , powoduje wystąpienie błędu MQRC_SOURCE_CCSID_ERROR. Ten błąd jest zwracany w czasie, gdy selektor jest prezentowany w menedżerze kolejek, czyli w wywołaniu MQSUB, MQOPEN lub MQPUT1 .

Wartością domyślną w polu *VSCCSID* jest MQCCSI_APPL, co oznacza, że identyfikator CCSID łańcucha wyboru jest równy CCSID menedżera kolejek lub identyfikator CCSID klienta, jeśli jest on połączony za pośrednictwem klienta. Stała MQCCSI_APPL może być jednak przesłonięta przez aplikację, która przedefiniowuje ją przed kompilacją.

Jeśli selektor MQCHARV reprezentuje łańcuch o wartości NULL, nie ma miejsca dla tego konsumenta komunikatów, a komunikaty są dostarczane tak, jakby selektor nie był używany.

Maksymalna długość łańcucha wyboru jest ograniczona tylko przez to, co może być opisane w polu MQCHARV *VSLength*.

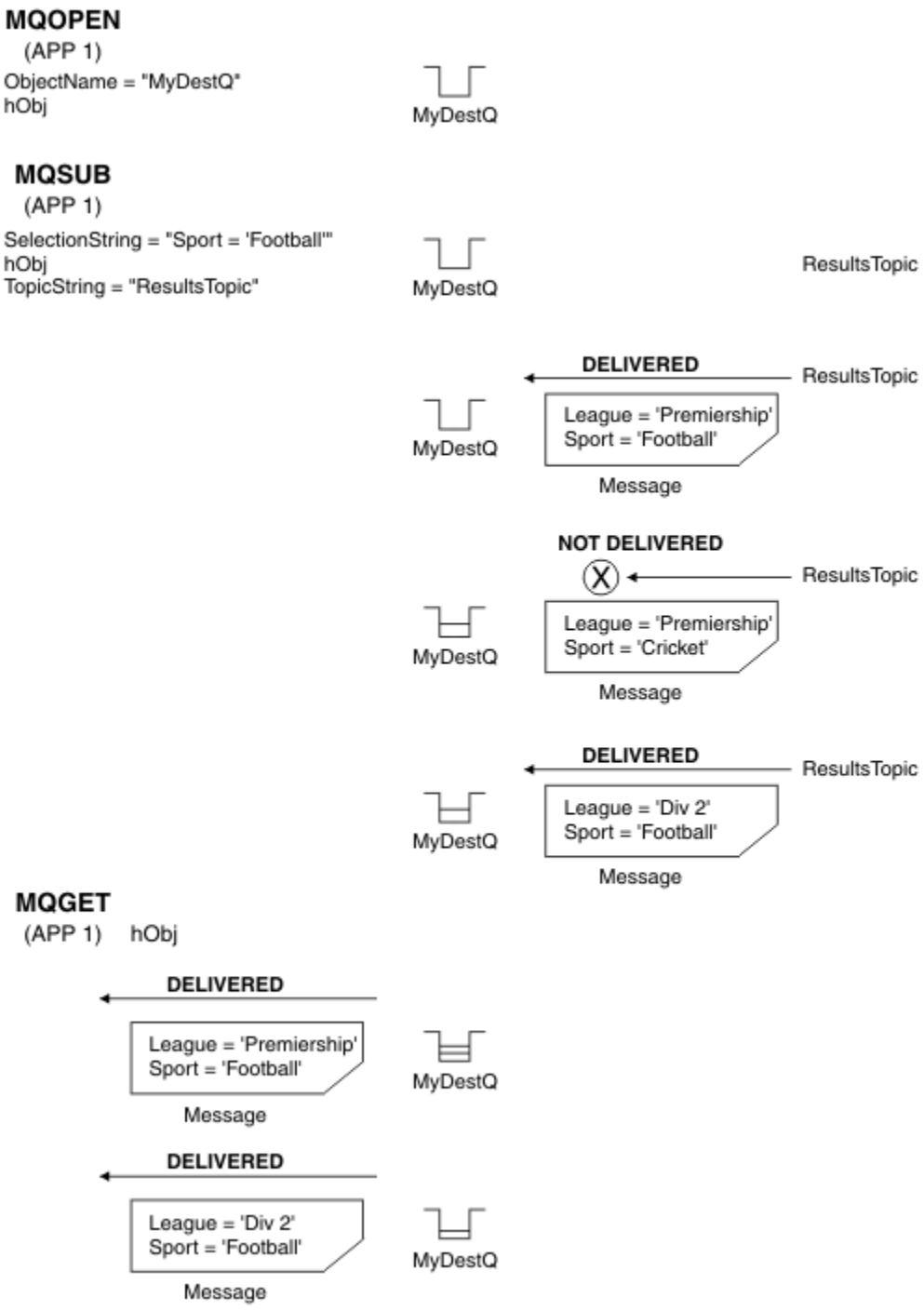
Element SelectionString jest zwracany w danych wyjściowych wywołania MQSUB przy użyciu opcji subskrypcji MQSO_RESUME, jeśli został podany bufor, a w polu VSBufSizeistnieje dodatnia długość buforu. Jeśli bufor nie zostanie podany, tylko długość łańcucha wyboru jest zwracana w polu długości VSLength tabeli MQCHARV. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w udostępnionym buforze zwracane są tylko bajty VSBufSize .

Aplikacja nie może zmienić łańcucha wyboru bez najpierw zamknięcia uchwytu do kolejki (dla MQOPEN) lub subskrypcji (dla MQSUB). Nowy łańcuch wyboru może zostać następnie określony przy kolejnych wywołaniu MQOPEN lub MQSUB.

MQOPEN
Użyj komendy MQCLOSE, aby zamknąć otwarty uchwyt, a następnie podaj nowy łańcuch wyboru w kolejnym wywołaniu MQOPEN.

MQSUB
Użyj komendy MQCLOSE, aby zamknąć zwrócony uchwyt subskrypcji (hSub), a następnie podaj nowy łańcuch wyboru w kolejnym wywołaniu MQSUB.

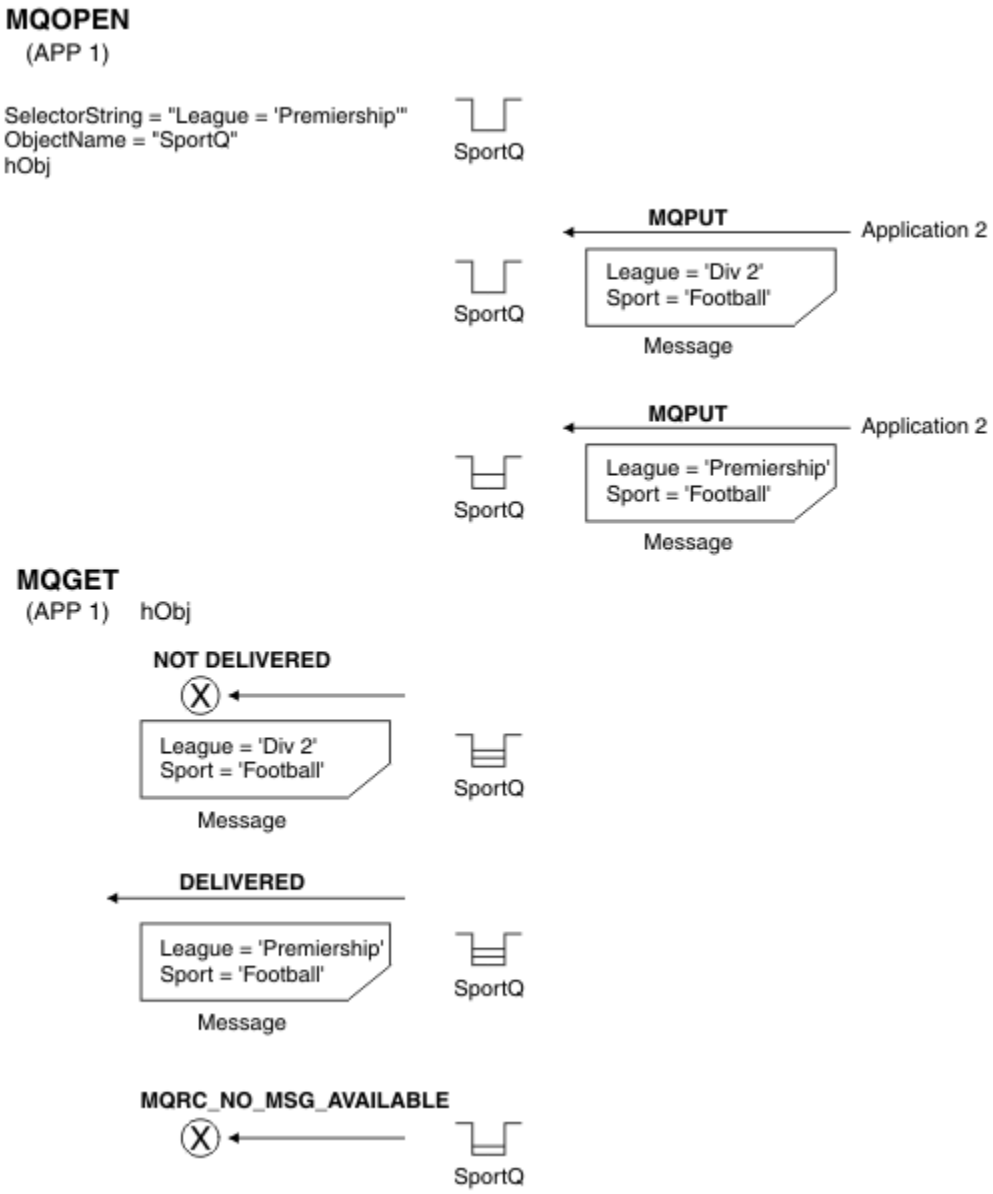
Rysunek 3 na stronie 24 przedstawia proces wyboru przy użyciu wywołania MQSUB.



Rysunek 3. Wybór przy użyciu wywołania MQSUB

Selektor można przekazać w wywołaniu do tabeli MQSUB, korzystając z pola *SelectionString* w strukturze MQSD. Efektem przekazania selektora w tabeli MQSUB jest to, że tylko te komunikaty publikowane w subskrybowanym temacie, które są zgodne z dostarczonym łańcuchem wyboru, są udostępniane w kolejce docelowej.

Rysunek 4 na stronie 25 przedstawia proces wyboru przy użyciu wywołania MQOPEN.



Rysunek 4. Wybór przy użyciu wywołania MQOPEN

Selektor może zostać przekazany w wywołaniu do wywołania MQOPEN za pomocą pola *SelectionString* w strukturze MQOD. Efektem przekazania selektora w wywołaniu MQOPEN jest to, że tylko te komunikaty w otwartej kolejce, które są zgodne z selektorem, są dostarczane do konsumenta komunikatów.

Głównym zastosowaniem selektora w wywołaniu MQOPEN jest przypadek punkt z punktem, w którym aplikacja może wybrać do odebrania tylko te komunikaty w kolejce, które są zgodne z selektorem. W poprzednim przykładzie przedstawiono prosty scenariusz, w którym dwa komunikaty są umieszczane w kolejce otwartej przez program MQOPEN, ale aplikacja pobierze tylko jeden z nich, ponieważ jest to jedyna taka aplikacja, która jest zgodna z selektorem.

Należy zauważyć, że kolejne wywołania MQGET w kolejce MQRC_NO_MSG_AVAILABLE nie istnieją w kolejce, która jest zgodna z podanym selektorem.

Zachowanie wyboru
Przegląd funkcji wyboru produktu IBM WebSphere MQ .

Pola w strukturze MQMDE są uznawane za właściwości komunikatu dla odpowiednich właściwości deskryptora komunikatu, jeśli deskryptor MQMD:

- Ma format MQFMT_MD_EXTENSION
- Jest natychmiast po niej następuje poprawna struktura MQMDE
- Jest wersją jedną lub zawiera tylko dwa pola w wersji domyślnej.

Możliwe jest, aby łańcuch wyboru był rozstrzygany na wartość PRAWDA lub FAŁSZ, zanim zostaną wybrane wszystkie zgodne właściwości komunikatu. Na przykład może to być przypadek, gdy łańcuch wyboru jest ustawiony na wartość `"TRUE <>FALSE"`. Taka wczesna ocena jest gwarantowana, aby mieć miejsce tylko wtedy, gdy w łańcuchu wyboru nie ma odwołań do właściwości komunikatu.

Jeśli łańcuch wyboru zostanie rozstrzygany na wartość TRUE, zanim zostaną uwzględnione wszystkie właściwości komunikatu, zostaną dostarczone wszystkie komunikaty opublikowane w temacie zasubskrybowanym przez konsumenta. Jeśli łańcuch wyboru zostanie rozstrzygany na FALSE przed uznaniem wszystkich właściwości komunikatu, w wywołaniu funkcji, która zaprezentowała selektor, zostanie zwrócony kod przyczyny MQRC_SELECTOR_ALWAYS_FALSE, a kod zakończenia MQCC_FAILED.

Nawet jeśli komunikat nie zawiera właściwości komunikatu (innych niż właściwości nagłówka), wówczas nadal może być zakwalifikowany do wyboru. Jeśli łańcuch wyboru odwołuje się do właściwości komunikatu, która nie istnieje, przyjmuje się, że ta właściwość ma wartość NULL lub wartość 'Unknown' (Nieznany).

Na przykład komunikat może nadal spełniać kryteria wyszukiwania, takie jak `'Color IS NULL'`, gdzie `'Color'` nie istnieje jako właściwość komunikatu w komunikacie.

Wybór może być wykonywany tylko dla właściwości powiązanych z komunikatem, a nie samego komunikatu, chyba że dostępny jest rozszerzony dostawca wyboru komunikatów. Wybór może być wykonywany na ładunku komunikatu tylko wtedy, gdy dostępny jest rozszerzony dostawca wyboru komunikatów.

Z każdą właściwością komunikatu jest powiązany typ. Podczas wykonywania wyboru należy upewnić się, że wartości używane w wyrażeniach do testowania właściwości komunikatu mają poprawny typ. Jeśli wystąpi niezgodność typów, wyrażenie, o którym mowa, jest tłumaczone na wartość FALSE.

Należy upewnić się, że łańcuch wyboru i właściwości komunikatu korzystają z zgodnych typów.

Kryteria wyboru są nadal stosowane w imieniu nieaktywnych trwałych subskrybentów, tak więc przechowywane są tylko komunikaty zgodne z łańcuchem wyboru, który został dostarczony.

Łańcuchy wyboru nie zmieniają się, gdy trwała subskrypcja jest wznawiana za pomocą komendy alter (MQSO_ALTER). Jeśli podczas wznawiania działania trwałego subskrybenta zostanie przedstawiony inny łańcuch wyboru, do aplikacji zostanie zwrócona wartość MQRC_SELECTOR_NOT_ALTERABLE.

Aplikacje otrzymują kod powrotu MQRC_NO_MSG_AVAILABLE, jeśli w kolejce nie ma komunikatu spełniający kryteria wyboru.

Jeśli aplikacja określiła łańcuch wyboru zawierający wartości właściwości, wówczas tylko te komunikaty, które zawierają zgodne właściwości, są zakwalifikowane do wyboru. Na przykład subskrybent określa łańcuch wyboru "a = 3", a publikowany jest komunikat, który nie zawiera właściwości, lub właściwości, w których wartość 'a' nie istnieje lub nie jest równa 3. Subskrybent nie odbiera tego komunikatu do kolejki docelowej.

Wydajność przesyłania komunikatów

Wybranie komunikatów z kolejki wymaga, aby program IBM WebSphere MQ sekwencyjnie inspekował każdy komunikat w kolejce. Komunikaty są sprawdzane do momentu znalezienia komunikatu, który jest zgodny z kryteriami wyboru lub nie ma więcej komunikatów do sprawdzenia. W związku z tym wydajność przesyłania komunikatów jest wystarczająca, jeśli wybór komunikatów jest używany w głębokich kolejkach.

Aby zoptymalizować wybór komunikatów w głębokich kolejkach, gdy wybór jest oparty na elemencie JMSCorrelationID lub JMSMessageID, należy użyć łańcucha wyboru w postaci `JMSCorrelationID = ...` lub `JMSMessageID = ...` i odwoływać się tylko do jednej właściwości.

Ta metoda zapewnia znaczącą poprawę wydajności dla wyboru w elemencie JMSCorrelationID i oferuje marginalną poprawę wydajności dla JMSMessageID.

Korzystanie z złożonych selektorów

Selektory mogą zawierać wiele komponentów, na przykład:

a and b or c and d or e and `f` or g and h or `i` and `j` ... lub y i z

Użycie takich złożonych selektorów może mieć poważne konsekwencje w zakresie wydajności i spowodować nadmierne zapotrzebowanie na zasoby. W związku z tym produkt IBM WebSphere MQ będzie zabezpieczał system przez niepowodzenie przetwarzania nadmiernie złożonych selektorów, które mogą spowodować niedobór zasobów systemowych. Ochrona może wystąpić po około 100 testach na niektórych platformach, więc selektory zbliżające się do tej liczby komponentów mogą widzieć awarie. Zaleca się, aby stosowanie selektorów z wieloma komponentami zostało dokładnie wypróbowanych i przetestowanych na odpowiednich platformach w celu zapewnienia, że nie zostaną osiągnięte limity ochronne.

Wydajność i złożoność selektorów można poprawić, upraszczając je przy użyciu dodatkowego nawiasu łączącego komponenty. Na przykład:

(a i b lub c i d) lub (e i `f` lub g i h) lub (`i` i `j`) ...

Pojęcia pokrewne

Składnia selektora komunikatów
Selektor komunikatów produktu WebSphere MQ jest łańcuchem składniowym, który jest oparty na podzbiorze składni wyrażeń warunkowych SQL92 .

Wybieranie treści komunikatu
Istnieje możliwość zasubskrybowania na podstawie wyboru zawartości ładunku komunikatu (zwanej również filtrowaniem treści), ale decyzja o tym, które komunikaty powinny zostać dostarczone do takiej subskrypcji, nie może być wykonywana bezpośrednio przez produkt WebSphere MQ. Zamiast tego wymagany jest rozszerzony dostawca wyboru komunikatów, na przykład IBM Integration Bus, w celu przetworzenia komunikatów.

Składnia selektora komunikatów
Selektor komunikatów produktu WebSphere MQ jest łańcuchem składniowym, który jest oparty na podzbiorze składni wyrażeń warunkowych SQL92 .

Kolejność, w jakiej selektor komunikatów jest wartościowany, jest od lewej do prawej w obrębie poziomu priorytetu. Aby zmienić to zamówienie, można użyć nawiasów. Predefiniowane literały selektora i nazwy operatorów są zapisywane w tym miejscu wielkimi literami, jednak nie są one w nich rozróżniane.

Produkt WebSphere MQ weryfikuje poprawność składową selektora komunikatów w momencie jego prezentacji. Jeśli składnia łańcucha wyboru jest niepoprawna lub nazwa właściwości nie jest poprawna, a dostawca wyboru komunikatów rozszerzonych jest niedostępny, produkt `MQRC_SELECTION_NOT_AVAILABLE` zostanie zwrócony do aplikacji. Jeśli składnia łańcucha wyboru jest niepoprawna lub nazwa właściwości nie jest poprawna, gdy subskrypcja jest wznawiana, do aplikacji jest zwracany `MQRC_SELECTOR_SYNTAX_ERROR` . Jeśli sprawdzanie poprawności nazwy właściwości zostało wyłączone podczas ustawiania właściwości (przez ustawienie `MQCMHO_NONE` zamiast `MQCMHO_VALIDATE`), a następnie aplikacja umieszcza komunikat z niepoprawną nazwą właściwości, ten komunikat nigdy nie zostanie wybrany.

Selektor może zawierać:

- Literały:
  - Literały łańcuchowe są ujęte w pojedynczy cudzysłów. Dwa kolejne apostrofowe znaki cudzysłowu reprezentują pojedynczy cudzysłów. Przykładami są literały i literał. Podobnie jak literały łańcuchowe języka Java, używają one kodowania znaków Unicode. Nie można używać podwójnych cudzysłowów, aby ująć literał łańcuchowy. Każda sekwencja bajtów może być używana między pojedynczymi znakami cudzysłowu.

– Łańcuch bajtowy to co najmniej jedna para znaków szesnastkowych ujęta w podwójny cudzysłów i poprzedzona przedrostkiem 0x. Przykłady: "0x2F1C" lub "0XD43A". Długość łańcucha bajtów musi wynosić co najmniej jeden bajt. Jeśli łańcuch bajtowy selektora jest dopasowany do właściwości komunikatu typu MQTYPE_BYTE_STRING, żadne działanie specjalne nie jest podejmowane w przypadku zera wiodącego lub końcowego. Bajty są traktowane jako inny znak. Nie rozważa się również endianness. Długość łańcuchów bajtów selektora i właściwości musi być taka sama, a sekwencja bajtów musi być taka sama.

  Przykłady wyboru łańcuchów bajtów (przyjęto, że *myBytes* = 0AFC23), które są zgodne:

  - "myBytes = "0x0AFC23"" = TRUE

  Następujące wybory łańcuchów nie są zgodne:

  - "myBytes = "0xAFC23"" = MQRC_SELECTOR_SYNTAX_ERROR (ponieważ liczba bajtów nie jest wielokrotnością dwóch)
  - "myBytes = "0x0AFC2300"" = FALSE (ponieważ końcowe zero jest istotne w porównaniu)
  - "myBytes = "0x000AFC23"" = FALSE (ponieważ wiodące zero jest istotne w porównaniu)
  - "myBytes = "0x23FC0A"" = FALSE (ponieważ nie jest brane pod uwagę endianness)

– Liczby szesnastkowe zaczynają się od zera, po którym następują wielkie lub małe litery x. Pozostała część literału zawiera jeden lub więcej poprawnych znaków szesnastkowych. Przykłady: 0xA, 0xAF, 0X2020.
– Wiodąca wartość zero, po której następuje jedna lub więcej cyfr z zakresu 0-7, jest zawsze interpretowana jako początek liczby ósemkowej. Nie można reprezentować liczby dziesiętnej z przedrostkiem zero, takiej jak ta, na przykład, 09 zwraca błąd składniowy, ponieważ 9 nie jest poprawną cyfrą ósemkową. Przykłady liczb ósemkowych to 0177, 0713.
– Dokładny literał liczbowy jest wartością liczbową bez przecinka dziesiętnego, np. 57, -957i +62. Dokładna literał liczbowy może zawierać wielkie lub małe litery L; nie ma to wpływu na sposób przechowywania lub interpretowania liczby. Produkt WebSphere MQ obsługuje dokładne liczby z zakresu od -9,223,372,036,854,775,808 do 9,223,372,036,854,775,807.
– Przybliżony literał liczbowy jest wartością liczbową w zapisie naukowym, takim jak 7E3 lub -57.9E2, lub wartością liczbową z dziesiętnym, takim jak 7., -95.7lub +6.2. Produkt WebSphere MQ obsługuje liczby z zakresu od -1.797693134862315E+308 do 1.797693134862315E+308.

  Znacząca wartość powinna być zgodna z opcjonalnym znakiem (+ lub -). Wartość ta powinna być liczbą całkowitą lub ułamkową. Ułamkowa część znaczeń nie musi mieć wiodącej cyfry.

  Wielkie lub małe litery E wskazują na początek opcjonalnego wykładnika. Wykładnik ma wartość dziesiętną, a część wykładnika może być poprzedzona opcjonalnym znakiem znaku.

  Przybliżone literały liczbowe mogą zostać zakończone przez znak F lub D (bez rozróżniania wielkości liter). Ta składnia istnieje w celu obsługi metody języka znaczników o numerach pojedynczych lub podwójnych precyzji. Te znaki są opcjonalne i nie mają wpływu na sposób przechowywania lub przetwarzania literału liczbowego przybliżonego. Liczby te są zawsze zapisywane i przetwarzane za pomocą podwójnej precyzji.
– Literały boolowskie TRUE i FALSE.

Uwaga: Nieskończone reprezentacje IEEE-754 , takie jak NaN, +Infinityi -Infinity , nie są obsługiwane w łańcuchach wyboru. Dlatego nie jest możliwe użycie tych wartości jako operandów w wyrażeniu. Wartość ujemna zero jest traktowana tak samo jak dodatnie zero dla operacji matematycznych.

- Identyfikatory:

  Identyfikator jest sekwencją znaków o zmiennej długości, która musi rozpoczynać się od poprawnego znaku początkowego identyfikatora, po którym następuje zero lub więcej poprawnych znaków części identyfikatora. Reguły dla nazw identyfikatorów są takie same jak reguły dla nazw właściwości komunikatów, patrz “Nazwy właściwości” na stronie 19 i “Ograniczenia dotyczące nazw właściwości” na stronie 20 , aby uzyskać więcej informacji.

Uwaga: Wybór może być wykonywany na ładunku komunikatu tylko wtedy, gdy dostępny jest rozszerzony dostawca wyboru komunikatów.

Identyfikatory są odwołaniami do pól nagłówka lub odwołaniami do właściwości. Typ wartości właściwości w selektorze komunikatów musi odpowiadać typowi użytej do ustawienia właściwości, chociaż w miarę możliwości wykonywana jest promocja numeryczna. Jeśli wystąpi niezgodność typów, wynikiem wyrażenia jest FALSE. Jeśli przywoływana jest właściwość, która nie istnieje w komunikacie, jej wartością jest NULL.

Konwersje typów, które mają zastosowanie do metod pobierania dla właściwości, nie mają zastosowania, jeśli właściwość jest używana w wyrażeniu selektora komunikatów. Jeśli na przykład właściwość zostanie ustawiona jako wartość łańcuchowa, a następnie zostanie użyta selektor w celu odpytania zapytania jako wartości liczbowej, wyrażenie zwróci wartość FALSE.

Nazwy pól i właściwości JMS, które są odwzorowywać na nazwy właściwości lub nazwy pól MQMD , są również poprawnymi identyfikatorami w łańcuchu wyboru. Produkt WebSphere MQ odwzorowuje rozpoznane pole JMS i nazwy właściwości na wartości właściwości komunikatu. Więcej informacji zawiera sekcja “Selektory komunikatów w usłudze JMS” na stronie 830. Jako przykład, łańcuch wyboru `"JMSPriority >="` wybiera właściwość `Pri` znalezionej w folderze `jms` bieżącego komunikatu.

- Przepełnienie/niedomiaru:

  W przypadku liczb dziesiętnych i przybliżonych liczb numerycznych nie są zdefiniowane następujące wartości:

  – Określanie liczby, która jest poza zdefiniowanym zakresem
  – Określanie wyrażenia arytmetycznego, które spowodowałoby przepełnienie lub niedomik.

  Dla tych warunków nie są wykonywane żadne sprawdzenia.

- Białe znaki:

  Zdefiniowana jako spacja, znak nowego wiersza, znak nowego wiersza, znak powrotu karetki, karta pozioma lub pionowa karta. Następujące znaki Unicode są rozpoznawane jako białe znaki:

  – `\u0009 to \u000D`
  – `\u0020`
  – `\u001C`
  – `\u001D`
  – `\u001E`
  – `\u001F`
  – `\u1680`
  – `\u180E`
  – `\u2000` do `\u200A`
  – `\u2028`
  – `\u2029`
  – `\u202F`
  – `\u205F`
  – `\u3000`

- Wyrażenia:

  – Selektor jest wyrażeniem warunkowym. Selektor wartościowany do prawdziwych dopasowań; selektor, którego wartościowanie ma wartość false lub nieznany, nie jest zgodny.
  – Wyrażenia arytmetyczne składają się z samych siebie, operacji arytmetycznych, identyfikatorów (wartość identyfikatora jest traktowana jako literał liczbowy) oraz literałów liczbowych.
  – Wyrażenia warunkowe składają się z samych siebie, operacji porównania i operacji logicznych.

- Standard bracketing (), aby ustawić kolejność, w jakiej wyrażenia są wartościowane, jest obsługiwane.

- Operatory logiczne w kolejności kolejności wykonywania: NOT, AND, OR.
- Operatory porównania: =, >, >=, <, <=, <> (nie są równe).
  - Dwa łańcuchy bajtów są równe tylko wtedy, gdy łańcuchy mają taką samą długość, a sekwencja bajtów jest równa.
  - Porównywane są tylko wartości tego samego typu. Jedynym wyjątkiem jest to, że poprawne jest porównanie dokładnych wartości liczbowych i przybliżonych wartości liczbowych, (wymagana konwersja typów jest zdefiniowana przez reguły promocji numerycznej języka Java). Jeśli istnieje próba porównania różnych typów, selektor zawsze ma wartość false.
  - Porównywanie łańcuchowe i boolowskie jest ograniczone do produktów = i <>. Dwa łańcuchy są równe tylko wtedy, gdy zawierają taką samą sekwencję znaków.
- Operatory arytmetyczne w kolejności kolejności wykonywania zadań:
  - +, - unary.
  - * mnożenie i / .
  - + oraz - odejmowanie.
  - Operacje arytmetyczne na wartości NULL nie są obsługiwane. Jeśli są one uruchomione, pełny selektor zawsze ma wartość false.
  - Operacje arytmetyczne muszą używać promocji numerycznej języka Java.
- Operator porównania arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 i arithmetic-expr3 :
  - `Age BETWEEN 15 and 19` jest odpowiednikiem `age >= 15 AND age <= 19`.
  - `Age NOT BETWEEN 15 and 19` jest odpowiednikiem `age < 15 OR age > 19`.
  - Jeśli dowolna z wyrażeń operacji BETWEEN ma wartość NULL, wartością tej operacji jest false (fałsz). Jeśli dowolna z wyrażeń operacji NOT BETWEEN to NULL, wartość tej operacji jest prawdziwa.
- identyfikator `[NOT] IN (string-literal1, string-literal2,...)` operator porównania, gdzie identyfikator ma wartość typu String lub NULL .
  - Wartość `Country IN ('UK', 'US', 'France')` ma wartość true dla `'UK'` i false dla `'Peru'`. Jest on równoważny z wyrażeniem `(Country = 'UK') OR (Country = 'US') OR (Country = 'France')`.
  - `Country NOT IN ('UK', 'US', 'France')` ma wartość false dla `'UK'` i ma wartość true dla `'Peru'`. Jest on równoważny z wyrażeniem `NOT ((Country = 'UK') OR (Country = 'US') OR (Country = 'France'))`.
  - Jeśli identyfikator operacji IN lub NOT IN ma wartość NULL, wartość tej operacji jest nieznana.
- `identifier [NOT] LIKE` *`pattern-value`* `[ESCAPE` *`escape-character`*`]` , operator porównania, gdzie `identifier` ma wartość łańcuchową. *wartość-wzorca* jest literałem łańcuchowym, gdzie _ oznacza dowolny pojedynczy znak, a % oznacza dowolną sekwencję znaków (włącznie z pustą sekwencją). Wszystkie inne postacie stoją dla siebie. Opcjonalny *znak zmiany znaczenia* to pojedynczy literał łańcuchowy, który jest używany do zmiany znaczenia specjalnego znaczenia _ i % w *wartości wzorca*. Operator LIKE musi być używany tylko do porównywania dwóch wartości łańcuchowych.
  - Wartość `phone LIKE '12%3'` ma wartość true dla 123 i 12993, a wartość false dla 1234.
  - `word LIKE 'l_se'` ma wartość true w przypadku utraty i wartości false dla poluzowania.
  - Wartość `underscored LIKE '\_%' ESCAPE '\'` ma wartość true dla `_foo` i false dla `bar`.
  - `phone NOT LIKE '12%3'` ma wartość false dla 123 oraz 12993 i ma wartość true dla 1234.
  - Jeśli identyfikator operacji LIKE lub NOT LIKE ma wartość NULL, wartość tej operacji jest nieznana.

  Uwaga: Operator LIKE musi być używany do porównywania dwóch wartości łańcuchowych. Wartość `Root.MQMD.CorrelId` to 24-bajtowa tablica bajtów, a nie łańcuch znaków. Łańcuch selektora `Root.MQMD.CorrelId LIKE 'ABC%'` jest akceptowany przez analizator składni jako poprawny pod względem składniowym, ale jest wartościowany do wartości false. W przypadku porównywania tablicy bajtów z łańcuchem znaków program LIKE nie może być w tym przypadku używany.

- Testy operatora porównania `identifier IS NULL` dla wartości pola nagłówka `NULL` lub brakującej wartości właściwości.
- Operator porównania `identifier IS NOT NULL` testuje istnienie wartości pola nagłówka o wartości innej niż NULL lub wartości właściwości.
- Wartości NULL

  Wartościowanie wyrażeń selektora, które zawierają wartości `NULL` , jest definiowane przez semantykę języka SQL 92 `NULL` w podsumowaniu:

  – Język SQL traktuje wartość `NULL` jako nieznaną.
  – Porównanie lub arytmetyka z nieznaną wartością zawsze daje nieznaną wartość.
  – Operatory `IS NULL` i `IS NOT NULL` przekształcają nieznaną wartość w wartości `TRUE` i `FALSE` .

  Operatory boolowskie używają logiki trójwartościowej (`T=TRUE, F=FALSE, U=UNKNOWN`)

Tabela 1. Wynik operatora boolowskiego, gdy logika jest A AND B

Operator A	**Operator B**	**Wynik (A I B)**
T	F	F
T	U	U
T	T	T
F	T	F
F	U	F
F	F	F
U	T	U
U	U	U
U	F	F

Tabela 2. Wynik operatora boolowskiego, gdy logika jest A OR B

Operator A	**Operator B**	**Wynik (A OR B)**
T	F	T
T	U	T
T	T	T
F	T	T
F	U	U
F	F	F
U	T	T
U	U	U
U	F	U

Tabela 3. Wynik operatora boolowskiego, gdy logika jest NOT A

Operator A	**Wynik (NOT A)**
T	F
F	T

Tabela 3. Wynik operatora boolowskiego, gdy logika jest NOT A (kontynuacja)	
Operator A	**Wynik (NOT A)**
U	U

Poniższy selektor komunikatów wybiera komunikaty z typem komunikatu samochodu, koloru niebieskiego i wagą większą niż 2500 funtów:

```
"JMSType = 'car' AND color = 'blue' AND weight > 2500"
```

Chociaż SQL obsługuje stałe porównania dziesiętne i arytmetyczne, selektory komunikatów nie są dostępne. Dlatego dokładne literały liczbowe są ograniczone do tych bez liczby dziesiętnej. Dlatego też istnieją cyfry z separatą dziesiętną jako alternatywna reprezentacja dla przybliżonej wartości liczbowej.

Komentarze SQL nie są obsługiwane.

Pojęcia pokrewne

Zachowanie wyboru
Przegląd funkcji wyboru produktu IBM WebSphere MQ .

Wybieranie treści komunikatu
Istnieje możliwość zasubskrybowania na podstawie wyboru zawartości ładunku komunikatu (zwanej również filtrowaniem treści), ale decyzja o tym, które komunikaty powinny zostać dostarczone do takiej subskrypcji, nie może być wykonywana bezpośrednio przez produkt WebSphere MQ. Zamiast tego wymagany jest rozszerzony dostawca wyboru komunikatów, na przykład IBM Integration Bus, w celu przetworzenia komunikatów.

“Właściwości komunikatu” na stronie 18
Za pomocą właściwości komunikatu można zezwolić aplikacji na wybór komunikatów do przetwarzania lub pobieranie informacji o komunikacie bez uzyskiwania dostępu do nagłówków MQMD lub MQRFH2 . Ułatwiają one także komunikację między aplikacjami WebSphere MQ i JMS.

Odsyłacze pokrewne

MsgHandle

MQBUFMH-przekształcanie buforu w uchwyt komunikatu

Reguły i ograniczenia dotyczące łańcuchów wyboru
Zapoznanie się z tymi regułami dotyczącymi sposobu interpretowania łańcuchów wyboru i ograniczenia znaków w celu uniknięcia potencjalnych problemów podczas korzystania z selektorów.

- Równoważność jest testowana przy użyciu pojedynczego znaku równości, na przykład `a = b` jest poprawna, podczas gdy `a == b` jest niepoprawna.
- Operatorem używanym przez wiele języków programowania do reprezentowania "not equal to" jest `!=`. Ta reprezentacja nie jest poprawnym synonimem produktu <>, na przykład `a <> b` jest poprawna, natomiast `a != b` nie jest poprawna.
- Pojedyncze cudzysłowy są rozpoznawane tylko wtedy, gdy " (U+0027) jest używany znak. Podobnie, podwójne cudzysłowy, poprawne tylko wtedy, gdy są używane do ujmowania łańcuchów bajtowych, muszą używać znaku " (U+0022) znak.
- Symbole &, &&, | i || nie są synonimami dla logicznego łączenia/rozłączenia; na przykład `a && b` musi być określony jako `a AND b`.
- Znaki wieloznaczne * i ? nie są synonimami dla produktów % i _.
- Selektory zawierające wyrażenia złożone, takie jak `20 < b < 30` , nie są poprawne. Analizator składni ocenia operatorów, którzy mają ten sam priorytet od lewej do prawej. W związku z tym przykładem może być `(20 < b) < 30`, co nie ma sensu. Zamiast tego wyrażenie musi być zapisane jako `(b > 20) AND (b < 30)`.
- Łańcuchy bajtowe muszą być ujęte w podwójny cudzysłów; jeśli używane są pojedyncze cudzysłowy, to łańcuch bajtowy jest przyjmowany jako literał łańcuchowy. Liczba znaków (nie liczba znaków reprezentowanych przez znaki) po `0x` musi być wielokrotnością dwóch znaków.

- Słowo kluczowe IS nie jest synonimem znaku równości. W ten sposób łańcuchy wyboru `a IS 3` i `b IS 'red'` nie są poprawne. Słowo kluczowe IS istnieje tylko w celu obsługi przypadków `IS NULL` i `IS NOT NULL`.

Pojęcia pokrewne

Uwagi dotyczące UTF-8 i Unicode podczas korzystania z selektorów komunikatów

Uwagi dotyczące UTF-8 i Unicode podczas korzystania z selektorów komunikatów

Znaki, które nie są ujęte w pojedynczy cudzysłów, które tworzą zastrzeżone słowa kluczowe łańcucha wyboru, muszą zostać wprowadzone w języku Basic Latin Unicode (od znaku U+0000 do U+0007F). Użycie innych reprezentacji znaków alfanumerycznych nie jest poprawne. Na przykład liczba 1 musi być wyrażona jako U+0031 w kodzie Unicode, nie jest ona poprawna, aby używać ekwiwalentu Fullwidth Digit U+FF11 lub arabskiego odpowiednika U+0661.

Nazwy właściwości komunikatów mogą być określane przy użyciu dowolnej poprawnej sekwencji znaków Unicode. Nazwy właściwości komunikatów zawarte w łańcuchach wyboru, które są kodowane w UTF-8 , będą sprawdzane nawet wtedy, gdy zawierają znaki wielobajtowe. Sprawdzanie poprawności wielobajtowych UTF-8 jest ścisłe i należy upewnić się, że do nazw właściwości komunikatów używane są poprawne sekwencje UTF-8 .

Podczas porównywania pod kątem równości nie jest wykonywane żadne dodatkowe przetwarzanie w nazwach właściwości ani wartościach. Oznacza to na przykład, że nie ma miejsca przed/de-kompozycja i ligatury nie są podane w sposób szczególny. Na przykład wstępnie złożony znak umlaut U+00FC nie jest uznawany za równoważny U+0075 + U+0308 , a sekwencja znaków nie jest uznawana za odpowiednik kodu Unicode U+FB00 (LATIN SMALL LIGATURE FF).

Dane właściwości ujęte w pojedynczy cudzysłów mogą być reprezentowane przez dowolną sekwencję bajtów i nie jest sprawdzana poprawność.

Pojęcia pokrewne

Reguły i ograniczenia dotyczące łańcuchów wyboru
Zapoznanie się z tymi regułami dotyczącymi sposobu interpretowania łańcuchów wyboru i ograniczenia znaków w celu uniknięcia potencjalnych problemów podczas korzystania z selektorów.

Wybieranie treści komunikatu
Istnieje możliwość zasubskrybowania na podstawie wyboru zawartości ładunku komunikatu (zwanej również filtrowaniem treści), ale decyzja o tym, które komunikaty powinny zostać dostarczone do takiej subskrypcji, nie może być wykonywana bezpośrednio przez produkt WebSphere MQ. Zamiast tego wymagany jest rozszerzony dostawca wyboru komunikatów, na przykład IBM Integration Bus, w celu przetworzenia komunikatów.

Gdy aplikacja publikuje w łańcuchu tematu, w którym co najmniej jeden subskrybent ma wybrany łańcuch wyboru w treści komunikatu, produkt WebSphere MQ zażądał, aby dostawca wyboru komunikatów rozszerzonych przeanalizował publikację i poinformował WebSphere MQ , czy publikacja jest zgodna z kryteriami wyboru określonymi przez każdy subskrybent z filtrem treści.

Jeśli rozszerzony dostawca wyboru komunikatów określa, że publikacja jest zgodna z łańcuchem wyboru subskrybenta, komunikat będzie nadal dostarczany do subskrybenta.

Jeśli rozszerzony dostawca wyboru komunikatów stwierdzi, że publikacja nie jest zgodna, komunikat nie zostanie dostarczony do subskrybenta. Może to spowodować, że wywołanie MQPUT lub MQPUT1 nie powiedzie się, a kod przyczyny MQRC_PUBLICATION_FAILURE. Jeśli dostawca wyboru rozszerzonego komunikatu nie może przeanalizować publikacji, zwracany jest kod przyczyny MQRC_CONTENT_ERROR, a wywołanie MQPUT lub MQPUT1 nie powiodło się.

Jeśli dostawca wyboru rozszerzonego komunikatu nie jest dostępny lub nie może określić, czy subskrybent powinien otrzymać publikację, zwracany jest kod przyczyny MQRC_SELECTION_NOT_AVAILABLE, a wywołanie MQPUT lub MQPUT1 nie powiedzie się.

Gdy subskrypcja jest tworzona z użyciem filtru treści, a dostawca wyboru rozszerzonego komunikatu nie jest dostępny, wywołanie MQSUB kończy się niepowodzeniem z kodem przyczyny MQRC_SELECTION_NOT_AVAILABLE. Jeśli subskrypcja z filtrem treści jest wznawiana, a dostawca

wyboru komunikatów rozszerzonych jest niedostępny, wywołanie MQSUB zwróci ostrzeżenie o wartości MQRC_SELECTION_NOT_AVAILABLE, ale subskrypcja może zostać wznowiona.

Pojęcia pokrewne

Zachowanie wyboru
Przegląd funkcji wyboru produktu IBM WebSphere MQ .

Składnia selektora komunikatów
Selektor komunikatów produktu WebSphere MQ jest łańcuchem składniowym, który jest oparty na podzbiorze składni wyrażeń warunkowych SQL92 .

Asynchroniczne wykorzystanie komunikatów produktu IBM WebSphere MQ

Wykorzystanie asynchroniczne korzysta z zestawu rozszerzeń MQI (Message Queue Interface), wywołania MQI MQCB i MQCTL, które pozwalają na zapisywanie aplikacji MQI w celu odbierania komunikatów z zestawu kolejek. Komunikaty są dostarczane do aplikacji przez wywołanie 'jednostki kodu', identyfikowanej przez aplikację, przekazując komunikat lub leksem reprezentującym komunikat.

W najprostszych środowiskach aplikacji, 'jednostka kodu' jest definiowana przez wskaźnik funkcji, jednak w innych środowiskach 'jednostka kodu' może być zdefiniowana przez nazwę programu lub modułu.

W przypadku asynchronicznego korzystania z komunikatów używane są następujące terminy:

Konsument komunikatu
Konstrukcja programistyczny, która umożliwia zdefiniowanie programu lub funkcji, które mają być wywoływane z komunikatem, gdy jest on zgodny z wymaganiami aplikacji.

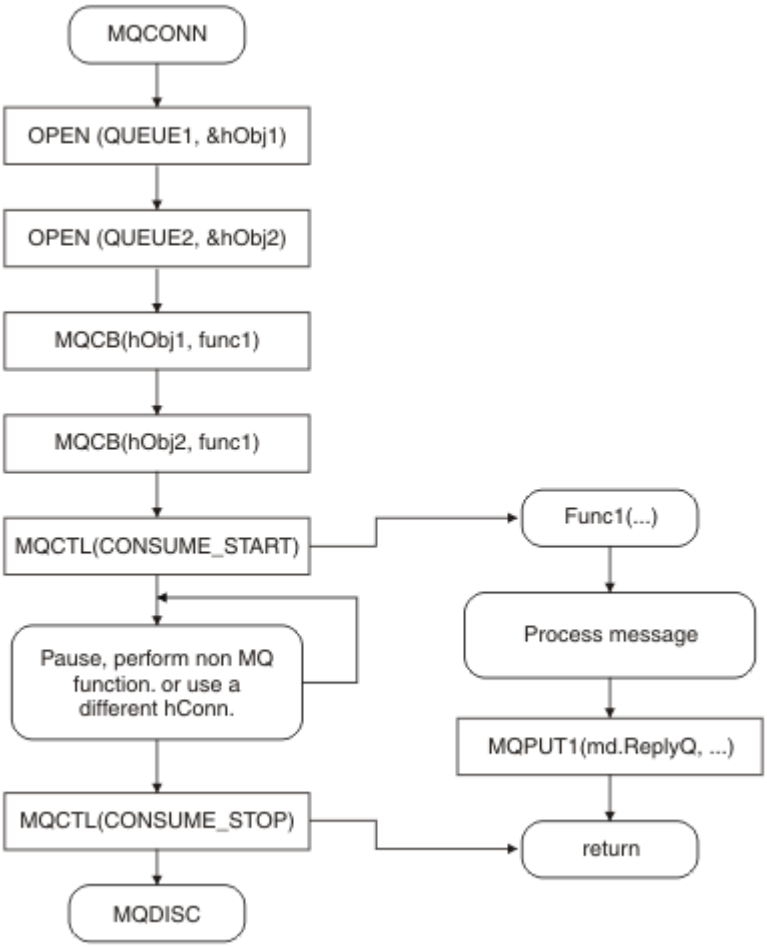
procedura obsługi zdarzeń
Konstrukcja programistyczny, która umożliwia zdefiniowanie programu lub funkcji, która ma zostać wywołana, gdy wystąpi zdarzenie asynchroniczne, takie jak wygaszanie menedżera kolejek.

Wywołanie zwrotne
Termin ogólny używany do odwołania się do procedury obsługi komunikatów lub do procedury obsługi zdarzeń.

Wykorzystanie asynchroniczne może uprościć projektowanie i implementację nowych aplikacji, w szczególności tych, które przetwarają wiele kolejek wejściowych lub subskrypcji. Jeśli jednak używana jest więcej niż jedna kolejka wejściowa, a komunikaty są przetwarzane w kolejności priorytetów, kolejność priorytetów jest obserwowana niezależnie w każdej kolejce: komunikaty o niskim priorytecie mogą być wysyłane z jednej kolejki przed komunikatami o wysokim priorytecie od drugiej. Kolejność komunikatów w wielu kolejkach nie jest gwarantowana. Należy również zauważyć, że jeśli używane są wyjścia funkcji API, może być konieczne ich zmianę w celu uwzględnienia wywołań MQCB i MQCTL.

Poniższe ilustracje dają przykład, w jaki sposób można użyć tej funkcji.

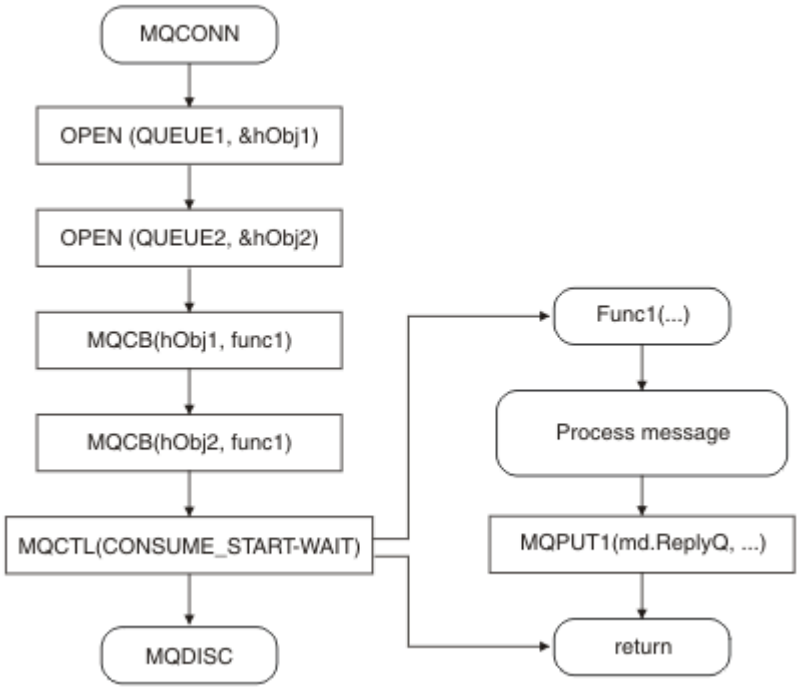
Program Rysunek 5 na stronie 35 wyświetla wielowątkową aplikację, która konsumuje komunikaty z dwóch kolejek. W przykładzie przedstawiono wszystkie komunikaty dostarczane do jednej funkcji.



Rysunek 5. Standardowa aplikacja sterowana komunikatami wykorzystująca dwie kolejki

Rysunek 6 na stronie 36 Ten przykładowy przepływ przedstawia jednowątkową aplikację, która konsumuje komunikaty z dwóch kolejek. W przykładzie przedstawiono wszystkie komunikaty dostarczane do jednej funkcji.

Różnica między przypadkiem asynchronicznym jest taka, że element sterujący nie wraca do wystawcy MQCTL, dopóki wszystkie konsumenty nie dezaktywują się. To oznacza, że jeden konsument wygenerował żądanie zatrzymania MQCTL lub wygaszanie menedżera kolejek.



Rysunek 6. Aplikacja jednowątkowa sterowana komunikatami korzystała z dwóch kolejek

Grupy komunikatów

Komunikaty mogą występować w grupach, aby zezwolić na porządkowanie komunikatów.

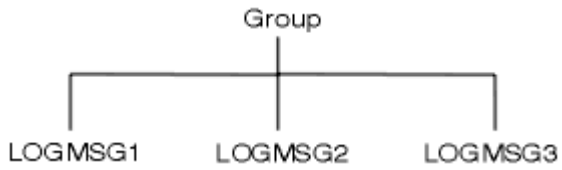
Grupy komunikatów umożliwiają oznaczanie wielu komunikatów jako powiązanych z nimi, a także kolejność logiczna, która ma być zastosowana do grupy (patrz sekcja “Uporządkowanie logiczne i fizyczne” na stronie 252). Na platformach innych niż z/OSpojęcie pokrewnej, “Segmentacja komunikatów” na stronie 270 umożliwia rozbicie dużych komunikatów na mniejsze segmenty. Podczas wprowadzania do tematu nie można używać zgrupowanych ani segmentowanych komunikatów.

Hierarchia w grupie jest następująca:

Grupa
Jest to najwyższy poziom w hierarchii i jest identyfikowany przez *GroupId*. Składa się on z jednego lub większej liczby komunikatów, które zawierają ten sam *GroupId*. Komunikaty te mogą być przechowywane w dowolnym miejscu w kolejce.

Uwaga: Termin *komunikat* jest używany w tym miejscu do oznaczenia jednego elementu w kolejce, na przykład zwracanego przez pojedynczą operację MQGET, która nie określa parametru MQGMO_COMPLETE_MSG.

Rysunek 7 na stronie 36 przedstawia grupę komunikatów logicznych:



Rysunek 7. Grupa komunikatów logicznych

Otwarcie kolejki i określenie parametru MQOO_BIND_ON_GROUP powoduje wymuszenie wysłania wszystkich komunikatów w grupie, które są wysyłane do tej kolejki, do tej samej instancji kolejki. Więcej informacji na temat opcji BIND_ON_GROUP znajduje się w sekcji Obsługa powinowactw komunikatów.

Komunikat logiczny

Komunikaty logiczne w grupie są identyfikowane za pomocą pól *GroupId* i *MsgSeqNumber* . *MsgSeqNumber* rozpoczyna się od 1 w przypadku pierwszego komunikatu w grupie, a jeśli komunikat nie znajduje się w grupie, to wartość pola wynosi 1.

Użyj komunikatów logicznych w grupie do:

- Upewnij się, że zamawiający (jeśli nie jest to gwarantowane w okolicznościach, w których wiadomość jest przesyłana).
- Zezwalaj aplikacjom na grupowanie podobnych komunikatów (na przykład te, które muszą być przetwarzane przez tę samą instancję serwera).

Każdy komunikat w grupie składa się z jednego komunikatu fizycznego, chyba że jest on podzielony na segmenty. Każdy komunikat jest logicznie osobną wiadomościami, a tylko pola *GroupId* i *MsgSeqNumber* w strukturze MQMD muszą zawierać dowolną relację z innymi komunikatami w grupie. Inne pola w strukturze MQMD są niezależne; niektóre z nich mogą być identyczne dla wszystkich komunikatów w grupie, podczas gdy inne mogą być różne. Na przykład komunikaty w grupie mogą mieć różne nazwy formatów, identyfikatory CCSID i kodowane.

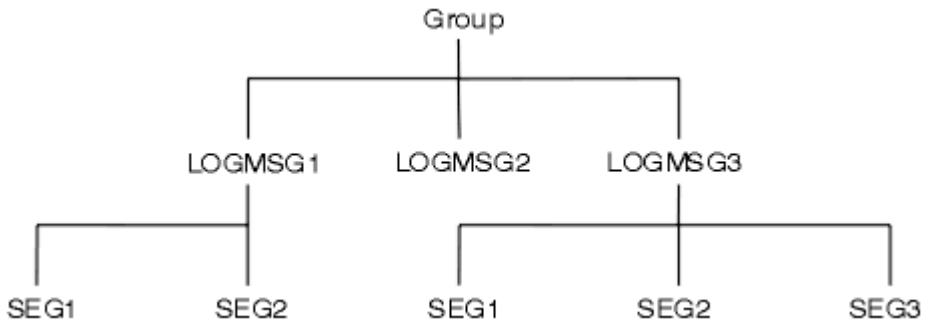
Segment

Segmenty są używane do obsługi komunikatów, które są zbyt duże dla umieszczania lub pobierania aplikacji lub menedżera kolejek (w tym między tymi menedżerami kolejek, przez które przekazywane są komunikaty). Więcej informacji na ten temat zawiera sekcja “Segmentacja komunikatów” na stronie 270.

Pojedynczy komunikat jest podzielony na mniejsze komunikaty o nazwie *segmenty*. Segment komunikatu jest identyfikowany przez pola *GroupId*, *MsgSeqNumber*i *Offset* . Pole *Offset* zaczyna się od zera dla pierwszego segmentu w komunikacie.

Każdy segment składa się z jednego komunikatu fizycznego, który może należeć do grupy (Rysunek 8 na stronie 37 przedstawia przykład komunikatów w grupie). Segment jest logicznie częścią pojedynczego komunikatu, dlatego tylko pola *MsgId*, *Offset*i *SegmentFlag* w strukturze MQMD powinny różnić się między oddzielnymi segmentami tego samego komunikatu. Jeśli nie uda się dojść do segmentu, kod przyczyny MQRC_INCOMPLETE_GROUP lub MQRC_INCOMPLETE_MSG jest zwracany jako odpowiedni.

Rysunek 8 na stronie 37 przedstawia grupę komunikatów logicznych, z których niektóre są segmentowane:



Rysunek 8. Posegmentowane komunikaty

Nie można używać segmentowanych lub zgrupowanych komunikatów z publikowania/subskrybowania.

Opis komunikatów logicznych i fizycznych znajduje się w sekcji “Uporządkowanie logiczne i fizyczne” na stronie 252. Więcej informacji na temat segmentacji komunikatów zawiera sekcja “Segmentacja komunikatów” na stronie 270.

Trwałość komunikatu

Komunikaty trwałe są zapisywane w dziennikach i plikach danych kolejki.

Jeśli menedżer kolejek jest restartowany po awarii, odtwarza te trwałe komunikaty w miarę potrzeb od zarejestrowanych danych. Komunikaty, które nie są trwałe, są usuwane w przypadku zatrzymania menedżera kolejek, bez względu na to, czy zatrzymanie jest wynikiem komendy operatora, czy też z powodu awarii pewnej części systemu.

Jeśli podczas tworzenia komunikatu zainicjowano deskryptor komunikatu (MQMD) przy użyciu wartości domyślnych, trwałość komunikatu jest pobierana z atrybutu *DefPersistence* kolejki określonego w komendzie MQOPEN. Można również ustawić trwałość komunikatu przy użyciu pola *Persistence* struktury MQMD, aby zdefiniować komunikat jako trwały lub nietrwały.

Wydajność aplikacji ma wpływ na komunikaty trwałe; zakres działania zależy od charakterystyki wydajności podsystemu we/wy komputera oraz sposobu korzystania z opcji punktu synchronizacji na każdej platformie:

- Trwały komunikat, poza bieżącą jednostką pracy, jest zapisywany na dysku przy każdym operacji umieszczania i pobierania. Patrz sekcja “Zatwierdzanie i wycofywanie jednostek pracy” na stronie 331.
- W produkcie IBM WebSphere MQ w systemach UNIX , IBM WebSphere MQ w systemach Linux , i IBM WebSphere MQ for Windows, trwały komunikat w bieżącej jednostce pracy jest rejestrowany tylko wtedy, gdy jednostka pracy jest zatwierdzana (a jednostka pracy może zawierać wiele operacji w kolejce).

Komunikaty nietrwałe mogą być używane do szybkiego przesyłania komunikatów. Więcej informacji na temat szybkich komunikatów zawiera sekcja Bezpieczeństwo komunikatów .

Uwaga: Kombinacja zapisu trwałych komunikatów w jednostce pracy oraz zapisywania trwałych komunikatów poza jednostką lub pracą może powodować potencjalnie poważne problemy z wydajnością aplikacji. Jest to szczególnie prawdziwe, gdy dla obu operacji używana jest ta sama kolejka docelowa.

Komunikaty, których dostarczenie nie powiodło się

Jeśli menedżer kolejek nie może umieścić komunikatu w kolejce, dostępne są różne opcje.

Można wykonać następujące czynności:

- Spróbuj ponownie umieścić komunikat w kolejce.
- Zażądaj, aby komunikat został zwrócony do nadawcy.
- Umieść komunikat w kolejce niedostarczonych komunikatów.

Więcej informacji na ten temat zawiera sekcja “Obsługa błędów programu” na stronie 561 .

Komunikaty, których kopie zapasowe zostały wycofane

W przypadku przetwarzania komunikatów z kolejki pod kontrolą jednostki pracy, jednostka pracy może składać się z jednego lub większej liczby komunikatów. W przypadku wystąpienia wycofania komunikaty, które zostały pobrane z kolejki, zostają przywrócone w kolejce i mogą być ponownie przetwarzane w innej jednostce pracy. Jeśli przetwarzanie konkretnego komunikatu jest przyczyną problemu, kopia zapasowa jednostki pracy jest wycofana ponownie. Może to spowodować pętlę przetwarzania. Komunikaty, które zostały umieszczone w kolejce, są usuwane z kolejki.

Aplikacja może wykrywać komunikaty, które są wychwytywać w takiej pętli, testując pole *BackoutCount* deskryptora MQMD. Aplikacja może poprawić sytuację lub wywołać ostrzeżenie dla operatora.

W produkcie WebSphere MQ for WebSphere MQ for Windows, WebSphere MQ on UNIX systems, WebSphere MQ on Linux systems liczba wycofanych operacji wycofania zawsze jest restartowana przez menedżera kolejek. Każda zmiana atrybutu *HardenGetBackout* jest ignorowana.

Więcej informacji na temat zatwierdzania i tworzenia kopii zapasowych komunikatów zawiera sekcja “Zatwierdzanie i wycofywanie jednostek pracy” na stronie 331.

Kolejka zwrotna i menedżer kolejek

Mogą wystąpić sytuacje, w których można odbierać komunikaty w odpowiedzi na komunikat, który jest wysyłany:

- Komunikat odpowiedzi w odpowiedzi na komunikat żądania
- Komunikat raportu na temat nieoczekiwanego zdarzenia lub utraty ważności
- Komunikat raportu o zdarzeniu COA (Potwierdzenie przyjazdu) lub COD (Potwierdzenie dostarczenia).
- Komunikat raportu o zdarzeniu PAN (Positive Action Notification) lub NAN (Negative Action Notification).

Przy użyciu struktury MQMD należy określić nazwę kolejki, do której mają być wysyłane komunikaty odpowiedzi i raporty w polu *ReplyToQ* . W polu *ReplyToQMgr* podaj nazwę menedżera kolejek, do którego należy kolejka odpowiedzi.

Jeśli pole *ReplyToQMgr* pozostanie puste, menedżer kolejek ustawia zawartość następujących pól w deskryptorze komunikatu w kolejce:

ReplyToQ
Jeśli *ReplyToQ* jest lokalną definicją kolejki zdalnej, pole *ReplyToQ* jest ustawione na nazwę kolejki zdalnej; w przeciwnym razie pole to nie jest zmieniane.

ReplyToQMgr
Jeśli *ReplyToQ* jest lokalną definicją kolejki zdalnej, pole *ReplyToQMgr* jest ustawione na nazwę menedżera kolejek, do którego należy kolejka zdalna. W przeciwnym razie pole *ReplyToQMgr* jest ustawione na nazwę menedżera kolejek, z którym połączona jest aplikacja.

Uwaga: Można zażądać, aby menedżer kolejek udostępnił więcej niż jedną próbę dostarczenia komunikatu, a następnie można zażądać, aby komunikat został odrzucony, jeśli nie powiedzie się. Jeśli komunikat, po niepowodzeniu dostarczenia, nie zostanie usunięty, zdalny menedżer kolejek umieszcza komunikat w swojej kolejce niedostarczonych komunikatów (komunikat niedostarczonych komunikatów) (patrz sekcja “Korzystanie z kolejki niedostarczonych komunikatów (niedostarczonych komunikatów)” na stronie 564).

Kontekst komunikatu

Informacja *Kontekst komunikatu* umożliwia aplikacji, która pobiera komunikat, w celu uzyskania informacji o inicjatorze komunikatu.

Pobieranie aplikacji może być następujące:

- Sprawdź, czy aplikacja wysyłający ma właściwy poziom uprawnień
- Wykonaj niektóre funkcje księgowe, aby móc pobierać aplikację wysyłający dla każdej pracy, którą musi wykonać.
- Przechowuj zapis kontrolny wszystkich komunikatów, z którymi współpracował

W przypadku użycia wywołania MQPUT lub MQPUT1 w celu umieszczenia komunikatu w kolejce można określić, że menedżer kolejek ma dodać pewne domyślne informacje kontekstu do deskryptora komunikatu. Aplikacje, które mają odpowiedni poziom uprawnień, mogą dodawać dodatkowe informacje kontekstowe. Więcej informacji na temat sposobu określania informacji o kontekście zawiera sekcja “Sterowanie informacjami kontekstową” na stronie 237.

Kontekst użytkownika jest używany przez menedżer kolejek podczas generowania następujących typów komunikatów raportu:

- Potwierdź przy dostarczeniu
- Utrata ważności

Gdy te komunikaty są generowane, kontekst użytkownika jest sprawdzany pod kątem uprawnień + put i + passid na miejscu docelowym raportu. Jeśli kontekst użytkownika ma niewystarczające uprawnienia, komunikat raportu jest umieszczany w kolejce niedostarczonych komunikatów, jeśli został on zdefiniowany. Jeśli nie ma kolejki niedostarczonych komunikatów, komunikat raportu jest odrzucany.

Wszystkie informacje kontekstowe są zapisywane w polach kontekstu deskryptora komunikatu. Typ informacji jest objęty tożsamością, pochodzeniem i informacjami o kontekście użytkownika.

kontekst tożsamości

Kontekst tożsamości -informacje identyfikują użytkownika aplikacji, która po raz pierwszy umiełała komunikat w kolejce. Odpowiednio autoryzowane aplikacje mogą ustawiać następujące pola:

- Menedżer kolejek wypełnia pole *UserIdentifier* nazwą identyfikującą użytkownika. Sposób działania menedżera kolejek zależy od środowiska, w którym aplikacja jest uruchomiona.
- Menedżer kolejek wypełnia pole *AccountingToken* znacznikiem lub numerem określonym w aplikacji, w której znajduje się komunikat.
- Aplikacje mogą korzystać z pola *ApplIdentityData* w celu uzyskania dodatkowych informacji, które mają zostać dołączone do użytkownika (na przykład zaszyfrowane hasło).

Identyfikator zabezpieczeń systemu Windows (SID) jest przechowywany w polu *AccountingToken* , gdy komunikat jest tworzony w produkcie WebSphere MQ for Windows. Identyfikator SID może zostać użyty do uzupełnienia pola *UserIdentifier* oraz do określenia informacji autoryzacyjnych użytkownika.

Informacje o tym, w jaki sposób menedżer kolejek wypełnia pola *UserIdentifier* i *AccountingToken* , można znaleźć w opisach tych pól w polach UserIdentifier i AccountingToken.

Aplikacje, które przekazują komunikaty z jednego menedżera kolejek do innego, powinny również przekazywać informacje o kontekście tożsamości, tak aby inne aplikacje znały tożsamość inicjatora komunikatu.

Kontekst źródłowy

Kontekst źródłowy zawiera opis aplikacji, która umieszczała komunikat w kolejce, w której komunikat jest *aktualnie* zapisany. Deskryptor komunikatu zawiera następujące pola dla informacji o kontekście pochodzenia:

PutApplType	Typ aplikacji, która umieszczała komunikat (na przykład transakcję CICS).
PutApplName	Nazwa aplikacji umieszczonej w komunikacie (na przykład nazwa zadania lub transakcji).
PutDate	Data umieszczenia komunikatu w kolejce.
PutTime	Godzina umieszczenia komunikatu w kolejce.
ApplOriginData	Wszelkie dodatkowe informacje, które aplikacja chce uwzględnić w związku z pochodzeniem komunikatu. Na przykład można ją ustawić za pomocą odpowiednio autoryzowanych aplikacji w celu wskazania, czy dane tożsamości są zaufane.

Informacje o kontekście pochodzenia są zwykle dostarczane przez menedżer kolejek. Średni czas Greenwich (GMT) jest używany dla pól *PutDate* i *PutTime* . Zapoznaj się z opisami tych pól w polach PutDate i PutTime.

Aplikacja z wystarczającą ilością uprawnień może udostępniać własny kontekst. Umożliwia to zachowanie informacji rozliczeniowych w przypadku, gdy jeden użytkownik ma inny identyfikator użytkownika w każdym z systemów przetwarzający komunikat, z którego pochodzą.

Obiekty produktu WebSphere MQ

W tej sekcji znajdują się szczegółowe informacje na temat obiektów WebSphere MQ , które obejmują: menedżery kolejek, grupy współużytkowania kolejek, kolejki, obiekty tematów administracyjnych, listy nazw, definicje procesów, obiekty informacji uwierzytelniających, kanały, klasy pamięci masowej, obiekty nasłuchiwania i usługi.

Menedżery kolejek definiują właściwości (znane jako atrybuty) tych obiektów. Wartości tych atrybutów wpływają na sposób, w jaki produkt WebSphere MQ przetwarza te obiekty. Z poziomu aplikacji do

sterowania tymi obiektami służy interfejs MQI (Message Queue Interface). Obiekty są identyfikowane przez *deskryptor obiektu* (MQOD), gdy jest on adresowany z programu.

W przypadku używania komend produktu WebSphere MQ do definiowania, modyfikowania lub usuwania obiektów, na przykład menedżer kolejek sprawdza, czy użytkownik posiada wymagany poziom uprawnień do wykonania tych operacji. Podobnie, jeśli aplikacja korzysta z wywołania MQOPEN w celu otwarcia obiektu, menedżer kolejek sprawdza, czy aplikacja ma wymagany poziom uprawnień, zanim zezwoli na dostęp do tego obiektu. Kontrole są przeprowadzane na podstawie nazwy otwieranego obiektu.

Pojęcia pokrewne

"Sterowanie informacjami kontekstową" na stronie 237
W przypadku użycia wywołania MQPUT lub MQPUT1 w celu umieszczenia komunikatu w kolejce można określić, że menedżer kolejek ma dodać pewne domyślne informacje kontekstu do deskryptora komunikatu. Aplikacje, które mają odpowiedni poziom uprawnień, mogą dodawać dodatkowe informacje kontekstowe. Do sterowania informacjami kontekstowych można użyć pola opcji w strukturze MQPMO.

Odsyłacze pokrewne

"Opcje MQOPEN związane z kontekstem komunikatu" na stronie 228
Aby możliwe było powiązanie informacji kontekstowych z komunikatem podczas umieszczania go w kolejce, należy użyć jednej z opcji kontekstu komunikatu podczas otwierania kolejki.

Przygotowywanie i uruchamianie aplikacji serwera Microsoft Transaction Server

Aby przygotować aplikację MTS do uruchamiania jako aplikacja kliencka MQI produktu WebSphere MQ , należy postępować zgodnie z poniższa instrukcjami, aby uzyskać informacje o środowisku.

Ogólne informacje na temat tworzenia aplikacji Microsoft Transaction Server (MTS), które uzyskują dostęp do zasobów produktu WebSphere MQ , można znaleźć w sekcji dotyczącej MTS w Centrum pomocy produktu WebSphere MQ .

Aby przygotować aplikację MTS do uruchamiania jako aplikacja kliencka MQI produktu WebSphere MQ , wykonaj jedną z następujących czynności dla każdego komponentu aplikacji:

- Jeśli komponent używa powiązań języka C dla interfejsu MQI, należy postępować zgodnie z instrukcjami w sekcji "Przygotowywanie programów w języku C w systemie Windows" na stronie 471 , ale należy połączyć komponent z biblioteką mqicxa.lib zamiast z biblioteką mqic.lib.
- Jeśli komponent korzysta z klas języka C++ WebSphere MQ , należy postępować zgodnie z instrukcjami w sekcji "Budowanie programów C++ w systemie Windows" na stronie 669 , ale należy połączyć ten komponent z biblioteką imqx23vn.lib zamiast imqc23vn.lib.
- Jeśli komponent używa powiązań języka Visual Basic dla interfejsu MQI, należy postępować zgodnie z instrukcjami w "Przygotowywanie programów Visual Basic w systemie Windows" na stronie 475 , ale po zdefiniowaniu projektu Visual Basic należy wpisać `MqType=3` w polu **Warunkowe argumenty kompilacji** .
- Jeśli komponent korzysta z klas automatyzacji produktu WebSphere MQ dla ActiveX (MQAX), należy zdefiniować zmienną środowiskową GMQ_MQ_LIB o wartości `mqic32xa.dll` .

  Można zdefiniować zmienną środowiskową z poziomu aplikacji lub zdefiniować ją w taki sposób, aby jej zasięg był szeroki. Jednak zdefiniowanie go jako całego systemu może spowodować, że dowolna istniejąca aplikacja MQAX, która nie definiuje zmiennej środowiskowej z aplikacji, zachowuje się niepoprawnie.

Używanie produktu IBM WebSphere MQ z produktem WebSphere Application Server

W tym temacie opisano sposób korzystania z produktu IBM WebSphere MQ z produktem WebSphere Application Server.

Aplikacje napisane w produkcie Java , które działają w produkcie WebSphere Application Server , mogą używać specyfikacji usługi przesyłania komunikatów (JMS) produktu Java do przesyłania komunikatów.

Przesyłanie komunikatów typu punkt z punktem w tym środowisku może być udostępniane przez menedżer kolejek produktu IBM WebSphere MQ .

Zaletą korzystania z menedżera kolejek produktu IBM WebSphere MQ w celu zapewnienia przesyłania komunikatów w trybie punkt z punktem jest to, że łączenie aplikacji JMS może w pełni uczestniczyć w funkcjach sieci produktu IBM WebSphere MQ , co umożliwia aplikacjom wymianę komunikatów z menedżerami kolejek uruchomionym na wielu platformach.

Aplikacje mogą korzystać z *transportu klienta* lub *transportu powiązań* dla obiektu fabryki połączeń kolejki. W przypadku *transportu powiązań* menedżer kolejek musi istnieć lokalnie w aplikacji, która wymaga połączenia. Jeśli menedżer kolejek nie jest lokalny względem aplikacji, należy zainstalować *Załącznik klienta* , aby umożliwić aplikacji nawiązanie połączenia z menedżerem kolejek działającym na innym komputerze lub obrazie.

Domyślnie komunikaty JMS przechowywane w kolejkach produktu IBM WebSphere MQ korzystają z nagłówka MQRFH2 w celu przechowywania niektórych informacji nagłówka komunikatu JMS. Wiele wcześniejszych aplikacji produktu IBM WebSphere MQ nie może przetwarzać komunikatów z tymi nagłówkami i wymagać ich własnych, charakterystycznych nagłówków, na przykład MQCIH for CICS Bridge lub MQWIH dla aplikacji IBM WebSphere MQ Workflow. Więcej szczegółowych informacji na temat tych szczególnych uwag zawiera sekcja “Odwzorowywanie komunikatów JMS na komunikaty produktu WebSphere MQ” na stronie 833.

Scenariusze obsługi transakcyjnej

Dzięki obsłudze transakcyjnej można umożliwić aplikacjom niezawodne działanie z bazami danych.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

W tej sekcji przedstawiono obsługę transakcyjną. Praca wymagana w celu umożliwienia aplikacjom korzystania z produktu IBM WebSphere MQ z produktem bazodanowym obejmuje obszary programowania aplikacji i administrowania systemem. Informacje podane w tym miejscu można znaleźć razem z produktem “Zatwierdzanie i wycofywanie jednostek pracy” na stronie 331.

Rozpoczynamy od wprowadzenia jednostek pracy, które tworzą transakcje, a następnie opisujemy sposoby, w jaki można włączyć IBM WebSphere MQ do koordynowania transakcji z bazami danych.

Pojęcia pokrewne

“Wprowadzenie jednostek pracy” na stronie 42
W tej sekcji przedstawiono ogólne pojęcia dotyczące jednostki pracy, zatwierdzania, wycofania i synchronizacji. Zawiera również dwa scenariusze ilustrujące globalne jednostki pracy.

IBM WebSphere MQ i HP NonStop TMF

Wprowadzenie jednostek pracy

W tej sekcji przedstawiono ogólne pojęcia dotyczące jednostki pracy, zatwierdzania, wycofania i synchronizacji. Zawiera również dwa scenariusze ilustrujące globalne jednostki pracy.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Gdy program umieszcza komunikaty w kolejkach w ramach jednostki pracy, komunikaty te są widoczne dla innych programów tylko wtedy, gdy program *zatwierdzil* jednostkę pracy. Aby zatwierdzić jednostkę pracy, wszystkie aktualizacje muszą być pomyślne, aby zachować integralność danych.

Jeśli program wykryje błąd i zdecyduje, że operacja put nie zostanie trwale wykonana, może on *wycofać* jednostkę pracy. Gdy program wykonuje wycofany program, program WebSphere MQ odtwarza kolejki, usuwając komunikaty umieszczone w kolejkach przez tę jednostkę pracy.

Podobnie, gdy program pobiera komunikaty z jednej lub większej liczby kolejek w jednostce pracy, komunikaty te pozostają w kolejkach do momentu zatwierdzenia przez program jednostki pracy, ale

komunikaty te nie są dostępne do pobrania przez inne programy. Komunikaty są trwale usuwane z kolejek, gdy program zatwierdza jednostkę pracy. Jeśli program utworzy kopię zapasową jednostki pracy, program WebSphere MQ odtwarza kolejki, udostępniając komunikaty do pobrania przez inne programy.

Decyzja o zatwierdzeniu lub wycofaniu zmian jest podejmowana, w najprostszym przypadku, na końcu zadania. Może być jednak bardziej przydatne dla aplikacji w celu zsynchronizowania zmian danych w innych punktach logicznych w ramach zadania. Te punkty logiczne są nazywane punktami synchronizacji (lub punktami synchronizacji), a okres przetwarzania zestawu aktualizacji między dwoma punktami synchronizacji jest nazywany *jednostką pracy*. Kilka wywołań MQGET i wywołań MQPUT może być częścią pojedynczej jednostki pracy.

W produkcie WebSphere MQkonieczne jest rozróżnienie jednostek pracy *local* i *global* :

Lokalne jednostki pracy
Są to te, w których jedynymi działaniami są operacje umieszczania i pobierania z kolejek produktu WebSphere MQ , a koordynacja każdej jednostki pracy jest udostępniana w menedżerze kolejek przy użyciu procesu *zatwierdzania jednofazowego* .

Lokalnych jednostek pracy należy używać, gdy jedynymi zasobami, które mają zostać zaktualizowane, są kolejki zarządzane przez pojedynczy menedżer kolejek produktu WebSphere MQ . Aktualizacje są zatwierdzane za pomocą komendy MQCMIT lub wycofanych przy użyciu komendy MQBACK.

Nie ma żadnych zadań administrowania systemem, innych niż zarządzanie dziennikiem, które są zaangażowane w korzystanie z lokalnych jednostek pracy. W aplikacjach, w których używane są wywołania MQPUT i MQGET z opcją MQCMIT i MQBACK, należy spróbować użyć opcji MQPMO_SYNCPOINT i MQGMO_SYNCPOINT. Informacje na temat zarządzania dziennikiem znajdują się w sekcji Zarządzanie plikami dzienników .

Globalne jednostki pracy
Czy te, w których inne zasoby, takie jak tabele w relacyjnej bazie danych, są również aktualizowane. Jeśli w grę uczestniczy więcej niż jeden *menedżer zasobów* , potrzebne jest oprogramowanie *menedżer transakcji* , które korzysta z procesu *zatwierdzania dwufazowego* w celu koordynowania globalnej jednostki pracy.

Globalnych jednostek pracy należy używać wtedy, gdy konieczne jest również uwzględnienie aktualizacji oprogramowania menedżera relacyjnych baz danych, takich jak Db2, Oracle, Sybasei Informix.

Istnieje kilka możliwych scenariuszy korzystania z globalnych jednostek pracy. Udokumentowane tutaj są dwa scenariusze:

1. W pierwszym przypadku menedżer kolejek sam pełni rolę menedżera transakcji. W tym scenariuszu czasowniki MQI sterują globalnymi jednostkami pracy. Są one uruchamiane w aplikacjach przy użyciu komendy MQBEGIN, a następnie zatwierdzane za pomocą komendy MQCMIT lub wycofanych przy użyciu komendy MQBACK.
2. W drugim przypadku rola menedżera transakcji jest wykonywana przez inne oprogramowanie, takie jak TXSeries, Encina, lub Tuxedo. W tym scenariuszu do sterowania jednostką pracy używany jest interfejs API udostępniany przez oprogramowanie menedżera transakcji (na przykład EXEC CICS SYNCPOINT dla TXSeries).

W poniższych sekcjach opisano wszystkie kroki niezbędne do korzystania z globalnych jednostek pracy, zorganizowane według dwóch scenariuszy:

- “Scenariusz 1: Menedżer kolejek wykonuje koordynację” na stronie 43
- “Scenariusz 2: Inne oprogramowanie zapewnia koordynację” na stronie 70

Scenariusz 1: Menedżer kolejek wykonuje koordynację

W scenariuszu 1 menedżer kolejek działa jako menedżer transakcji. W tym scenariuszu czasowniki MQI sterują globalnymi jednostkami pracy. Są one uruchamiane w aplikacjach przy użyciu komendy MQBEGIN, a następnie zatwierdzane za pomocą komendy MQCMIT lub wycofanych przy użyciu komendy MQBACK.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Poziom odseparowania

W produkcie IBM WebSphere MQkomunikat w kolejce może być widoczny przed aktualizacją bazy danych, w zależności od projektu odseparowania transakcji zaimplementowanego w bazie danych.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Gdy menedżer kolejek produktu IBM WebSphere MQ pracuje jako menedżer transakcji XA w celu koordynowania aktualizacji do menedżerów zasobów XA, następuje następujący protokół zatwierdzania:

1. Przygotuj wszystkie menedżery zasobów XA.
2. Zatwierdź menedżera zasobów menedżera kolejek produktu IBM WebSphere MQ .
3. Zatwierdź inne menedżery zasobów.

Między krokiem 2 i 3 aplikacja może wyświetlić komunikat, który jest zatwierdzany w kolejce, ale odpowiadający mu wiersz w bazie danych nie odzwierciedla tego komunikatu.

To nie jest problem, jeśli baza danych jest skonfigurowana w taki sposób, że wywołania API bazy danych aplikacji oczekują na zakończenie oczekujących aktualizacji.

Tę opcję można rozwiązać, konfigurując bazę danych w inny sposób. Wymagany typ konfiguracji jest określany jako "poziom odseparowania". Więcej informacji na temat poziomów odseparowania można znaleźć w dokumentacji bazy danych. Można, alternatywnie, skonfigurować menedżera kolejek w celu zatwierdzenia menedżerów zasobów w następującej kolejności odwrotnej:

1. Przygotuj wszystkie menedżery zasobów XA.
2. Zatwierdź inne menedżery zasobów.
3. Zatwierdź menedżera zasobów menedżera kolejek produktu IBM WebSphere MQ .

W przypadku zmiany protokołu menedżer kolejek produktu IBM WebSphere MQ jest zatwierdzany jako ostatni, dlatego aplikacje odczytane z kolejek będą widzieć komunikat dopiero po zakończeniu odpowiedniej aktualizacji bazy danych.

Aby skonfigurować menedżer kolejek w taki sposób, aby używał tego zmienionego protokołu, należy ustawić zmienną środowiskową **AMQ_REVERSE_COMMIT_ORDER** .

Ustaw tę zmienną środowiskową w środowisku, z którego uruchamiany jest produkt **strmqm** w celu uruchomienia menedżera kolejek. Na przykład przed uruchomieniem menedżera kolejek należy wykonać następujące czynności w powłoce:

```
export AMQ_REVERSE_COMMIT_ORDER=1
```

Uwaga: Ustawienie tej zmiennej środowiskowej może spowodować dodatkową pozycję dziennika na transakcję, więc będzie to miało niewielki wpływ na wydajność każdej transakcji.

Koordynacja bazy danych

Gdy menedżer kolejek koordynuje globalne jednostki pracy, staje się możliwe zintegrowanie aktualizacji bazy danych w ramach jednostek pracy. Oznacza to, że można zapisywać mieszane aplikacje MQI i SQL, a także można użyć komend MQCMIT i MQBACK w celu zatwierdzenia lub wycofania zmian w kolejkach i bazach danych.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Menedżer kolejek osiąga to za pomocą protokołu zatwierdzania dwufazowego opisanego w sekcji *X/Open Distributed Transaction Processing: The XA Specification*(Specyfikacja rozproszonej transakcji X/Open:

Specyfikacja XA). Gdy jednostka pracy ma zostać zatwierdzona, menedżer kolejek najpierw zwraca się do każdego uczestniczącego menedżera bazy danych, czy jest on przygotowany do zatwierdzenia jego aktualizacji. Tylko wtedy, gdy wszystkie uczestnicy, w tym sam menedżer kolejek, są przygotowani do zatwierdzenia, są zatwierdzane wszystkie aktualizacje kolejki i bazy danych. Jeśli żaden z uczestników nie może przygotować swoich aktualizacji, tworzona jest kopia zapasowa jednostki pracy.

Generalnie globalna jednostka pracy jest implementowana w aplikacji za pomocą następującej metody (w pseudocode):

```
MQBEGIN
MQGET (zawiera flagę MQGMO_SYNCPOINT w opcjach komunikatu)
MQPUT (zawiera flagę MQPMO_SYNCPOINT w opcjach komunikatu)
SQL INSERT
MQCMIT
```

Celem komendy MQBEGIN jest oznaczanie początku globalnej jednostki pracy. Celem wywołania MQCMIT jest oznaczanie końca globalnej jednostki pracy i zakończenie jej wraz z wszystkimi uczestniczącymi menedżerami zasobów, przy użyciu protokołu zatwierdzania dwufazowego.

Jeśli jednostka pracy (zwana również *transakcją*) została pomyślnie zakończona za pomocą komendy MQCMIT, wszystkie działania podejmowane w ramach tej jednostki pracy są trwałe lub nieodwracalne. Jeśli z jakiegokolwiek powodu jednostka pracy nie powiedzie się, wszystkie działania zostaną wycofane. Nie jest możliwe, aby jedno działanie w jednostce pracy mogło zostać wykonane na stałe, podczas gdy inna jest wycofana. Jest to zasada jednostki pracy: albo wszystkie działania w ramach jednostki pracy są stałe, albo żadne z nich nie jest.

Uwaga:

1. Programista aplikacji może wymusić utworzenie kopii zapasowej jednostki pracy, wywołując komendę MQBACK. Kopia zapasowa jednostki pracy jest również wycofana przez menedżer kolejek, jeśli aplikacja lub baza danych *nie powiedzie się* przed wywołaniem komendy MQCMIT.
2. Jeśli aplikacja wywołuje komendę MQDISC bez wywoływania komendy MQCMIT, menedżer kolejek zachowuje się tak, jakby wywołano komendę MQCMIT, a następnie zatwierdza jednostkę pracy.

W przypadku między opcją MQBEGIN i MQCMIT menedżer kolejek nie wywołuje żadnych wywołań bazy danych w celu zaktualizowania jej zasobów. Oznacza to, że jedynym sposobem zmiany tabel bazy danych jest kod (na przykład: SQL INSERT w pseudocode).

Pełna obsługa odtwarzania jest udostępniana, jeśli menedżer kolejek utraci kontakt z dowolnym z menedżerów baz danych podczas zatwierdzania protokołu. Jeśli menedżer bazy danych stanie się niedostępny w czasie, gdy jest wątpliwy, oznacza to, że pomyślnie przygotował zatwierdzenie, ale nie otrzymał decyzji o zatwierdzeniu lub wycofaniu, menedżer kolejek pamięta wynik jednostki pracy, dopóki wynik ten nie zostanie pomyślnie dostarczony do bazy danych. Podobnie, jeśli menedżer kolejek kończy się niekompletnymi operacjami zatwierdzania, są one zapamiętywany przy restarcie menedżera kolejek. Jeśli aplikacja nieoczekiwanie zakończy działanie, integralność jednostki pracy nie zostanie naruszona, ale wynik zależy od miejsca, w którym proces aplikacji został zakończony, zgodnie z opisem w sekcji Tabela 5 na stronie 46.

Co się dzieje, gdy baza danych lub program użytkowy nie powiedzie się, należy w następujących tabelach:

Tabela 4. Co się dzieje, gdy serwer bazy danych ulegnie awarii

Wystąpienie usterki	**Wynik**
Przed wywołaniem aplikacji MQCMIT.	Kopia zapasowa jednostki pracy jest wycofana.
Podczas wywoływania aplikacji do komendy MQCMIT, **przed** wszystkie bazy danych wskazują, że zostały one pomyślnie przygotowane.	Kopia zapasowa jednostki pracy jest tworzona z kodem przyczyny MQRC_BACKED_OUT.

Tabela 4. Co się dzieje, gdy serwer bazy danych ulegnie awarii (kontynuacja)	
Wystąpienie usterki	**Wynik**
Podczas wywoływania aplikacji do programu MQCMIT **po** wszystkie bazy danych wskazują, że zostały pomyślnie przygotowane, ale zanim wszystkie wskazywały na to, że zostały pomyślnie zatwierdzone.	Jednostka pracy jest wstrzymana przez menedżera kolejek w stanie odtwarzalnym, z kodem przyczyny MQRC_OUTCOME_PENDING.
Podczas wywoływania aplikacji do komendy MQCMIT **po** wszystkie bazy danych wskazują, że zostały one pomyślnie zatwierdzone.	Jednostka pracy jest zatwierdzana z kodem przyczyny MQRC_NONE.
Po wywołaniu aplikacji na MQCMIT.	Jednostka pracy jest zatwierdzana z kodem przyczyny MQRC_NONE.

Tabela 5. Co się dzieje, gdy działanie programu nie powiedzie się	
Wystąpienie usterki	**Wynik**
Przed wywołaniem aplikacji MQCMIT.	Kopia zapasowa jednostki pracy jest wycofana.
Podczas wywoływania aplikacji do produktu MQCMIT **przed** menedżerem kolejek odebrał żądanie MQCMIT aplikacji.	Kopia zapasowa jednostki pracy jest wycofana.
Podczas wywoływania aplikacji do produktu MQCMIT **po** żądanie menedżera kolejek odebrało żądanie MQCMIT aplikacji.	Menedżer kolejek próbuje zatwierdzić użycie zatwierdzania dwufazowego (z zastrzeżeniem, że produkty bazodanowe pomyślnie wykonają i zatwierdzają swoje części jednostki pracy).

W przypadku, gdy kod przyczyny zwrotu z komendy MQCMIT ma wartość MQRC_OUTCOME_PENDING, jednostka pracy jest zapamiętana przez menedżer kolejek do czasu, aż będzie mogła ponownie nawiązać kontakt z serwerem bazy danych i przekazać jej część jednostki pracy. Więcej informacji na temat sposobu i czasu odtwarzania zawiera sekcja “Uwagi dotyczące utraty kontaktu z menedżerem zasobów XA” na stronie 63 .

Menedżer kolejek komunikuje się z menedżerami bazy danych za pomocą interfejsu XA zgodnie z opisem w sekcji *Przetwarzanie rozproszonego transakcji X/Open: Specyfikacja XA*. Przykładami tych wywołań funkcji są xa_open, xa_start, xa_end, xa_prepare i xa_commit. Używamy terminów *menedżer transakcji* i *menedżer zasobów* w tym samym znaczeniu, w jakim są używane w specyfikacji XA.

Ograniczenia
Istnieją ograniczenia dotyczące obsługi koordynacji bazy danych.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Zastosowanie mają następujące ograniczenia:

- Możliwość koordynowania aktualizacji bazy danych w jednostkach pracy produktu WebSphere MQ **nie** jest obsługiwana w aplikacji klienckiej MQI. Użycie komendy MQBEGIN w aplikacji klienckiej nie powiodło się. Program, który wywołuje komendę MQBEGIN, musi działać jako aplikacja *serwer* na tym samym komputerze, co menedżer kolejek.

  Uwaga: Aplikacja *serwer* to program, który został powiązany z niezbędnymi bibliotekami serwera WebSphere MQ ; aplikacja *klient* to program, który został połączony z niezbędnymi bibliotekami klienta WebSphere MQ . Szczegółowe informacje na temat kompilowania i łączenia programów znajdują się w sekcji “Budowanie aplikacji dla klientów MQI produktu WebSphere MQ” na stronie 366 i “Budowanie aplikacji IBM WebSphere MQ” na stronie 439 .

- Serwer bazy danych może znajdować się na innym komputerze niż serwer menedżera kolejek, o ile klient bazy danych jest zainstalowany na tym samym komputerze, co menedżer kolejek, i obsługuje tę funkcję. Zapoznaj się z dokumentacją produktu bazodanowego, aby określić, czy ich oprogramowanie klienckie może być używane w przypadku dwufazowych systemów zatwierdzania.
- Mimo że menedżer kolejek zachowuje się jak menedżer zasobów (do celów związanych ze scenariuszem 2 globalnych jednostek pracy), nie jest możliwe, aby jeden menedżer kolejek koordynował inny menedżer kolejek w obrębie jego scenariusza 1 globalnych jednostek pracy.

Załaduj pliki ładowania

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Plik ładowania przełącznika jest biblioteką współużytkowaną (DLL w systemach Windows), która jest ładowana przez kod w aplikacji IBM WebSphere MQ i menedżerze kolejek. Jego celem jest uproszczenie ładowania biblioteki współużytkowanej klienta bazy danych, a także zwrócenie wskaźników do funkcji XA.

Szczegóły pliku ładowania przełącznika muszą zostać określone przed uruchomieniem menedżera kolejek. Szczegółowe informacje znajdują się w pliku qm.ini w systemach Windowsi UNIX and Linux .

- W systemach Windows i Linux (platformyx86 i x86-64) należy użyć IBM WebSphere MQ Explorer w celu zaktualizowania pliku qm.ini .
- We wszystkich innych systemach należy edytować plik qm.inibezpośrednio.

Źródło C dla pliku ładowania przełącznika jest dostarczane wraz z instalacją produktu IBM WebSphere MQ , jeśli obsługuje scenariusz 1 globalnych jednostek pracy. Źródło zawiera funkcję o nazwie MQStart. Gdy ładowany jest plik ładowania przełącznika, menedżer kolejek wywołuje tę funkcję, która zwraca adres struktury o nazwie *Przełącznik XA*.

Struktura przełącznika XA istnieje w bibliotece współużytkowanej klienta bazy danych i zawiera pewną liczbę wskaźników funkcji, zgodnie z opisem w sekcji Tabela 6 na stronie 47:

Tabela 6. Wskaźniki funkcji przełącznika XA

Nazwa wskaźnika funkcji	**XA, funkcja**	**Przeznaczenie**
xa_open_entry	xa_open	Łączenie z bazą danych
xa_close_entry	xa_close	Rozłącz z bazą danych
Pozycja xa_start_entry	xa_start	Uruchomienie gałęzi globalnej jednostki pracy
Wpis xa_end_entry	xa_end	Zawieś gałąź globalnej jednostki pracy
Wpis xa_rollback_entry	xa_rollback	Wycofaj gałąź globalnej jednostki pracy
Pozycja xa_prepare_entry	xa_prepare	Przygotuj się do zatwierdzenia gałęzi globalnej jednostki pracy
xa_commit_entry	xa_commit	Zatwierdzanie gałęzi globalnej jednostki pracy
Pozycja xa_recover_entry	xa_recover	Wykryj z bazy danych, czy ma ona wątpliwe jednostki pracy
Pozycja xa_forget_entry	xa_forget	Zezwalaj bazie danych na zapominanie o gałęzi globalnej jednostki pracy
xa_complete_entry	xa_complete	Wypełnij gałąź globalnej jednostki pracy

Podczas pierwszego wywołania komendy MQBEGIN w aplikacji kod IBM WebSphere MQ , który jest wykonywany jako część komendy MQBEGIN, ładuje plik ładowania przełącznika i wywołuje funkcję xa_open w bibliotece współużytkowanej bazy danych. Podobnie podczas uruchamiania menedżera kolejek i przy innych kolejnych okazjach, niektóre procesy menedżera kolejek ładują plik ładowania przełącznika i wywołują xa_open.

Liczbę wywołań xa_ * można zmniejszyć za pomocą *rejestracji dynamicznej*. Pełny opis tej techniki optymalizacji znajduje się w sekcji “Rejestracja dynamiczna XA” na stronie 68.

Konfigurowanie systemu na potrzeby koordynacji bazy danych

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Istnieje kilka zadań, które należy wykonać, zanim menedżer bazy danych będzie mógł uczestniczyć w globalnych jednostkach pracy koordynowanych przez menedżer kolejek. Są one opisane w następujący sposób:

- “Instalowanie i konfigurowanie produktu bazodanowego” na stronie 48
- “Tworzenie plików ładowania przełącznika” na stronie 49
- “Dodawanie informacji konfiguracyjnych do menedżera kolejek” na stronie 49
- “Pisanie i modyfikowanie aplikacji” na stronie 51
- “Testowanie systemu” na stronie 52

Instalowanie i konfigurowanie produktu bazodanowego
Aby zainstalować i skonfigurować produkt bazodanowy, należy zapoznać się z dokumentacją tego produktu. W tej sekcji opisano ogólne problemy związane z konfiguracją oraz sposób ich powiązania między produktem WebSphere MQ a bazą danych.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Połączenia bazy danych

Aplikacja, która nawiązuje standardowe połączenie z menedżerem kolejek, jest powiązana z wątkiem w oddzielnym procesie agenta lokalnego menedżera kolejek. (Połączenie, które nie jest połączeniem *krótkiej ścieżki* , jest połączeniem *standardowym* w tym kontekście. Więcej informacji na ten temat zawiera sekcja “Nawiąże połączenie z menedżerem kolejek przy użyciu wywołania MQCONNX” na stronie 214.)

Gdy aplikacja wydaje `MQBEGIN`, zarówno proces ten, jak i proces agenta wywołują funkcję xa_open w bibliotece klienta bazy danych. W odpowiedzi na to kod biblioteki klienta bazy danych *łączy się* z bazą danych, która ma być włączana do jednostki pracy *zarówno z aplikacji, jak i z procesów menedżera kolejek*. Te połączenia z bazą danych są obsługiwane tak długo, jak długo aplikacja pozostaje połączona z menedżerem kolejek.

Jest to ważne, jeśli baza danych obsługuje tylko ograniczoną liczbę użytkowników lub połączeń, ponieważ do bazy danych są nawiązane dwa połączenia w celu obsługi jednego programu aplikacji.

Konfiguracja klient/serwer

Biblioteka klienta bazy danych załadowana do menedżera kolejek produktu WebSphere MQ i procesy aplikacji **musi** być w stanie wysyłać i odbierać z serwera. Upewnij się, że:

- Szczegółowe informacje znajdują się w plikach konfiguracyjnych klient/serwer bazy danych.
- Odpowiednie zmienne środowiskowe są ustawiane w środowisku menedżera kolejek **i** procesy aplikacji.

Tworzenie plików ładowania przełącznika
Produkt WebSphere MQ jest dostarczany z przykładowym plikiem makefile, używanym do budowania plików ładowania przełącznika dla obsługiwanych menedżerów baz danych.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

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

Przykładowy plik makefile, wraz ze wszystkimi powiązanymi plikami źródłowymi C, które są wymagane do zbudowania plików ładowania przełącznika, jest instalowany w następujących katalogach:

- W przypadku produktu WebSphere MQ for Windows, w katalogu *MQ_INSTALLATION_PATH*\tools\c\samples\xatm\
- W przypadku systemów WebSphere MQ dla systemów UNIX and Linux w katalogu *MQ_INSTALLATION_PATH*/samp/xatm/

Przykładowe moduły źródłowe używane do budowania plików ładowania przełączników są następujące:

- W przypadku bazy danych DB2, db2swit.c
- W przypadku bazy danych Oracle: oraswit.c
- W przypadku produktu Informix: infswit.c
- W przypadku bazy danych Sybase: sybswit.c

Podczas generowania plików ładowania przełącznika należy zainstalować 32-bitowe pliki ładowania przełączników w produkcie /var/mqm/exits i zainstalować 64-bitowe pliki ładowania przełączników w produkcie /var/mqm/exits64.

Jeśli istnieją 32-bitowe menedżery kolejek, przykładowy plik make, xaswit.mak, instaluje 32-bitowy plik ładowania przełącznika w produkcie /var/mqm/exits.

Jeśli istnieją 64-bitowe menedżery kolejek, przykładowy plik make, xaswit.mak, instaluje 32-bitowy plik ładowania przełącznika w programie /var/mqm/exitsi 64-bitowy plik ładowania przełącznika w produkcie /var/mqm/exits64.

Zabezpieczenia pliku

Możliwe jest, że system operacyjny może zakończyć ładowanie pliku ładowania przełącznika za pomocą programu WebSphere MQz powodów poza kontrolą produktu WebSphere MQ. W takim przypadku komunikaty o błędach są zapisywane w dziennikach błędów produktu WebSphere MQ , a potencjalnie wywołanie komendy MQBEGIN może się nie powieść. Aby upewnić się, że system operacyjny nie zawiedzie ładowania pliku ładowania przełącznika, należy spełnić następujące wymagania:

1. Plik ładowania przełącznika musi być dostępny w położeniu określonym w pliku qm.ini .
2. Plik ładowania przełącznika musi być dostępny dla wszystkich procesów, które muszą zostać załadowane, w tym procesy menedżera kolejek i procesy aplikacji.
3. Wszystkie biblioteki, od których zależy plik ładowania przełącznika, w tym biblioteki udostępniane przez produkt bazodanowy, muszą być obecne i dostępne.

Dodawanie informacji konfiguracyjnych do menedżera kolejek
Po utworzeniu pliku ładowania przełącznika dla menedżera bazy danych i umieszczanie go w bezpiecznym miejscu należy określić to położenie w menedżerze kolejek.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Aby określić położenie, wykonaj następujące kroki:

- On Okna and Linux (x86 and x86-64 platforms) systems use the WebSphere MQ Explorer. Określ szczegóły pliku ładowania przełącznika na panelu właściwości menedżera kolejek pod kontrolą menedżera zasobów XA.
- We wszystkich pozostałych systemach należy określić szczegóły pliku ładowania przełącznika w sekcji XAResourceManager w pliku qm.ini menedżera kolejek.

Dodaj sekcję XAResourceManager dla bazy danych, która ma być koordynowana przez menedżer kolejek. Najczęstszym przypadkiem jest to, że istnieje tylko jedna baza danych, a więc tylko jedna sekcja XAResourceManager . Szczegółowe informacje na temat bardziej skomplikowanych konfiguracji obejmujących wiele baz danych można znaleźć w sekcji “Wiele konfiguracji bazy danych” na stronie 62. Atrybuty sekcji XAResourceManager są następujące:

Nazwa=nazwa
Łańcuch wybrany przez użytkownika, który identyfikuje menedżera zasobów. W efekcie nadaje on nazwę sekcji XAResourceManager . Nazwa jest obowiązkowa i może mieć długość do 31 znaków.

Wybrana nazwa musi być unikalna; w tym pliku qm.ini musi istnieć tylko jedna sekcja XAResourceManager o tej nazwie. Nazwa powinna również mieć znaczenie, ponieważ menedżer kolejek używa jej do odwołania się do tego menedżera zasobów zarówno w komunikatach dziennika błędów menedżera kolejek, jak i w danych wyjściowych, gdy używana jest komenda `dspmqtrn` . Więcej informacji na ten temat zawiera sekcja “Wyświetlanie zaległych jednostek pracy za pomocą komendy dspmqtrn” na stronie 64 .

Po wybraniu nazwy i uruchomieniu menedżera kolejek, nie należy zmieniać atrybutu Nazwa. Więcej szczegółowych informacji na temat zmiany informacji o konfiguracji zawiera sekcja “Zmiana informacji konfiguracyjnych” na stronie 67.

SwitchFile= nazwa
Jest to nazwa pliku ładowania przełącznika XA, który został utworzony wcześniej. Jest to atrybut obowiązkowy. Kod w menedżerze kolejek i w procesach aplikacji WebSphere MQ próbuje dwukrotnie załadować plik ładowania przełącznika:

1. Przy uruchamianiu menedżera kolejek
2. Po pierwszym wywołaniu komendy MQBEGIN w procesie aplikacji produktu WebSphere MQ

Atrybuty zabezpieczeń i uprawnień w pliku ładowania przełącznika muszą zezwalać na wykonanie tego działania przez te procesy.

XAOpenString= łańcuch
Jest to łańcuch danych, który kod WebSphere MQ przekazuje w swoich wywołaniach do funkcji xa_open menedżera bazy danych. Jest to atrybut opcjonalny. Jeśli zostanie pominięty łańcuch o zerowej długości, zostanie przyjęty.

Kod w menedżerze kolejek i procesy aplikacji WebSphere MQ wywołują funkcję xa_open przy dwóch okazjach:

1. Przy uruchamianiu menedżera kolejek
2. Po pierwszym wywołaniu komendy MQBEGIN w procesie aplikacji produktu WebSphere MQ

Format tego łańcucha jest określony w szczególności dla każdego produktu bazodanowego, który zostanie opisany w dokumentacji tego produktu. W ogólnym przypadku łańcuch xa_open zawiera informacje o uwierzytelnianiu (nazwa użytkownika i hasło), aby umożliwić połączenie z bazą danych zarówno w menedżerze kolejek, jak i w procesach aplikacji.

XACloseString= łańcuch
Jest to łańcuch danych, który kod WebSphere MQ przekazuje w swoich wywołaniach do funkcji xa_close menedżera bazy danych. Jest to atrybut opcjonalny. Jeśli zostanie pominięty łańcuch o zerowej długości, zostanie przyjęty.

Kod w menedżerze kolejek i procesy aplikacji WebSphere MQ wywołują funkcję xa_close przy dwóch okazjach:

1. Przy uruchamianiu menedżera kolejek

2. Po wywołaniu funkcji MQDISC w procesie aplikacji produktu WebSphere MQ , po wcześniejszym wywołaniu komendy MQBEGIN

Format tego łańcucha jest określony w szczególności dla każdego produktu bazodanowego, który zostanie opisany w dokumentacji tego produktu. W ogólnym przypadku łańcuch jest pusty i często pomija atrybut XACloseString z sekcji XAResourceManager .

ThreadOfControl=THREAD |PROCESS

Wartością elementu sterującego ThreadOfmoże być THREAD lub PROCESS. Menedżer kolejek używa go do serializacji. Jest to atrybut opcjonalny. Jeśli zostanie pominięty, przyjmowany jest wartość PROCESS.

Jeśli kod klienta bazy danych zezwala wątkom na wywołowanie funkcji XA bez serializacji, wartość elementu sterującego ThreadOfmoże być wartością THREAD. Menedżer kolejek zakłada, że może wywoływać funkcje XA w bibliotece współużytkowanej klienta bazy danych z wielu wątków w tym samym czasie, jeśli to konieczne.

Jeśli kod klienta bazy danych nie zezwala na wywożanie przez wątki jego funkcji XA w ten sposób, wartość elementu sterującego ThreadOfmusi być ustawiona na PROCESS. W tym przypadku menedżer kolejek serializuje wszystkie wywołania do biblioteki współużytkowanej klienta bazy danych w taki sposób, aby tylko jedno wywołanie w danym momencie było wykonywane z poziomu określonego procesu. Prawdopodobnie konieczne jest również zapewnienie, że aplikacja wykonuje podobną serializację, jeśli jest ona uruchamiana z wieloma wątkami.

Należy zwrócić uwagę, że ta kwestia, w której produkt bazodanowy ma zdolność radzenia sobie w procesach wielowątkowych w ten sposób, jest problemem dla dostawcy tego produktu. Szczegółowe informacje na temat tego, czy można ustawić atrybut elementu sterującego ThreadOfna THREAD lub PROCESS, można znaleźć w dokumentacji produktu bazodanowego. Zaleca się, aby, jeśli można, ustawić parametr ThreadOfControl na THREAD. W razie wątpliwości opcja *bezpieczniej* ma ustawić ją na wartość PROCESS, chociaż utracisz potencjalne korzyści wynikające z używania THREAD.

Pisanie i modyfikowanie aplikacji
W jaki sposób zaimplementować globalną jednostkę pracy.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Przykładowe aplikacje dla globalnych jednostek pracy Scenariusza 1, które są dostarczane z instalacją produktu WebSphere MQ , są opisane w publikacji “Wprowadzenie jednostek pracy” na stronie 42.

Generalnie globalna jednostka pracy jest implementowana w aplikacji za pomocą następującej metody (w pseudocode):

```
MQBEGIN
MQGET
MQPUT
SQL INSERT
MQCMIT
```

Celem komendy MQBEGIN jest oznaczanie początku globalnej jednostki pracy. Celem wywołania MQCMIT jest oznaczanie końca globalnej jednostki pracy i zakończenie jej wraz z wszystkimi uczestniczącymi menedżerami zasobów, przy użyciu protokołu zatwierdzania dwufazowego.

W przypadku między opcją MQBEGIN i MQCMIT menedżer kolejek nie wywołuje żadnych wywołań bazy danych w celu zaktualizowania jej zasobów. Oznacza to, że jedynym sposobem zmiany tabel bazy danych jest kod (na przykład: SQL INSERT w pseudocode).

Rolą menedżera kolejek, jeśli chodzi o bazę danych, jest określenie, kiedy globalna jednostka pracy została uruchomiona, kiedy została zakończona, oraz czy globalna jednostka pracy powinna być zatwierdzona, czy wycofana.

Jeśli chodzi o aplikację, menedżer kolejek wykonuje dwie role: menedżera zasobów (w którym zasoby są komunikatami w kolejkach) oraz menedżera transakcji dla globalnej jednostki pracy.

Należy rozpocząć od dostarczonych programów przykładowych i pracować z użyciem różnych wywołań funkcji API WebSphere MQ i baz danych, które są wykonywane w tych programach. Te wywołania interfejsu API są w pełni udokumentowane w “Przykładowe programy produktu WebSphere MQ” na stronie 98, Typy danych używane w MQIoraz (w przypadku interfejsu własnego interfejsu API bazy danych) własną dokumentację bazy danych.

Testowanie systemu
Użytkownik wie, czy aplikacja i system są poprawnie skonfigurowane tylko przez uruchomienie ich podczas testowania. Istnieje możliwość przetestowania konfiguracji systemu (pomyślnej komunikacji między menedżerem kolejek i bazą danych) przez budowanie i uruchamianie jednego z dostarczonych programów przykładowych.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Konfigurowanie Db2
Informacje dotyczące obsługi i konfiguracji produktu DB2 .

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Obsługiwane poziomy interfejsu Db2 są definiowane na stronie IBM WebSphere MQ detailed system requirements .

Uwaga: 32-bitowe instancje serwera Db2 nie są obsługiwane na platformach, na których menedżer kolejek jest 64-bitowy.

Wykonaj następujące czynności:

1. Sprawdź ustawienia zmiennych środowiskowych.
2. Utwórz plik ładowania przełącznika Db2 .
3. Dodaj informacje o konfiguracji menedżera zasobów.
4. Jeśli to konieczne, zmień parametry konfiguracyjne Db2 .

Należy zapoznać się z ogólnymi informacjami dostarczonym w produkcie “Konfigurowanie systemu na potrzeby koordynacji bazy danych” na stronie 48.

Ostrzeżenie: Jeśli produkt db2`profile` jest uruchamiany na platformach UNIX and Linux , ustawiane są zmienne środowiskowe LIBPATH i LD_LIBRARY_PATH. Zaleca się, aby `unset` te zmienne środowiskowe, patrz odpowiednie *Krótkie wprowadzenie* .

Sprawdzanie ustawień zmiennej środowiskowej Db2

Upewnij się, że zmienne środowiskowe Db2 są ustawione dla procesów menedżera kolejek ***, jak również*** procesów aplikacji. W szczególności należy zawsze ustawić zmienną środowiskową DB2INSTANCE ***przed*** , aby uruchomić menedżer kolejek. Zmienna środowiskowa DB2INSTANCE identyfikuje instancję bazy danych Db2 zawierającą zaktualizowane bazy danych Db2 . Na przykład:

- W systemach UNIX and Linux należy użyć:

```
export DB2INSTANCE=db2inst1
```

- W systemach Windows należy użyć:

```
set DB2INSTANCE=DB2
```

W systemie Windows z bazą danych Db2 należy dodać użytkownika MUSR_MQADMIN do grupy DB2USERS , aby umożliwić uruchomienie menedżera kolejek.

Tworzenie pliku ładowania przełącznika Db2

Najprostszym sposobem utworzenia pliku ładowania przełącznika Db2 jest użycie przykładowego pliku xaswit.mak, który produkt WebSphere MQ udostępnia w celu zbudowania plików ładowania przełącznika dla różnych produktów bazodanowych.

W systemach Windows plik xaswit.mak znajduje się w katalogu *MQ_INSTALLATION_PATH*\tools\c\samples\xatm. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ . Aby utworzyć plik ładowania przełącznika Db2 za pomocą programu Microsoft Visual C + +, należy użyć:

```
nmake /f xaswit.mak db2swit.dll
```

Wygenerowany plik przełącznika jest umieszczany w produkcie c:\Program Files\IBM\WebSphere MQ\exits.

Plik xaswit.mak znajduje się w katalogu *MQ_INSTALLATION_PATH*/samp/xatm. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .

Zmodyfikuj plik xaswit.mak na *usuń komentarz* z linii odpowiednich do używanej wersji Db2 . Następnie wykonaj komendę makefile, używając komendy:

```
make -f xaswit.mak db2swit
```

Wygenerowany 32-bitowy plik ładowania przełącznika jest umieszczany w produkcie /var/mqm/exits.

Wygenerowany 64-bitowy plik ładowania przełącznika jest umieszczany w produkcie /var/mqm/exits64.

Dodawanie informacji konfiguracyjnych menedżera zasobów dla Db2

Należy zmodyfikować informacje konfiguracyjne dla menedżera kolejek, aby zadeklarować wartość Db2 jako uczestnika w globalnych jednostkach pracy. Modyfikowanie informacji konfiguracyjnych w ten sposób jest opisane w bardziej szczegółowych informacjach w programie “Dodawanie informacji konfiguracyjnych do menedżera kolejek” na stronie 49.

- W systemach Windows i Linux (platformyx86 i x86-64) należy użyć programu WebSphere MQ Explorer. Określ szczegóły pliku ładowania przełącznika na panelu właściwości menedżera kolejek pod kontrolą menedżera zasobów XA.
- We wszystkich pozostałych systemach należy określić szczegóły pliku ładowania przełącznika w sekcji XAResourceManager w pliku qm.ini menedżera kolejek.

Rysunek 9 na stronie 53 to przykład systemu UNIX , który przedstawia wpis XAResourceManager , w którym baza danych, która ma być koordynowana, nosi nazwę mydbname, przy czym ta nazwa jest określana w XAOpenString:

```
XAResourceManager:
  Name=mydb2
  SwitchFile=db2swit
  XAOpenString=mydbname,myuser,mypasswd,toc=t
  ThreadOfControl=THREAD
```

Rysunek 9. Przykładowa pozycja XAResourceManager dla bazy danych Db2 na platformach UNIX

Uwaga:

1. Produkt `ThreadOfControl=THREAD` nie może być używany z wersjami Db2 w wersjach wcześniejszych niż wersja 8. Ustaw parametr `ThreadOfControl` i parametr XAOpenString `toc` na jedną z następujących kombinacji:
   - `ThreadOfControl=THREAD` i `toc=t`
   - `ThreadOfControl=PROCESS` i `toc=p`

   Jeśli do włączenia koordynacji JDBC/JTA używany jest plik ładowania przełącznika jdbcdb2 XA, należy użyć produktów `ThreadOfControl=PROCESS` i `toc=p`.

Zmiana parametrów konfiguracyjnych Db2

Dla każdej bazy danych Db2 , którą koordynuje menedżer kolejek, należy ustawić uprawnienia do bazy danych, zmienić parametr tp_mon_name i zresetować parametr maxappls. W tym celu należy wykonać następujące kroki:

Ustaw uprawnienia do bazy danych

Procesy menedżera kolejek są uruchamiane z efektywnym użytkownikiem i grupą mqm w systemach UNIX and Linux . W systemach Windows są one uruchamiane jako użytkownik, który uruchomił menedżer kolejek. Może to być jeden z następujących elementów:

1. Użytkownik, który wydał komendę `strmqm` , lub
2. Użytkownik, na podstawie którego działa serwer IBM MQSeries Service COM.

Domyślnie ten użytkownik ma nazwę MUSR_MQADMIN.

Jeśli w łańcuchu xa_open nie podano nazwy użytkownika i hasła, **użytkownik, w którym jest uruchomiony menedżer kolejek** jest używany przez Db2 do uwierzytelniania wywołania xa_open. Jeśli ten użytkownik (na przykład użytkownik mqm w systemach UNIX and Linux) nie ma minimalnych uprawnień w bazie danych, baza danych odmawia uwierzytelnienia wywołania xa_open.

Te same uwagi mają zastosowanie do procesu aplikacji. Jeśli nie podano nazwy użytkownika i hasła w łańcuchu xa_open, użytkownik, pod którym aplikacja jest uruchomiona, jest używany przez Db2 w celu uwierzytelnienia wywołania xa_open, które jest wykonywane podczas pierwszej komendy MQBEGIN. Ten użytkownik musi mieć minimalne uprawnienia w bazie danych, aby to działało.

Aby na przykład nadać użytkownikowi mqm uprawnienie do połączenia w bazie danych mydbname, wydając następujące komendy Db2 :

```
db2 connect to mydbname
db2 grant connect on database to user mqm
```

Więcej informacji na temat zabezpieczeń zawiera sekcja "Zagadnienia związane z zabezpieczeniami" na stronie 63 .

Windows **Zmień parametr TP_MON_NAME**

W przypadku bazy danych Db2 tylko w systemach Windows zmień parametr konfiguracyjny TP_MON_NAME, aby nazwać bibliotekę DLL, której Db2 używa do wywoływania menedżera kolejek w celu dynamicznej rejestracji.

Użyj komendy `db2 update dbm cfg using TP_MON_NAME mqmax` do nazwy MQMAX.DLL jest biblioteką, której Db2 używa do wywołania menedżera kolejek. Musi być ona obecna w katalogu określonym w zmiennej PATH.

Zresetuj parametr maxappls

Może być konieczne przejrzenie ustawienia parametru *maxappls* , który ogranicza maksymalną liczbę aplikacji, które mogą być połączone z bazą danych. Patrz "Instalowanie i konfigurowanie produktu bazodanowego" na stronie 48.

Konfigurowanie produktu Oracle

Informacje dotyczące obsługi i konfiguracji produktu Oracle .

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Wykonaj następujące czynności:

1. Sprawdź ustawienia zmiennych środowiskowych.
2. Utwórz plik ładowania przełącznika Oracle .
3. Dodaj informacje o konfiguracji menedżera zasobów.
4. Jeśli to konieczne, zmień parametry konfiguracyjne Oracle .

Aktualna lista poziomów Oracle obsługiwanych przez produkt IBM WebSphere MQ jest dostępna na stronie IBM WebSphere MQ detailed system requirements .

Sprawdzanie ustawień zmiennych środowiskowych Oracle

Upewnij się, że zmienne środowiskowe Oracle są ustawione zarówno dla procesów menedżera kolejek, jak i w procesach aplikacji. W szczególności przed uruchomieniem menedżera kolejek zawsze należy ustawić następujące zmienne środowiskowe:

ORACLE_HOME
Katalog główny Oracle . Na przykład w systemach UNIX and Linux należy użyć następujących elementów:

```
export ORACLE_HOME=/opt/oracle/product/8.1.6
```

W systemach Windows należy użyć:

```
set ORACLE_HOME=c:\oracle\ora81
```

ID__SID_ORACLE
Używany identyfikator SID bazy danych Oracle . Jeśli używany jest serwer Net8 dla połączeń klient/serwer, może nie być konieczne ustawienie tej zmiennej środowiskowej. Zapoznaj się z dokumentacją Oracle .

Kolejnym przykładem jest przykład ustawienia tej zmiennej środowiskowej w systemach UNIX and Linux :

```
export ORACLE_SID=sid1
```

Odpowiedniki w systemach Windows są następujące:

```
set ORACLE_SID=sid1
```

Uwaga: Zmienna środowiskowa PATH musi być ustawiona w taki sposób, aby zawierała katalog plików binarnych (na przykład `ORACLE_INSTALL_DIR/VERSION/32BIT_NAME/bin` lub `ORACLE_INSTALL_DIR/VERSION/64BIT_NAME/bin`), w przeciwnym razie może zostać wyświetlony komunikat informujący o tym, że w komputerze brakuje bibliotek Oracle.

Jeśli menedżery kolejek są uruchamiane w 64-bitowym systemie Windows , należy zainstalować zarówno 64-bitowe, jak i 32-bitowe klienty Oracle . Należy zainstalować oba klienty, ponieważ menedżer kolejek jest uruchamiany jako 32-bitowe procesy, które korzystają z 32-bitowego pliku ładowania przełącznika, który z kolei musi uruchomić 32-bitową bibliotekę DLL klienta Oracle .

Plik ładowania przełącznika, załadowany przez 64-bitowe menedżery kolejek, musi mieć dostęp do 64-bitowej biblioteki klienta Oracle . 32-bitowe menedżery kolejek muszą mieć dostęp do 32-bitowego klienta Oracle , gdy produkt IBM WebSphere MQ jest uruchomiony w 64-bitowym systemie Windows .

Tworzenie pliku ładowania przełącznika Oracle

Aby utworzyć plik ładowania przełącznika Oracle , należy użyć przykładowego pliku `xaswit.mak`, który produkt IBM WebSphere MQ udostępnia w celu zbudowania plików ładowania przełącznika dla różnych produktów bazodanowych. W systemach Windows program `xaswit.mak` znajduje się w katalogu `C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm`. Aby utworzyć plik ładowania przełącznika Oracle przy użyciu programu Microsoft Visual C + +, należy użyć: `nmake /f xaswit.mak oraswit.dll`

Wygenerowany plik przełącznika jest umieszczany w produkcie *MQ_INSTALLATION_PATH*`\exits`.*MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowano produkt IBM WebSphere MQ .

Produkt `xaswit.mak` znajduje się w katalogu *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowano produkt IBM WebSphere MQ .

Edytuj `xaswit.mak` , aby usunąć znaki komentarza z wierszy właściwych dla używanej wersji bazy danych Oracle . Następnie wykonaj komendę makefile, używając komendy:

```
make -f xaswit.mak oraswit
```

Wygenerowany 32-bitowy plik ładowania przełącznika jest umieszczany w katalogu `/var/mqm/exits`.

Wygenerowany 64-bitowy plik ładowania przełącznika jest umieszczany w produkcie `/var/mqm/exits64`.

Dodawanie informacji konfiguracyjnych menedżera zasobów dla bazy danych Oracle

Należy zmodyfikować informacje konfiguracyjne dla menedżera kolejek, aby zadeklarować bazę danych Oracle jako uczestnika globalnych jednostek pracy. Modyfikowanie informacji konfiguracyjnych dla menedżera kolejek w ten sposób jest opisane bardziej szczegółowo w sekcji “Dodawanie informacji konfiguracyjnych do menedżera kolejek” na stronie 49.

- W systemach Windows i Linux (platformyx86 i x86-64) należy użyć IBM WebSphere MQ Explorer. Określ szczegóły pliku ładowania przełącznika na panelu właściwości menedżera kolejek pod kontrolą menedżera zasobów XA.
- We wszystkich pozostałych systemach należy określić szczegóły pliku ładowania przełącznika w sekcji XAResourceManager w pliku `qm.ini` menedżera kolejek.

Rysunek 10 na stronie 56 to próbka systemów UNIX and Linux zawierająca wpis XAResourceManager . Należy dodać `LogDir` do otwartego łańcucha XA w taki sposób, aby wszystkie informacje o błędach i śledzeniu były rejestrowane w tym samym miejscu.

```
XAResourceManager:
  Name=myoracle
  SwitchFile=oraswit
  XAOpenString=Oracle_XA+Acc=P/myuser/mypasswd+SesTm=35+LogDir=/tmp+threads=true
  ThreadOfControl=THREAD
```

Rysunek 10. Przykładowy wpis XAResourceManager dla bazy danych Oracle na platformach UNIX and Linux

Uwaga:

1. W programie Rysunek 10 na stronie 56łańcuch xa_open został użyty z czterema parametrami. Dodatkowe parametry można dołączyć zgodnie z opisem w dokumentacji Oracle.
2. Jeśli używany jest parametr IBM WebSphere MQ `ThreadOfControl=THREAD` , należy użyć parametru `+threads=true` Oracle w sekcji XAResourceManager .

Więcej informacji na temat łańcucha xa_open zawiera podręcznik *Oracle8 Server Application Developer's Guide* .

Zmiana parametrów konfiguracyjnych Oracle

W przypadku każdej bazy danych Oracle koordynującej menedżer kolejek należy przejrzeć maksymalną liczbę sesji i ustawić uprawnienia do bazy danych. Aby to zrobić, wykonaj następujące kroki:

Przejrzyj maksymalną liczbę sesji
Może być konieczne przejrzenie ustawień LICENSE_MAX_SESSIONS i PROCESSES w celu uwzględnienia dodatkowych połączeń wymaganych przez procesy należące do menedżera kolejek. Więcej szczegółów na ten temat zawiera sekcja “Instalowanie i konfigurowanie produktu bazodanowego” na stronie 48.

Ustaw uprawnienia do bazy danych
Nazwa użytkownika Oracle określona w łańcuchu xa_open musi mieć uprawnienia do uzyskiwania dostępu do widoku DBA_PENDING_TRANSACTIONS, zgodnie z opisem w dokumentacji Oracle .

Niezbędne uprawnienia można nadać za pomocą następującej komendy:

```
grant select on DBA_PENDING_TRANSACTIONS to myuser;
```

Konfigurowanie produktu Informix

Informacje dotyczące obsługi i konfiguracji produktu Informix .

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Wykonaj następujące czynności:

1. Upewnij się, że zainstalowany jest odpowiedni pakiet SDK klienta Informix :
   - 32-bitowe menedżery kolejek i aplikacje wymagają 32-bitowego pakietu SDK klienta Informix .
   - 64-bitowe menedżery kolejek i aplikacje wymagają 64-bitowego pakietu SDK klienta Informix .
2. Upewnij się, że bazy danych Informix są poprawnie utworzone w produkcie .
3. Sprawdź ustawienia zmiennych środowiskowych.
4. Zbuduj plik ładowania przełącznika Informix .
5. Dodaj informacje o konfiguracji menedżera zasobów.

Aktualna lista poziomów produktu Informix obsługiwana przez produkt WebSphere MQ jest dostępna na stronieIBM WebSphere MQ detailed system requirements .

Sprawdzanie, czy bazy danych Informix są poprawnie utworzone

Każda baza danych Informix , która ma być koordynowana przez menedżer kolejek produktu WebSphere MQ , musi zostać utworzona z podaniem parametru `log` . Na przykład:

```
create database mydbname with log;
```

Menedżery kolejek produktu WebSphere MQ nie mogą koordynować baz danych Informix , które nie mają określonego parametru `log` podczas tworzenia. Jeśli menedżer kolejek próbuje skoordynować bazę danych Informix , która nie ma określonego parametru `log` podczas tworzenia, wywołanie xa_open Informix nie powiedzie się, a zostanie wygenerowana liczba błędów FFST .

Sprawdzanie ustawień zmiennej środowiskowej Informix

Upewnij się, że zmienne środowiskowe Informix są ustawione dla procesów menedżera kolejek ***jak również*** procesów aplikacji. W szczególności należy zawsze ustawić następujące zmienne środowiskowe **przed** uruchomieniem menedżera kolejek:

INFORMIXDIR
Katalog instalacji produktu Informix .

- W przypadku 32-bitowych aplikacji produktu UNIX and Linux należy użyć następującej komendy:

```
export INFORMIXDIR=/opt/informix/32-bit
```

- W przypadku 64-bitowych aplikacji produktu UNIX and Linux należy użyć następującej komendy:

```
export INFORMIXDIR=/opt/informix/64-bit
```

- W przypadku aplikacji Windows należy użyć następującej komendy:

```
set INFORMIXDIR=c:\informix
```

W przypadku systemów z 64-bitowymi menedżerami kolejek, które muszą obsługiwać zarówno aplikacje 32-bitowe, jak i 64-bitowe, wymagane jest zainstalowanie zarówno 32-bitowych, jak i 64-bitowych klientów SDKs Informix . Przykładowy plik makefile `xaswit.mak`, używany do tworzenia pliku ładowania przełącznika, ustawia również katalogi instalacyjne produktu.

INFORMIXSERVER
Nazwa serwera Informix . Na przykład w systemach UNIX and Linux należy użyć następujących elementów:

```
export INFORMIXSERVER=hostname_1
```

W systemach Windows należy użyć:

```
set INFORMIXSERVER=hostname_1
```

ONCONFIG
Nazwa pliku konfiguracyjnego serwera Informix . Na przykład w systemach UNIX and Linux należy użyć następujących elementów:

```
export ONCONFIG=onconfig.hostname_1
```

W systemach Windows należy użyć:

```
set ONCONFIG=onconfig.hostname_1
```

Tworzenie pliku ładowania przełącznika Informix

Aby utworzyć plik ładowania przełącznika Informix , należy użyć przykładowego pliku xaswit.mak, który produkt WebSphere MQ udostępnia w celu zbudowania plików ładowania przełącznika dla różnych produktów bazodanowych. W systemach Windows plik xaswit.mak znajduje się w katalogu *MQ_INSTALLATION_PATH*`\tools\c\samples\xatm`. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ . Aby utworzyć plik ładowania przełącznika Informix za pomocą programu Microsoft Visual C + +, należy użyć następującej komendy:

```
nmake /f xaswit.mak infswit.dll
```

Wygenerowany plik przełącznika jest umieszczany w produkcie `c:\Program Files\IBM\WebSphere MQ\exits`.

Plik xaswit.mak znajduje się w katalogu *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .

Edytuj plik xaswit.mak w celu *anulowania komentarza* wierszy odpowiednich do używanej wersji produktu Informix . Następnie wykonaj komendę makefile, używając komendy:

```
make -f xaswit.mak infswit
```

Wygenerowany 32-bitowy plik ładowania przełącznika jest umieszczany w katalogu `/var/mqm/exits`.

Wygenerowany 64-bitowy plik ładowania przełącznika jest umieszczany w produkcie `/var/mqm/exits64`.

Dodawanie informacji konfiguracyjnych menedżera zasobów dla produktu Informix

Należy zmodyfikować informacje konfiguracyjne dla menedżera kolejek, aby zadeklarować bazę danych Informix jako uczestnika globalnych jednostek pracy. Modyfikowanie informacji konfiguracyjnych dla menedżera kolejek w ten sposób jest opisane bardziej szczegółowo w produkcie “Dodawanie informacji konfiguracyjnych do menedżera kolejek” na stronie 49.

- W systemach Windows i Linux (platformyx86 i x86-64) należy użyć programu WebSphere MQ Explorer. Określ szczegóły pliku ładowania przełącznika na panelu właściwości menedżera kolejek pod kontrolą menedżera zasobów XA.
- We wszystkich pozostałych systemach należy określić szczegóły pliku ładowania przełącznika w sekcji XAResourceManager w pliku qm.ini menedżera kolejek.

Rysunek 11 na stronie 59 to przykład systemu UNIX , który przedstawia wpis qm.ini XAResourceManager , w którym baza danych, która ma być koordynowana, nosi nazwę `mydbname`, przy czym nazwa ta jest określana w `XAOpenString`:

```
XAResourceManager:
  Name=myinformix
  SwitchFile=infswit
  XAOpenString=DB=mydbname@myinformixserver\;USER=myuser\;PASSWD=mypasswd
  ThreadOfControl=THREAD
```

Rysunek 11. Przykładowa pozycja XAResourceManager dla bazy danych Informix na platformach UNIX

Uwaga: Domyślnie przykładowy plik xaswit.mak na platformach UNIX tworzy plik ładowania przełącznika, który korzysta z wielowątkowych bibliotek Informix . Podczas korzystania z tych bibliotek Informix należy upewnić się, że parametr ThreadOfControl jest ustawiony na THREAD. In Rysunek 11 na stronie 59, the qm.ini file XAResourceManager stanza attribute ThreadOfControl is set to THREAD. Jeśli określono parametr THREAD, aplikacje muszą być budowane przy użyciu wielowątkowych bibliotek produktu Informix oraz bibliotek interfejsu API wielowątkowych WebSphere MQ .

Atrybut XAOpenString musi zawierać nazwę bazy danych, po której znajduje się symbol @, a następnie nazwę serwera Informix .

To use the nonthreaded Informix libraries, you must ensure that the qm.ini file XAResourceManager stanza attribute ThreadOfControl is set to PROCESS. Należy również wprowadzić następujące zmiany w przykładowym pliku xaswit.mak:

1. Usuń komentarz z generowania pliku ładowania przełączanego w trybie niewątkowym.
2. Przekształć w komentarz generowanie pliku ładowania z przełącznikiem wielowątkowym.

Konfiguracja Sybase

Informacje na temat obsługi i konfiguracji bazy danych Sybase .

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Wykonaj następujące czynności:

1. Upewnij się, że zainstalowano biblioteki XA produktu Sybase , na przykład instalując opcję XA DTM.
2. Sprawdź ustawienia zmiennych środowiskowych.
3. Włącz obsługę interfejsu Sybase XA.
4. Utwórz plik ładowania przełącznika Sybase .
5. Dodaj informacje o konfiguracji menedżera zasobów.

Bieżąca lista poziomów produktu Sybase obsługiwana przez produkt WebSphere MQ jest dostępna na stronie IBM WebSphere MQ detailed system requirements .

Sprawdzanie ustawień zmiennych środowiskowych Sybase

Upewnij się, że zmienne środowiskowe Sybase są ustawione dla procesów menedżera kolejek ***, jak również*** procesów aplikacji. W szczególności należy zawsze ustawić następujące zmienne środowiskowe **przed** uruchomieniem menedżera kolejek:

SYBASE
Położenie instalacji produktu Sybase . Na przykład w systemach UNIX and Linux należy użyć następujących elementów:

```
export SYBASE=/sybase
```

W systemach Windows należy użyć:

```
set SYBASE=c:\sybase
```

SYBASE_OCS
Katalog w bazie SYBASE, w którym zainstalowano pliki klienta Sybase . Na przykład w systemach UNIX and Linux należy użyć następujących elementów:

```
export SYBASE_OCS=OCS-12_0
```

W systemach Windows należy użyć:

```
set SYBASE_OCS=OCS-12_0
```

Włączanie obsługi interfejsu Sybase XA

W pliku konfiguracyjnym Sybase XA $SYBASE/$SYBASE_OCS/xa_configzdefiniuj logiczny Resource Manager (LRM) dla każdego połączenia z serwerem Sybase , który jest aktualizowany. Przykład zawartości pliku $SYBASE/$SYBASE_OCS/xa_config jest przedstawiony w sekcji Rysunek 12 na stronie 61.

```
# The first line must always be a comment

[xa]

  LRM=lrmname
  server=servername
```

Rysunek 12. Przykładowa treść pliku $SYBASE/ $SYBASE_OCS/xa_config

Tworzenie pliku ładowania przełącznika Sybase

Aby utworzyć plik ładowania przełącznika Sybase , należy użyć przykładowych plików dostarczonych z produktem WebSphere MQ.W systemach Windows plik xaswit.mak znajduje się w katalogu `C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm`. Aby utworzyć plik ładowania przełącznika Sybase przy użyciu programu Microsoft Visual C + +, należy użyć następującej komendy:

```
nmake /f xaswit.mak sybswit.dll
```

Wygenerowany plik przełącznika jest umieszczany w produkcie `c:\Program Files\IBM\WebSphere MQ\exits`.

Plik xaswit.mak znajduje się w katalogu *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .

Zmodyfikuj plik xaswit.mak na *usuń komentarz* z wierszy właściwych dla używanej wersji bazy danych Sybase . Następnie wykonaj komendę makefile, używając komendy:

```
make -f xaswit.mak sybswit
```

Wygenerowany 32-bitowy plik ładowania przełącznika jest umieszczany w produkcie `/var/mqm/exits`.

Wygenerowany 64-bitowy plik ładowania przełącznika jest umieszczany w produkcie `/var/mqm/exits64`.

Dodawanie informacji konfiguracyjnych menedżera zasobów dla bazy danych Sybase

Należy zmodyfikować informacje konfiguracyjne dla menedżera kolejek, aby zadeklarować produkt Sybase jako uczestnik globalnych jednostek pracy. Modyfikowanie informacji konfiguracyjnych jest bardziej szczegółowo opisane w sekcji “Dodawanie informacji konfiguracyjnych do menedżera kolejek” na stronie 49.

- W systemach Windows i Linux (platformyx86 i x86-64) należy użyć programu WebSphere MQ Explorer. Określ szczegóły pliku ładowania przełącznika na panelu właściwości menedżera kolejek pod kontrolą menedżera zasobów XA.
- We wszystkich pozostałych systemach należy określić szczegóły pliku ładowania przełącznika w sekcji XAResourceManager w pliku qm.ini menedżera kolejek.

Rysunek 13 na stronie 62 przedstawia przykład produktu UNIX and Linux , który korzysta z bazy danych powiązanej z definicją *lrmname* LRM w pliku konfiguracyjnym Sybase XA, $SYBASE/$SYBASE_OCS/`xa_config`. Jeśli chcesz, aby wywołania funkcji XA były protokołowane, należy podać nazwę pliku dziennika:

```
XAResourceManager:
  Name=mysybase
  SwitchFile=sybswit
  XAOpenString=-Uuser -Ppassword -Nlrmname -L/tmp/sybase.log -Txa
  ThreadOfControl=THREAD
```

Rysunek 13. Przykładowy wpis XAResourceManager dla bazy danych Sybase na platformach UNIX and Linux

Korzystanie z programów wielowątkowych za pomocą programu Sybase

Jeśli używane są programy wielowątkowe z globalnymi jednostkami pracy WebSphere MQ uwzględniających aktualizacje Sybase, użytkownik ***musi*** użyć wartości THREAD dla parametru ThreadOfControl. Upewnij się również, że program (i plik ładowania przełącznika) został dowiązany z bibliotekami Sybase (wersje _r) z bezpieczną ochroną wątków. Użycie wartości THREAD dla parametru ThreadOfControl jest przedstawione w sekcji Rysunek 13 na stronie 62.

Wiele konfiguracji bazy danych

Aby skonfigurować menedżer kolejek w taki sposób, aby aktualizacje wielu baz danych mogły zostać uwzględnione w globalnych jednostkach pracy, należy dodać sekcję XAResourceManager dla każdej bazy danych.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Jeśli wszystkie bazy danych są zarządzane przez ten sam menedżer bazy danych, każda sekcja definiuje osobną bazę danych. Każda sekcja określa ten sam plik *SwitchFile*, ale zawartość pliku *XAOpenString* jest inna, ponieważ określa nazwę aktualizowanej bazy danych. Na przykład sekcje pokazywane w programie Rysunek 14 na stronie 62 umożliwiają skonfigurowanie menedżera kolejek za pomocą baz danych Db2 *MQBankDB* i *MQFeeDB* w systemach UNIX and Linux .

Ważne: Nie można utworzyć wielu sekcji wskazujących tę samą bazę danych. Ta konfiguracja nie działa w żadnych okolicznościach, a jeśli ta konfiguracja zostanie wypróbowana, nie powiedzie się.

Zostaną wyświetlone błędy w postaci `when the MQ code makes its second xa_open call in any process in this environment, the database software fails the second xa_open with a -5 error, XAER_INVAL`.

```
XAResourceManager:
  Name=DB2 MQBankDB
  SwitchFile=db2swit
  XAOpenString=MQBankDB

XAResourceManager:
  Name=DB2 MQFeeDB
  SwitchFile=db2swit
  XAOpenString=MQFeeDB
```

Rysunek 14. Przykładowe wpisy XAResourceManager dla wielu baz danych Db2

Jeśli bazy danych, które mają zostać zaktualizowane, są zarządzane przez różne menedżery baz danych, dodaj dla każdego z nich sekcję XAResourceManager . W tym przypadku każda sekcja określa inny plik *SwitchFile*. Na przykład, jeśli *MQFeeDB* jest zarządzane przez Oracle zamiast bazy danych DB2, użyj następujących sekcji w systemach UNIX and Linux :

```
XAResourceManager:
  Name=DB2 MQBankDB
  SwitchFile=db2swit
  XAOpenString=MQBankDB

XAResourceManager:
  Name=Oracle MQFeeDB
  SwitchFile=oraswit
  XAOpenString=Oracle_XA+Acc=P/myuser/mypassword+SesTm=35+LogDir=/tmp/ora.log+DB=MQFeeDB
```

Rysunek 15. Przykładowe wpisy XAResourceManager dla bazy danych DB2 i bazy danych Oracle

W zasadzie nie ma limitu liczby instancji bazy danych, które można skonfigurować za pomocą pojedynczego menedżera kolejek.

Uwaga: Aby uzyskać informacje na temat obsługi baz danych Informix w wielu aktualizacjach bazy danych w globalnych jednostkach pracy, należy sprawdzić plik readme produktu.

Zagadnienia związane z zabezpieczeniami

Uwagi dotyczące uruchamiania bazy danych w modelu XA.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Poniższe informacje podano wyłącznie w celach informacyjnych. We wszystkich przypadkach należy zapoznać się z dokumentacją dostarczonej wraz z menedżerem bazy danych, aby określić konsekwencje związane z bezpieczeństwem uruchamiania bazy danych w modelu XA.

Proces aplikacji oznacza początek globalnej jednostki pracy przy użyciu komendy `MQBEGIN` . Pierwsze wywołanie `MQBEGIN` , które wywołuje problemy z aplikacją, łączy się ze wszystkimi uczestniczącymi bazami danych, wywołując kod biblioteki klienta w punkcie wejścia xa_open. Wszystkie menedżery baz danych udostępniają mechanizm dostarczania ID użytkownika i hasła w ich XAOpenString. Jest to jedyny czas, w którym informacje o uwierzytelnianiu będą przepływać.

Należy zauważyć, że na platformach UNIX and Linux aplikacje fastpath muszą być uruchamiane z efektywnym identyfikatorem użytkownika mqm podczas wykonywania wywołań MQI.

Uwagi dotyczące utraty kontaktu z menedżerem zasobów XA

Menedżer kolejek toleruje, że menedżery baz danych nie są dostępne. Oznacza to, że menedżer kolejek może być uruchamiany i zatrzymany niezależnie od serwera bazy danych. Po odtworzeniu kontaktu menedżer kolejek i resynchronizacja bazy danych są resynchronizowane. Można również użyć komendy rsvmqtrn, aby ręcznie rozstrzygnąć wątpliwe jednostki pracy.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

W normalnych operacjach tylko minimalna ilość czynności administracyjnych jest niezbędna po zakończeniu czynności konfiguracyjnych. Zadanie administracyjne jest łatwiejsze, ponieważ menedżer kolejek toleruje, że menedżery baz danych nie są dostępne. W szczególności oznacza to, że:

- Menedżer kolejek może zostać uruchomiony w dowolnym momencie bez pierwszego uruchomienia każdego z menedżerów bazy danych.
- Menedżer kolejek nie musi być zatrzymany i restartowany, jeśli jeden z menedżerów bazy danych stanie się niedostępny.

Pozwala to na uruchamianie i zatrzymywanie menedżera kolejek niezależnie od serwera bazy danych.

Za każdym razem, gdy między menedżerem kolejek a bazą danych zostanie utracony kontakt, muszą one zostać ponownie zsynchronizowane, gdy oba te elementy staną się dostępne ponownie. Resynchronizacja to proces, za pomocą którego wykonywane są wszystkie wątpliwe jednostki pracy związane z tą

bazą danych. Generalnie jest to wykonywane automatycznie bez konieczności interwencji użytkownika. Menedżer kolejek zwraca się do bazy danych o listę jednostek pracy, które są wątpliwe. Następnie instruuje bazę danych, aby zatwierdzała lub wycofała każdą z tych wątpliwych jednostek pracy.

Gdy uruchamiany jest menedżer kolejek, resynchronizuje z każdą bazą danych. Gdy pojedyncza baza danych stanie się niedostępna, należy ponownie zsynchronizować tę bazę danych z następnym razem, gdy menedżer kolejek zauważe, że jest ona dostępna ponownie.

Menedżer kolejek odzyskuje kontakt z wcześniej niedostępną bazą danych automatycznie, ponieważ nowe globalne jednostki pracy są uruchamiane za pomocą komendy MQBEGIN. W tym celu należy wywołanie funkcji xa_open w bibliotece klienta bazy danych. Jeśli wywołanie funkcji xa_open zakończy się niepowodzeniem, komenda MQBEGIN zwróci kod zakończenia MQCC_WARNING i kod przyczyny MQRC_JDBIPANT_NOT_AVAILABLE. Można ponowić wywołanie komendy MQBEGIN w późniejszym czasie.

Nie należy kontynuować próby wykonania globalnej jednostki pracy, która obejmuje aktualizacje bazy danych, która wskazała niepowodzenie podczas wykonywania komendy MQBEGIN. Nie będzie połączenia z tą bazą danych, za pomocą której mogą być wykonywane aktualizacje. Jedynymi opcjami są zakończenie programu lub ponowna próba wykonania komendy MQBEGIN w nadziei, że baza danych stanie się ponownie dostępna.

Alternatywnie można użyć komendy `rsvmqtrn` , aby rozstrzygnąć jawnie wszystkie wątpliwe jednostki pracy.

Wątpliwe jednostki pracy

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Baza danych może być pozostawiona z wątpliwą jednostką pracy, jeśli kontakt z menedżerem kolejek zostanie utracony po tym, jak menedżer bazy danych został poinstruowany w celu przygotowania. Dopóki serwer bazy danych nie otrzyma wyniku z menedżera kolejek (zatwierdź lub wycofaj zmiany), musi on zachować blokady bazy danych powiązane z aktualizacjami.

Ponieważ blokady te uniemożliwiają innym aplikacjom aktualizowanie lub odczytywanie rekordów bazy danych, resynchronizacja musi odbywać się tak szybko, jak to możliwe.

Jeśli z jakiegoś powodu nie można czekać na automatyczne ponowne zsynchronizowanie z bazą danych przez menedżer kolejek, można użyć udogodnień udostępnianych przez menedżera bazy danych w celu ręcznego zatwierdzenia lub wycofania zmian w bazie danych. W specyfikacji *X/Open Distributed Transaction Processing: The XA Specification*(Specyfikacja XA: Specyfikacja XA) jest to nazywane decyzją *heurystyczną* . Używaj go tylko w ostateczności ze względu na możliwość naruszenia integralności danych; możesz na przykład omyłkowo wycofać aktualizacje bazy danych, gdy wszyscy inni uczestnicy zatwierdzili swoje aktualizacje.

Znacznie lepiej jest restartować menedżera kolejek lub użyć komendy `rsvmqtrn` po zrestartowaniu bazy danych, aby zainicjować automatyczną resynchronizację.

Wyświetlanie zaległych jednostek pracy za pomocą komendy dspmqtrn

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Gdy menedżer bazy danych jest niedostępny, można użyć komendy **dspmqtrn** w celu sprawdzenia stanu zaległych globalnych jednostek pracy związanych z tą bazą danych.

Komenda **dspmqtrn** wyświetla tylko te jednostki pracy, w których co najmniej jeden uczestnik ma wątpliwości. Uczestnicy oczekują na decyzję od menedżera kolejek w celu zatwierdzenia lub wycofania przygotowanych aktualizacji.

Dla każdej z tych globalnych jednostek pracy stan każdego uczestnika jest wyświetlany w danych wyjściowych komendy **dspmqtrn**. Jeśli jednostka pracy nie zaktualizuje zasobów określonego menedżera zasobów, nie jest ona wyświetlana.

Jeśli chodzi o wątpliwe jednostki pracy, menedżer zasobów jest w stanie wykonać jedną z następujących czynności:

Przygotowany
Menedżer zasobów jest przygotowany do zatwierdzenia jego aktualizacji.

Zatwierdzone
Menedżer zasobów zatwierdził swoje aktualizacje.

Wycofano
Menedżer zasobów wycofał swoje aktualizacje.

Uczestniczył
Menedżer zasobów jest uczestnikiem, ale nie został przygotowany, zatwierdzony lub wycofany z aktualizacji.

Po zrestartowaniu menedżera kolejek każda baza danych zawiera sekcję XAResourceManager , która zawiera listę swoich wątpliwych globalnych jednostek pracy. Jeśli baza danych nie została zrestartowana lub jest niedostępna, menedżer kolejek nie może jeszcze dostarczyć do bazy danych końcowych wyników dla tych jednostek pracy. Wynik wątpliwych jednostek pracy jest dostarczany do bazy danych przy pierwszej okazji, gdy baza danych jest ponownie dostępna.

W takim przypadku menedżer bazy danych jest raportowana jako stan *przygotowany* do czasu wystąpienia resynchronizacji.

Za każdym razem, gdy komenda `dspmqtrn` wyświetla wątpliwe jednostki pracy, w pierwszej kolejności wyświetlane są wszystkie możliwe menedżery zasobów, które mogą uczestniczyć w pracy. Są one przydzielane unikalnym identyfikatorem *RMId*, który jest używany zamiast *Nazwa* menedżerów zasobów podczas raportowania ich stanu w odniesieniu do wątpliwej jednostki pracy.

Przykładowe dane wyjściowe dspmqtrn przedstawiają wynik wywołania następującej komendy:

```
dspmqtrn -m MY_QMGR
```

```
AMQ7107: Resource manager 0 is MQSeries.
AMQ7107: Resource manager 1 is DB2 MQBankDB.
AMQ7107: Resource manager 2 is DB2 MQFeeDB.

AMQ7056: Transaction number 0,1.
    XID: formatID 5067085, gtrid_length 12, bqual_length 4
         gtrid [3291A506000020137465737 4]
         bqual [00000001]
AMQ7105: Resource manager 0 has committed.
AMQ7104: Resource manager 1 has prepared.
AMQ7104: Resource manager 2 has prepared.
```

gdzie *numer_transakcji* to identyfikator transakcji, która może być używana z komendą rsvmqtrn . Więcej informacji na temat komunikatu AMQ7056 zawiera sekcja AMQ7000-7999: produkt WebSphere MQ . Zmienne *XID* są częścią specyfikacji *X/Open XA Specification*. Są to najbardziej aktualne informacje na temat tej specyfikacji, patrz: https://publications.opengroup.org/c193.

Rysunek 16. Przykładowe dane wyjściowe dspmqtrn

Dane wyjściowe w pliku Przykładowe dane wyjściowe dspmqtrn wskazują, że istnieją trzy menedżery zasobów powiązane z menedżerem kolejek. Pierwszym z nich jest menedżer zasobów `0`, który jest menedżerem kolejek. The other two resource manager instances are the `MQBankDB` and `MQFeeDB` Db2 databases.

W przykładzie przedstawiono tylko jedną wątpliwe jednostki pracy. Komunikat jest generowany dla wszystkich trzech menedżerów zasobów, co oznacza, że aktualizacje zostały wprowadzone do menedżera kolejek i obu baz danych Db2 w ramach jednostki pracy.

Aktualizacje wprowadzone w menedżerze kolejek, menedżerze zasobów **0**, zostały *zatwierdzone*. Aktualizacje baz danych produktu Db2 są w stanie *przygotowane* , co oznacza, że produkt Db2 musi stać się niedostępny, zanim został wywołany w celu zatwierdzenia aktualizacji baz danych *MQBankDB* i *MQFeeDB* .

Wątpliwa jednostka pracy ma zewnętrzny identyfikator o nazwie XID (*identyfikator transakcji*). Jest to fragment danych podany w programie Db2 przez menedżera kolejek w celu zidentyfikowania jego części globalnej jednostki pracy.

Rozstrzyganie zaległych jednostek pracy za pomocą komendy rsvmqtrn
Zaległe jednostki pracy są kompletne, gdy menedżer kolejek i DB2 resynchronizują.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Dane wyjściowe przedstawione w programie Rysunek 16 na stronie 65 przedstawiają pojedynczą wątpliwość jednostki pracy, w której decyzja zatwierdzana musi być jeszcze dostarczona do obu baz danych DB2 .

Aby ukończyć tę jednostkę pracy, menedżer kolejek i baza danych DB2 muszą resynchronizować, gdy produkt DB2 będzie dostępny. Menedżer kolejek korzysta z uruchamiania nowych jednostek pracy jako możliwości odzyskania kontaktu z bazą danych DB2. Alternatywnie można poinstruować menedżera kolejek, aby resynchronizował jawnie za pomocą komendy **rsvmqtrn** .

Należy to zrobić wkrótce po zrestartowaniu bazy danych DB2 , tak aby wszystkie blokady bazy danych powiązane z wątpliwo-wątpliwe jednostki pracy były zwalniane tak szybko, jak to możliwe. Użyj opcji -a, która informuje menedżera kolejek o rozstrzygnięcie wszystkich wątpliwych jednostek pracy. W poniższym przykładzie program DB2 został zrestartowany, dlatego menedżer kolejek może rozstrzygnąć wątpliwe jednostki pracy:

```
> rsvmqtrn -m MY_QMGR -a
Any in-doubt transactions have been resolved.
```

Mieszane rezultaty i błędy
Mimo że menedżer kolejek używa protokołu zatwierdzania dwufazowego, nie powoduje to całkowitego usunięcia możliwości wykonania niektórych jednostek pracy z mieszanymi wynikami. W tym miejscu niektórzy uczestnicy zatwierdzają swoje aktualizacje, a niektóre z nich są uaktualniani.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Jednostki pracy zakończone z wynikiem mieszanym mają poważne konsekwencje, ponieważ zasoby współużytkowane, które powinny zostać zaktualizowane jako pojedyncza jednostka pracy, nie są już w stanie spójnym.

Mieszane wyniki są głównie spowodowane, gdy podejmowane są decyzje heurystyczne o jednostkach pracy, zamiast zezwalać menedżerowi kolejek na rozstrzyganie w wątpliwych jednostkach samego działania. Takie decyzje są poza kontrolą menedżera kolejek.

Za każdym razem, gdy menedżer kolejek wykryje mieszany wynik, generuje informacje FFST i dokumentuje niepowodzenie w dziennikach błędów przy użyciu jednego z dwóch komunikatów:

- Jeśli menedżer bazy danych wycofuje się, zamiast zatwierdzać:

```
AMQ7606 A transaction has been committed but one or more resource
        managers have rolled back.
```

- Jeśli menedżer bazy danych zatwierdzil zamiast wycofywanie zmian:

```
AMQ7607 A transaction has been rolled back but one or more resource
        managers have committed.
```

Dalsze komunikaty identyfikują bazy danych, które są heurystycznie uszkodzone. Jest to wówczas Twoja odpowiedzialność za lokalne przywrócenie spójności do odpowiednich baz danych. Jest to skomplikowana procedura, w której najpierw należy wyizolować aktualizację, która została niepoprawnie zatwierdzona lub wycofana, a następnie ręcznie cofnąć lub przywrócić zmiany w bazie danych.

Zmiana informacji konfiguracyjnych
Po pomyślnym uruchomieniu menedżera kolejek w celu koordynowania globalnych jednostek pracy nie należy zmieniać żadnych informacji o konfiguracji menedżera zasobów.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Jeśli konieczna jest zmiana informacji konfiguracyjnych, można to zrobić w dowolnym momencie, ale zmiany te nie zostaną uwzględnione dopiero po zrestartowaniu menedżera kolejek.

Jeśli informacje konfiguracyjne menedżera zasobów zostaną usunięte dla bazy danych, można skutecznie usunąć możliwość skontaktowania się z menedżerem kolejek w celu skontaktowania się z menedżerem bazy danych.

Nigdy zmień atrybut *Nazwa* w dowolnym z informacji konfiguracyjnych menedżera zasobów. Ten atrybut jednoznacznie identyfikuje instancję menedżera bazy danych dla menedżera kolejek. Jeśli ten unikalny identyfikator zostanie zmieniony, menedżer kolejek przyjmuje, że baza danych została usunięta i została dodana zupełnie nowa instancja. Menedżer kolejek nadal wiąże oczekujące jednostki pracy ze starą *nazwą*, prawdopodobnie pozostawiając bazę danych w stanie wątpliwej.

Usuwanie instancji menedżera bazy danych
Jeśli konieczne jest trwałe usunięcie bazy danych z konfiguracji, przed zrestartowaniem menedżera kolejek należy upewnić się, że baza danych nie jest wątpliwa.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Produkty bazodanowe udostępniają komendy służące do wyświetlania listy wątpliwych transakcji. Jeśli istnieją jakiekolwiek transakcje wątpliwe, należy najpierw zezwolić menedżerowi kolejek na resynchronizację z bazą danych. W tym celu należy uruchomić menedżer kolejek. Istnieje możliwość sprawdzenia, czy resynchronizacja została wykonana za pomocą komendy **rsvmqtrn** lub własnej komendy bazy danych w celu wyświetlenia wątpliwych jednostek pracy. Po upewnieniu się, że resynchronizacja miała miejsce, zakończysz menedżer kolejek i usuń informacje o konfiguracji bazy danych.

Jeśli ta procedura nie powiedzie się, menedżer kolejek nadal pamięta wszystkie wątpliwe jednostki pracy dotyczące tej bazy danych. Komunikat ostrzegawczy AMQ7623jest generowany za każdym razem, gdy menedżer kolejek jest restartowany. Jeśli nigdy nie zamierzasz ponownie skonfigurować tej bazy danych przy użyciu menedżera kolejek, użyj opcji -r komendy **rsvmqtrn** , aby poinformować menedżera kolejek o tym, aby zapomniał o udziale bazy danych w jego wątpliwych transakcjach. Menedżer kolejek zapomina o takich transakcjach tylko wtedy, gdy wątpliwe transakcje zostały zakończone ze wszystkimi uczestnikami.

Czasami może być konieczne tymczasowe usunięcie niektórych informacji konfiguracyjnych menedżera zasobów. W systemach UNIX and Linux najlepiej jest to osiągnąć, komentując sekcję, tak aby można ją było łatwo przywrócić w późniejszym czasie. Można to zrobić, jeśli wystąpią błędy przy każdym kontaktowaniu się menedżera kolejek z określoną bazą danych lub menedżerem bazy danych. Tymczasowe usunięcie danych konfiguracyjnych menedżera zasobów pozwala menedżerowi kolejek na uruchamianie globalnych jednostek pracy z udziałem wszystkich pozostałych uczestników. Poniżej przedstawiono przykład sekcji skomentowanych XA`ResourceManager` :

```
# This database has been temporarily removed
#XAResourceManager:
#  Name=mydb2
#  SwitchFile=db2swit
#  XAOpenString=mydbname,myuser,mypassword,toc=t
#  ThreadOfControl=THREAD
```

Rysunek 17. Skomentowana sekcja XAResourceManager w systemach UNIX and Linux

W systemach Windows należy użyć programu WebSphere MQ Explorer w celu usunięcia informacji o instancji menedżera bazy danych. Podczas ponownego wprowadzenia nazwy w polu *Nazwa* należy zwrócić szczególną uwagę na poprawną nazwę. Jeśli nazwa jest mistype, mogą wystąpić problemy z wątpliwostkami, zgodnie z opisem w sekcji “Zmiana informacji konfiguracyjnych” na stronie 67.

Rejestracja dynamiczna XA

Specyfikacja interfejsu XA umożliwia skrócenie liczby wywołań programu `xa_*` , które menedżer transakcji udostępnia do menedżera zasobów. Ta optymalizacja jest znana jako *rejestracja dynamiczna*.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Rejestracja dynamiczna jest obsługiwana przez produkt DB2. Inne bazy danych mogą go obsługiwać. Szczegółowe informacje można znaleźć w dokumentacji produktu bazodanowego.

Dlaczego dynamiczna optymalizacja rejestracji jest przydatna? W aplikacji niektóre globalne jednostki pracy mogą zawierać aktualizacje tabel bazy danych; inne mogą nie zawierać takich aktualizacji. Jeśli do tabel bazy danych nie została wykonana żadna trwała aktualizacja, nie ma potrzeby włączania tej bazy danych do protokołu zatwierdzania, który występuje podczas operacji MQCMIT.

Niezależnie od tego, czy baza danych obsługuje rejestrację dynamiczną, aplikacja wywołuje produkt `xa_open` podczas pierwszego wywołania komendy MQBEGIN w połączeniu z produktem WebSphere MQ . Wywołuje ona również `xa_close` w kolejnych wywołaniach MQDISC. Wzorzec kolejnych wywołań XA zależy od tego, czy baza danych obsługuje rejestrację dynamiczną:

Jeśli baza danych nie obsługuje rejestracji dynamicznej ...
Każda globalna jednostka pracy obejmuje kilka wywołań funkcji XA wykonanych przez kod WebSphere MQ w bibliotece klienta bazy danych, niezależnie od tego, czy w ramach jednostki pracy dokonano trwałej aktualizacji tabel tej bazy danych. Są to:

- `xa_start` i `xa_end` z procesu aplikacji. Są one używane do deklarowania początku i końca globalnej jednostki pracy.
- `xa_prepare`, `xa_commit`i `xa_rollback` z procesu agenta menedżera kolejek, amqzlaa0. Są one używane do dostarczania wyników globalnej jednostki pracy: decyzja w sprawie zatwierdzenia lub wycofania zmian.

Ponadto proces agenta menedżera kolejek wywołuje również program `xa_open` podczas pierwszej komendy MQBEGIN.

Jeśli baza danych obsługuje rejestrację dynamiczną ...
Kod WebSphere MQ wykonuje tylko te wywołania funkcji XA, które są niezbędne. W przypadku globalnej jednostki pracy, która **nie** jest związana z trwałymi aktualizacjami zasobów bazy danych, **nie** jest wywołań XA do bazy danych. W przypadku globalnej jednostki pracy, która **zawiera** takie trwałe aktualizacje, wywołania mają następujące wartości:

- `xa_end` z procesu aplikacji, aby zadeklarować zakończenie globalnej jednostki pracy.
- `xa_prepare`, `xa_commit`i `xa_rollback` z procesu agenta menedżera kolejek, amqzlaa0. Są one używane do dostarczania wyników globalnej jednostki pracy: decyzja w sprawie zatwierdzenia lub wycofania zmian.

W przypadku dynamicznej rejestracji do pracy istotne jest, aby baza danych informowała produkt WebSphere MQ , gdy wykonała trwałą aktualizację, która ma zostać włączona do bieżącej globalnej jednostki pracy. Produkt WebSphere MQ udostępnia w tym celu funkcję `ax_reg` .

Kod klienta bazy danych, który jest uruchamiany w procesie aplikacji, znajduje funkcję `ax_reg` i wywołuje ją, w celu *dynamicznego rejestrowania* faktu, że wykonała trwałe działanie w bieżącej globalnej jednostce pracy. W odpowiedzi na to wywołanie funkcji `ax_reg` , WebSphere MQ rejestruje, że baza danych uczestniczyła. Jeśli jest to pierwsze wywołanie produktu `ax_reg` w tym połączeniu z produktem WebSphere MQ , proces agenta menedżera kolejek wywołuje `xa_open`.

Kod klienta bazy danych powoduje, że jest to wywołanie funkcji `ax_reg` , gdy jest on uruchomiony w procesie, na przykład podczas wywołania SQL UPDATE lub dowolnego wywołania w interfejsie API klienta bazy danych.

Warunki błędu
W przypadku rejestracji dynamicznej XA istnieje możliwość pomylenia się w menedżerze kolejek.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Częstym przykładem jest to, że jeśli przed uruchomieniem menedżera kolejek zostaną poprawnie ustawione zmienne środowiskowe bazy danych, wywołania menedżera kolejek w programie `xa_open` nie powiodą się. Żadne globalne jednostki pracy nie mogą być używane.

Aby tego uniknąć, należy przed uruchomieniem menedżera kolejek upewnić się, że zostały ustawione odpowiednie zmienne środowiskowe. Zapoznaj się z dokumentacją produktu bazy danych i poradą podanymi w produktach “Konfigurowanie Db2” na stronie 52, “Konfigurowanie produktu Oracle” na stronie 54i “Konfiguracja Sybase” na stronie 60.

W przypadku wszystkich produktów bazodanowych menedżer kolejek wywołuje funkcję `xa_open` raz podczas uruchamiania menedżera kolejek w ramach sesji odtwarzania (zgodnie z opisem w sekcji “Uwagi dotyczące utraty kontaktu z menedżerem zasobów XA” na stronie 63). Wywołanie tej `xa_open` nie powiedzie się, jeśli zmienne środowiskowe bazy danych zostaną ustawione niepoprawnie, ale nie spowoduje to, że menedżer kolejek nie zostanie uruchomiony. Jest to spowodowane tym, że ten sam kod błędu produktu `xa_open` jest używany przez bibliotekę klienta bazy danych w celu wskazania, że serwer bazy danych jest niedostępny. Produkt WebSphere MQ nie traktuje tego jako poważnego błędu, ponieważ menedżer kolejek musi być w stanie uruchomić przetwarzanie danych poza globalnymi jednostkami pracy, które dotyczą tej bazy danych.

Kolejne wywołania programu `xa_open` są wykonywane z menedżera kolejek podczas pierwszej komendy MQBEGIN w połączeniu z produktem WebSphere MQ (jeśli rejestracja dynamiczna nie jest używana) lub podczas wywoływania przez kod klienta bazy danych do funkcji `ax_reg` udostępnianej przez produkt WebSphere MQ(jeśli używana jest rejestracja dynamiczna).

Czas wystąpienia jakichkolwiek warunków błędu (lub, czasami, raportów FFST) zależy od tego, czy używana jest rejestracja dynamiczna:

- Jeśli używana jest rejestracja dynamiczna, wywołanie funkcji MQBEGIN może się powieść, ale wywołanie SQL UPDATE (lub podobnych) bazy danych nie powiedzie się.
- Jeśli rejestracja dynamiczna nie jest używana, wywołanie MQBEGIN nie powiedzie się.

Upewnij się, że zmienne środowiskowe są poprawnie ustawione w procesach aplikacji i menedżera kolejek.

Podsumowywanie wywołań XA
Poniżej znajduje się lista wywołań, które są wykonywane w funkcjach XA w bibliotece klienta bazy danych w wyniku różnych wywołań MQI, które sterują globalnymi jednostkami pracy. Nie jest to pełny opis protokołu opisanego w specyfikacji XA. Jest on dostępny jako krótki przegląd.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Należy zauważyć, że wywołania `xa_start` i `xa_end` są zawsze wywoływane przez kod WebSphere MQ w procesie aplikacji, natomiast `xa_prepare`, `xa_commit`i `xa_rollback` są zawsze wywoływane z procesu agenta menedżera kolejek, amqzlaa0.

Wywołania `xa_open` i `xa_close` pokazywane w tej tabeli są wykonywane z poziomu procesu aplikacji. Proces agenta menedżera kolejek wywołuje `xa_open` w okolicznościach opisanych w sekcji “Warunki błędu” na stronie 69.

Tabela 7. Podsumowanie wywołań funkcji XA

Wywołanie MQI	**Wywołania XA z dynamiczną rejestracją**	**Wywołania XA wykonane bez rejestracji dynamicznej**
Pierwsze MQBEGIN	`xa_open`	`xa_open` `xa_start`
Kolejne komendy MQBEGIN	Brak wywołań XA	`xa_start`

MQCMIT (**bez** `ax_reg` wywoływany podczas bieżącej globalnej jednostki pracy)	Brak wywołań XA	`xa_end`
		`xa_prepare`
		`xa_commit`
		`xa_rollback`

MQCMIT (**z** `ax_reg` wywoływanym podczas bieżącej globalnej jednostki pracy)	`xa_end`
	`xa_prepare`
	`xa_commit`
	`xa_rollback`
		Nie dotyczy. W trybie innym niż dynamiczny wywołania nie są wykonywane w trybie ax_reg.

MQBACK (**bez** `ax_reg` wywoływane podczas bieżącej globalnej jednostki pracy)	Brak wywołań XA	`xa_end` `xa_rollback`
MQBACK (**z** `ax_reg` wywoływanym podczas bieżącej globalnej jednostki pracy)	`xa_end` `xa_rollback`	Nie dotyczy. W trybie innym niż dynamiczny wywołania nie są wykonywane w trybie ax_reg.
MQDISC, gdzie najpierw wywołano komendę MQCMIT lub MQBACK. Jeśli tak nie było, przetwarzanie MQCMIT jest wykonywane po raz pierwszy podczas operacji MQDISC.	`xa_close`	`xa_close`

Uwagi:

1. W przypadku komendy MQCMIT program `xa_commit` jest wywoływany, jeśli program `xa_prepare` zakończy się pomyślnie. W przeciwnym razie wywoływana jest wartość `xa_rollback` .

Scenariusz 2: Inne oprogramowanie zapewnia koordynację

W scenariuszu 2 zewnętrzny menedżer transakcji koordynuje globalne jednostki pracy, rozpoczynając i zatwierdzając je pod kontrolą interfejsu API menedżera transakcji. Czasowniki MQBEGIN, MQCMIT i MQBACK są niedostępne.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

W tej sekcji opisano ten scenariusz, w tym:

- “Koordynacja zewnętrznej punktu synchronizacji” na stronie 71
- “Korzystanie z programu CICS” na stronie 73
- “Korzystanie z serwera Microsoft Transaction Server (COM +)” na stronie 78

Klient IBM WebSphere MQ dla produktu HP Integrity NonStop Server może używać narzędzia HP NonStop Transaction Management Facility (TMF) do koordynowania globalnych jednostek pracy. Więcej informacji na ten temat zawiera sekcja Korzystanie z programu HP NonStop TMF.

Koordynacja zewnętrznej punktu synchronizacji

Globalna jednostka pracy może być również koordynowana przez zewnętrzny menedżer transakcji zgodny z interfejsem XA X/Open. W tym miejscu menedżer kolejek produktu WebSphere MQ uczestniczy w jednostce pracy, ale nie koordynuje jej działania.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Przepływ sterowania w globalnej jednostce pracy koordynowany przez zewnętrznego menedżera transakcji jest następujący:

1. Aplikacja informuje koordynatora zewnętrznego punktu synchronizacji (na przykład TXSeries) o tym, że chce uruchomić transakcję.
2. Koordynator punktu synchronizacji informuje znanych menedżerów zasobów, takich jak WebSphere MQ, o bieżącej transakcji.
3. Aplikacja wysyła wywołania do menedżerów zasobów powiązanych z bieżącą transakcją. Na przykład aplikacja może wydawać wywołania MQGET w produkcie WebSphere MQ.
4. Aplikacja wysyła żądanie zatwierdzenia lub wycofania do koordynatora zewnętrznego punktu synchronizacji.
5. Koordynator punktu synchronizacji kończy transakcję, wysyłając odpowiednie wywołania do każdego menedżera zasobów, zwykle używając protokołów zatwierdzania dwufazowego.

Obsługiwane poziomy zewnętrznych koordynatorów punktów synchronizacji, które mogą udostępniać proces zatwierdzania dwufazowego dla transakcji, w których uczestniczy WebSphere MQ , są definiowane w produkcie IBM WebSphere MQ detailed system requirements.

W pozostałej części tej sekcji opisano, w jaki sposób można włączyć zewnętrzne jednostki pracy.

Struktura przełącznika IBM WebSphere MQ XA

Każdy menedżer zasobów uczestniczący w zewnętrznie koordynowanej jednostce pracy musi udostępniać strukturę przełącznika XA. Ta struktura definiuje zarówno możliwości menedżera zasobów, jak i funkcje, które mają być wywoływane przez koordynatora punktu synchronizacji.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Produkt IBM WebSphere MQ udostępnia dwie wersje tej struktury:

- *MQRMIXASwitch* na potrzeby statycznego zarządzania zasobami XA
- *MQRMIXASwitchDynamic* dla dynamicznego zarządzania zasobami XA

Zapoznaj się z dokumentacją menedżera transakcji, aby określić, czy ma być używany statyczny lub dynamiczny interfejs zarządzania zasobami. W każdym przypadku, gdy menedżer transakcji obsługuje tę obsługę, zaleca się użycie dynamicznego zarządzania zasobami XA.

Niektóre 64-bitowe menedżery transakcji traktują typ *long* w specyfikacji XA jako 64-bitowe, a niektóre traktują go jako wersję 32-bitową. Produkt WebSphere MQ obsługuje oba modele:

- Jeśli menedżer transakcji jest 32-bitowy, lub menedżer transakcji jest 64-bitowy, ale traktuje typ *long* jako 32-bitowy, należy użyć pliku ładowania przełącznika z listy Tabela 8 na stronie 72.

- Jeśli menedżer transakcji jest 64-bitowy i traktuje typ *long* jako 64-bitowy, należy użyć pliku ładowania przełącznika, który znajduje się na liście w sekcji Tabela 9 na stronie 72.

Lista znanych 64-bitowych menedżerów transakcji, które traktują typ *long* jako 64-bitowe, jest dostępna w produkcie Tabela 10 na stronie 72. Zapoznaj się z dokumentacją menedżera transakcji, jeśli nie masz pewności, który model jest używany przez menedżer transakcji.

Tabela 8. Nazwy plików ładowania przełącznika XA

Platforma	**Nazwa pliku ładowania przełącznika (serwer)**	**Nazwa pliku ładowania przełącznika (rozszerzony klient transakcyjny)**
Windows	*mqmxa.dll*	*mqcxa.dll*
AIX (nonthreaded)	*libmqmxa.a*	*libmqcxa.a*
AIX (wielowątkowy)	*libmqmxa_r.a*	*libmqcxa_r.a*
HP-UX (nonthreaded)	*libmqmxa.so*	*libmqcxa.so*
HP-UX (wersja wielowątkowa)	*libmqmxa_r.so*	*libmqcxa_r.so*
Linux (nonthreaded)	*libmqmxa.so*	*libmqcxa.so*
Linux (gwintowane)	*libmqmxa_r.so*	*libmqcxa_r.so*
Solaris	*libmqmxa.so*	*libmqcxa.so*

Tabela 9. Alternatywne 64-bitowe nazwy plików ładowania przełącznika XA

Platforma	**Nazwa pliku ładowania przełącznika (serwer)**	**Nazwa pliku ładowania przełącznika (rozszerzony klient transakcyjny)**
AIX (nonthreaded)	*libmqmxa64.a*	*libmqcxa64.a*
AIX (wielowątkowy)	*libmqmxa64_r.a*	*libmqcxa64_r.a*
HP-UX (nonthreaded)	*libmqmxa64.so*	*libmqcxa64.so*
HP-UX (wersja wielowątkowa)	*libmqmxa64_r.so*	*libmqcxa64_r.so*
Linux (nonthreaded)	*libmqmxa64.so*	*libmqcxa64.so*
Linux (gwintowane)	*libmqmxa64_r.so*	*libmqcxa64_r.so*
Solaris	*libmqmxa64.so*	*libmqcxa64.so*

Tabela 10. 64-bitowe menedżery transakcji, które wymagają alternatywnego 64-bitowego pliku ładowania przełączników

Menedżer transakcji
Tuxedo

Niektóre zewnętrzne koordynatory punktów synchronizacji (nie CICS) wymagają, aby każdy menedżer zasobów uczestniczący w jednostce pracy dostarczyli swoją nazwę w polu nazwy w strukturze przełącznika XA. Nazwa menedżera zasobów produktu WebSphere MQ to MQSeries_XA_RMI.

Koordynator punktu synchronizacji definiuje sposób, w jaki struktura przełącznika WebSphere MQ XA łączy się z nim. Informacje na temat łączenia struktury przełącznika XA produktu WebSphere MQ z systemem CICS są dostępne w produkcie “Korzystanie z programu CICS” na stronie 73. Aby uzyskać informacje na temat łączenia struktury przełącznika WebSphere MQ XA z innymi koordynatorami punktów synchronizacji zgodnych z interfejsem XA, należy zapoznać się z dokumentacją dostarczonej z tymi produktami.

Poniższe uwagi dotyczą używania produktu WebSphere MQ ze wszystkimi koordynatorami punktów synchronizacji zgodnych z interfejsem XA:

- Struktura xa_info przekazana w dowolnym wywołaniu xa_open przez koordynatora punktu synchronizacji zawiera nazwę menedżera kolejek produktu WebSphere MQ . Nazwa ta ma taką samą postać, jak nazwa menedżera kolejek przekazana do wywołania `MQCONN` . Jeśli nazwa przekazana w wywołaniu xa_open jest pusta, zostanie użyty domyślny menedżer kolejek.

  Alternatywnie, struktura xa_info może zawierać wartości dla parametrów *TPM* i *AXLIB* . Parametr *TPM* określa używany menedżer transakcji. Poprawne wartości to CICS, TUXEDO i ENCINA. Parametr *AXLIB* określa nazwę biblioteki, która zawiera funkcje ax_reg i ax_unreg menedżera transakcji. Więcej informacji na temat tych parametrów znajduje się w sekcji Konfigurowanie rozszerzonego klienta transakcyjnego. Jeśli struktura xa_info zawiera jeden z tych parametrów, nazwa menedżera kolejek jest określona w parametrze *QMNAME* , o ile nie jest używany domyślny menedżer kolejek.
- Tylko jeden menedżer kolejek w danym momencie może uczestniczyć w transakcji koordynowanej przez instancję koordynatora zewnętrznego punktu synchronizacji. Koordynator punktu synchronizacji jest efektywnie połączony z menedżerem kolejek i podlega regułce, że obsługiwane jest tylko jedno połączenie w danym momencie.
- Wszystkie aplikacje, które zawierają wywołania do zewnętrznego koordynatora punktu synchronizacji, mogą łączyć się tylko z menedżerem kolejek, który uczestniczy w transakcji zarządzanej przez zewnętrznego koordynatora (ponieważ są one już w rzeczywistości połączone z tym menedżerem kolejek). Jednak takie aplikacje muszą wywoływać wywołanie `MQCONN` w celu uzyskania uchwytu połączenia, a wywołanie `MQDISC` przed wyjściem.
- Menedżer kolejek z aktualizacjami zasobów koordynowanymi przez koordynatora zewnętrznego punktu synchronizacji musi rozpoczynać się przed koordynatorem zewnętrznego punktu synchronizacji. Podobnie, koordynator punktu synchronizacji musi kończyć się przed menedżerem kolejek.
- Jeśli koordynator zewnętrznego punktu synchronizacji zakończy działanie w sposób nieprawidłowy, zatrzymaj i zrestartuj menedżer kolejek ***przed*** restartując koordynatora punktów synchronizacji, aby upewnić się, że wszystkie operacje przesyłania komunikatów niezatwierdzone w momencie niepowodzenia są prawidłowo rozstrzygnięte.

Korzystanie z programu CICS

CICS jest jednym z elementów programu TXSeries.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Wersje programu TXSeries , które są zgodne z interfejsem XA (i używają procesu zatwierdzania dwufazowego) są zdefiniowane pod adresem: IBM WebSphere MQ detailed system requirements

Produkt WebSphere MQ obsługuje również inne menedżery transakcji. Aktualne listy obsługiwanego oprogramowania znajdują się w sekcji IBM WebSphere MQ detailed system requirements .

Wymagania dotyczące procesu zatwierdzania dwufazowego

Wymagania procesu zatwierdzania dwufazowego w przypadku korzystania z procesu zatwierdzania dwufazowego CICS za pomocą produktu WebSphere MQ. Te wymagania nie mają zastosowania do produktu z/OS.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Należy zwrócić uwagę na następujące wymagania:

- Produkty WebSphere MQ i CICS muszą znajdować się na tym samym komputerze fizycznym.
- Produkt WebSphere MQ nie obsługuje programu CICS na kliencie MQI produktu WebSphere MQ .
- Należy uruchomić menedżer kolejek, podając jego nazwę określoną w sekcji definicji zasobu XAD, ***przed*** , aby uruchomić program CICS. Niezastosowanie się do tej opcji uniemożliwi uruchomienie programu CICS , jeśli dodano sekcję definicji zasobu XAD dla produktu WebSphere MQ do regionu CICS .
- Tylko jeden menedżer kolejek produktu WebSphere MQ może być dostępny w danym momencie z jednego regionu CICS .
- Transakcja CICS musi wydać żądanie `MQCONN` , zanim będzie można uzyskać dostęp do zasobów WebSphere MQ . Wywołanie `MQCONN` musi określać nazwę menedżera kolejek produktu WebSphere MQ określonego w pozycji XAOpen w sekcji definicji zasobu XAD dla regionu CICS . Jeśli ta pozycja jest pusta, żądanie `MQCONN` musi określać domyślny menedżer kolejek.
- Transakcja CICS , która uzyskuje dostęp do zasobów WebSphere MQ , musi wywołać wywołanie `MQDISC` z transakcji przed powrotem do programu CICS. W przypadku niepowodzenia może to oznaczać, że serwer aplikacji CICS jest nadal połączony, pozostawiając otwarte kolejki. Ponadto, jeśli nie zostanie zainstalowane wyjście zakończenia zadania (patrz “Przykładowe wyjście zakończenia zadania” na stronie 77), serwer aplikacji CICS może później zakończyć się nieprawidłowo, być może podczas kolejnej transakcji.
- Należy upewnić się, że identyfikator użytkownika CICS (cics) należy do grupy mqm, tak aby kod CICS miał uprawnienia do wywoływania produktu WebSphere MQ.

  W przypadku transakcji działających w środowisku CICS menedżer kolejek adaptuje swoje metody autoryzacji i określania kontekstu w następujący sposób:

  – Menedżer kolejek wysyła zapytanie do identyfikatora użytkownika, w ramach którego program CICS uruchamia transakcję. Jest to identyfikator użytkownika sprawdzany przez menedżera uprawnień do obiektów i jest używany do informacji kontekstowych.
  – W kontekście komunikatu typem aplikacji jest MQAT_CICS.
  – Nazwa aplikacji w kontekście jest kopiowana z nazwy transakcji CICS .

Ogólne wsparcie XA
Ogólny interfejs XA nie jest obsługiwany w produkcie IBM i. Dostępny jest moduł ładowalny z przełącznikiem XA, który umożliwia połączenie programu CICS z produktem WebSphere MQ w systemach UNIX and Linux . Dodatkowo udostępniono przykładowe pliki kodu źródłowego, które umożliwią tworzenie przełączników XA dla innych komunikatów transakcji.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Dostępne są następujące nazwy modułów ładowania przełącznika:

Tabela 11. Podstawowy kod dla aplikacji CICS : procedura inicjowania interfejsu XA

C (źródło)	**C (exec)-dodaj jeden z następujących elementów do XAD.Stanza**
amqzscix.c	amqzsc- TXSeries dla AIX, Wersja 5.1, amqzsc- TXSeries for HP-UX, wersja 5.1 amqzsc- TXSeries for Sun Solaris, wersja 5.1

Tabela 11. Podstawowy kod dla aplikacji CICS : procedura inicjowania interfejsu XA (kontynuacja)	
C (źródło)	**C (exec)-dodaj jeden z następujących elementów do XAD.Stanza**
amqzscin.c	mqmc4swi - TXSeries dla Windows, wersja 5.1

Budowanie bibliotek do użytku z programem TXSeries for Multiplatforms
Te informacje są używane podczas budowania bibliotek do użytku z programem TXSeries for Multiplatforms.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Wstępnie zbudowane pliki ładowania przełączników są bibliotekami współużytkowanymi (o nazwie *DLLs* w systemie Windows), które mogą być używane z programami CICS, które wymagają transakcji zatwierdzania dwufazowego przy użyciu protokołu XA. Nazwy tych wstępnie zbudowanych bibliotek znajdują się w tabeli Essential code for CICS applications: XA initialization routine(Kod Essential dla aplikacji CICS: procedura inicjowania interfejsu XA). Przykładowy kod źródłowy jest również dostarczany w następujących katalogach:

Tabela 12. Katalogi instalacyjne w systemach operacyjnych Windows, UNIX and Linux		
Platforma	**Katalog**	**Plik źródłowy**
UNIX and Linux	*MQ_INSTALLATION_PATH*/ samp/	amqzscix.c
Windows	*MQ_INSTALLATION_PATH*\Tools \c \ Przykłady	amqzscin.c

gdzie*MQ_INSTALLATION_PATH* to katalog, w którym zainstalowano produkt IBM WebSphere MQ.

Aby zbudować plik ładowania przełącznika z przykładowego źródła, należy postępować zgodnie z instrukcjami właściwymi dla danego systemu operacyjnego:

AIX
Wydaj następującą komendę:

```
export MQM_HOME=/usr/mqm
echo "amqzscix" > tmp.exp
xlc_r $MQM_HOME/samp/amqzscix.c -I/usr/lpp/cics/include -I$MQM_HOME/inc -e amqzscix -bE:tmp.exp
-bM:SRE -o amqzsc /usr/lpp/cics/lib/regxa_swxa.o -L$MQM_HOME/lib -L/usr/lpp/cics/lib -lcicsrt -lEncina
-lEncServer -lpthreads -lsarpc -lmqmcics_r -lmqmxa_r -lmqzi_r -lmqmcs_r
rm tmp.exp
```

Solaris
Wydaj następującą komendę:

```
/opt/SUNWspro/bin/cc -s -l/opt/encina/include amqzscix.c -G -o amqzscix  -e
    CICS_XA_Init -LMQ_INSTALLATION_PATH/lib -L/opt/encina/lib
    -L/opt/dcelocal/lib /opt/cics/lib/reqxa_swxa.o
    -lmqmcics -lmqmxa -lmqzi -lmqmcs -lmqmzse -lcicsrt -lEncina -lEncSfs -ldce
```

HP-UX
Wydaj następującą komendę:

```
cc -c -s -I/opt/encina/include MQ_INSTALLATION_PATH/samp/amqzscix.c -Aa +z -o  amqzscix.o ld -b
    -o amqzscix amqzscix.o /opt/cics/lib/regxa_swxa.o +e CICS_XA_Init \
    -LMQ_INSTALLATION_PATH/lib -L/opt/encina/lib -L/opt/cics/lib
    -lmqmxa_r -lmqzi_r -lmqmcs_r -lmqmzse -ldbm -lc -lm
```

Platformy Linux

Wydaj następującą komendę:

```
gcc -m32 -shared -fPIC -o amqzscix amqzscix.c
    \-IMQ_INSTALLATION_PATH/inc -I CICS_INSTALLATION_PATH/include
    \-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
    \-Wl,-rpath=/usr/lib -Wl,-rpath-link,/usr/lib -Wl,--no-undefined
    -Wl,--allow-shlib-undefined \-L CICS_LIB_PATH/regxa_swxa.o \-lpthread -ldl -lc
    -shared -lmqzi_r -lmqmxa_r -lmqmcics_r -ldl -lc
```

Windows

Wykonaj następujące kroki:

1. Użyj komendy cl, aby zbudować plik amqzscin.obj , kompilując co najmniej następujące zmienne:

   ```
   cl.exe -c -IEncinaPath\include -IMQ_INSTALLATION_PATH\include -Gz -LD amqzscin.c
   ```

2. Utwórz plik definicji modułu o nazwie `mqmc1415.def`, który zawiera następujące wiersze:

   ```
   LIBRARY MQMC4SWI
   EXPORTS
   CICS_XA_Init
   ```

3. Użyj komendy **lib** , aby zbudować plik eksportu i bibliotekę importu przy użyciu co najmniej następującej opcji:

   ```
   lib -def:mqmc4swi.def -out:mqmc4swi.lib
   ```

   Jeśli komenda lib powiedzie się, tworzony jest również plik `mqmc4swi.exp` .

4. Użyj komendy link do zbudowania `mqmc4swi.dll` przy użyciu co najmniej następującej opcji:

   ```
   link.exe -dll -nod -out:mqmc4swi.dll
       amqzscin.obj CicsPath\lib\regxa_swxa.obj
       mqmc4swi.exp mqmcics4.lib
      CicsPath\lib\libcicsrt.lib
      DcePath\lib\libdce.lib DcePath\lib\pthreads.lib
      EncinaPath\lib\libEncina.lib
      EncinaPath\lib\libEncServer.lib
      msvcrt.lib kernel32.lib
   ```

Obsługa interfejsu XA produktu IBM WebSphere MQ i Tuxedo

IBM WebSphere MQ w systemie Windows, systemyUNIX and Linux mogą blokować aplikacje typu Tuxedo-koordynowane przez interfejs XA na czas nieokreślony w xa_start.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Może się to zdarzyć tylko wtedy, gdy dwa lub więcej procesów koordynowanych przez Tuxedo w ramach pojedynczej transakcji globalnej próbuje uzyskać dostęp do produktu IBM WebSphere MQ przy użyciu tego samego identyfikatora gałęzi transakcji (XID). Jeśli program Tuxedo daje każdemu procesowi w transakcji globalnej inny identyfikator XID, który ma być używany z produktem IBM WebSphere MQ, nie może to nastąpić.

Aby uniknąć tego problemu, należy skonfigurować każdą aplikację w Tuxedo, która uzyskuje dostęp do produktu IBM WebSphere MQ w ramach jednego identyfikatora transakcji globalnej (gtrid), w obrębie własnej grupy serwerów Tuxedo. Procesy z tej samej grupy serwerów używają tego samego identyfikatora XID podczas uzyskiwania dostępu do menedżerów zasobów w imieniu pojedynczego gtrid, a zatem są podatne na blokowanie w xa_start w produkcie IBM WebSphere MQ. Procesy w różnych grupach serwerów używają osobnych identyfikatorów XID podczas uzyskiwania dostępu do menedżerów zasobów i dlatego nie muszą być przekształcane do postaci szeregowej swojej pracy transakcji w produkcie IBM WebSphere MQ.

Włączanie procesu zatwierdzania dwufazowego CICS
Aby umożliwić CICS korzystanie z procesu zatwierdzania dwufazowego w celu koordynowania transakcji, które obejmują wywołania MQI, należy dodać wpis sekcji definicji zasobu CICS XAD do regionu CICS . Uwaga: ten temat nie ma zastosowania do z/OS.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Poniżej znajduje się przykład dodania pozycji z sekcji XAD dla produktu WebSphere MQ for Windows, gdzie `<Drive>` oznacza dysk, na którym zainstalowano produkt WebSphere MQ (na przykład D:).

```
cicsadd -cxad -r<cics_region> \
     ResourceDescription="MQM XA Product Description" \
     SwitchLoadFile="<Drive>:\Program Files\IBM\WebSphere MQ\bin\mqmc4swi.dll" \
     XAOpen=<queue_manager_name>
```

W przypadku rozszerzonych klientów transakcyjnych należy użyć pliku ładowania przełącznika mqcc4swi.dll.

Poniżej przedstawiono przykład dodania pozycji sekcji XAD dla produktu WebSphere MQ for UNIX and Linux systems, gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowano produkt WebSphere MQ :

```
cicsadd -cxad -r<cics_region> \
      ResourceDescription="MQM XA Product Description" \
      SwitchLoadFile="MQ_INSTALLATION_PATH/lib/amqzsc" \
      XAOpen=<queue_manager_name>
```

W przypadku rozszerzonych klientów transakcyjnych należy użyć pliku ładowania amqczsc.

Informacje na temat korzystania z komendy **cicsadd** zawiera podręcznik *CICS Administration Reference*lub *CICS Administration Guide* dla używanej platformy.

Wywołania produktu WebSphere MQ można włączyć do transakcji CICS , a zasoby produktu WebSphere MQ zostaną zatwierdzone lub wycofane zgodnie z zaleceniami CICS. Obsługa ta nie jest dostępna dla aplikacji klienckich.

Użytkownik ***musi*** wydać `MQCONN` z transakcji CICS w celu uzyskania dostępu do zasobów WebSphere MQ , po którym następuje odpowiedni `MQDISC` przy wyjściu.

Włączanie wyjść użytkownika CICS
Wyjście użytkownika CICS *punkt* (zwykle określane jako *wyjście użytkownika*) jest miejscem w module CICS , w którym program CICS może przesyłać sterowanie do programu, który został napisany (program użytkownika obsługi wyjścia *program*) i w którym program CICS może wznowić kontrolę, gdy program obsługi wyjścia zakończy pracę.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Przed użyciem programu użytkownika CICS należy zapoznać się z podręcznikiem *CICS Administration Guide* (Podręcznik administrowania programem CICS) dla używanej platformy.

Przykładowe wyjście zakończenia zadania
Produkt WebSphere MQ dostarcza przykładowy kod źródłowy dla wyjścia zakończenia zadania CICS .

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Przykładowy kod źródłowy znajduje się w następujących katalogach:

Tabela 13. Wyjścia zakończenia zadania CICS		
Platforma	**Katalog**	**Plik źródłowy**
Systemy UNIX and Linux	*MQ_INSTALLATION_PATH*/samp	amqzscgx.c
Windows	*MQ_INSTALLATION_PATH*\Tools \c \ Przykłady	amqzscgn.c

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

Instrukcje budowania dla przykładowego wyjścia zakończenia zadania są zawarte w komentarzach znajdujących się w górnej części każdego pliku źródłowego.

To wyjście jest wywoływane przez program CICS podczas normalnego i nieprawidłowego zakończenia zadania (po każdym punkcie synchronizacji został on zabrany). W programie obsługi wyjścia nie są dozwolone żadne prace naprawialne.

Te funkcje są używane tylko w kontekście produktu WebSphere MQ i CICS , w którym wersja CICS obsługuje interfejs XA. CICS odnosi się do tych bibliotek jako "programów" lub "wyjść użytkownika".

W systemie CICS istnieje wiele wyjść użytkownika, a amqzscgx, jeśli jest używany, jest zdefiniowany i włączony w systemie CICS jako "wyjście użytkownika zakończenia zadania (UE014015)", to znaczy wyjście nr 15.

Gdy wyjście zakończenia zadania jest wywoływane przez program CICS, program CICS poinformował już WebSphere MQ o stanie zakończenia zadania, a produkt WebSphere MQ podjął odpowiednie działanie (zatwierdź lub wycofaj zmiany). Wszystkie wyjścia mają na celu wydanie komendy MQDISC w celu wyczyszczenia.

Jednym z celów instalowania i konfigurowania systemu CICS w celu korzystania z wyjścia zakończenia zadania jest ochrona systemu przed niektórymi konsekwencjami błędnego kodu aplikacji. Na przykład, jeśli transakcja CICS zakończy się nieprawidłowo, nie wywołując najpierw MQDISC i nie ma zainstalowanego wyjścia zakończenia zadania, to w ciągu około 10 sekund może zostać wyświetlony kolejny nienaprawialny błąd regionu CICS . Wynika to z faktu, że wątek poprawności działania produktu WebSphere MQ, który jest uruchamiany w procesie cicsas, nie został opublikowany ani nie zostanie podany w celu czyszczenia i powrotu. Może się to okazać, że proces cicsas kończy się natychmiast, po zapisaniu raportów FFST w katalogu /var/mqm/errors lub w równoważnej lokalizacji w systemie Windows.

Korzystanie z serwera Microsoft Transaction Server (COM +)

COM + (Microsoft Transaction Server) został zaprojektowany w celu ułatwienia użytkownikom uruchamiania aplikacji logiki biznesowej na typowym serwerze warstwy pośredniej.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Informacje na temat ważnych informacji można znaleźć w sekcji Funkcje, które mogą być używane tylko z instalacją podstawową w systemie Windows .

COM + dzieli pracę na *działania*, które są zwykle krótkimi niezależnymi porcjami logiki biznesowej, takimi jak *transfer funduszy z konta A na konto B*. COM + w dużej mierze opiera się na orientacji obiektu, a w szczególności na COM; luźno jest to działanie COM + reprezentowane przez obiekt COM (business).

COM + jest zintegrowaną częścią systemu operacyjnego. Aby można było używać COM + w systemach Windows 2000 i Windows XP, należy użyć poprawki Hotfix Q313582 (znanej również jako pakiet COM + Rollup Package 19.1).

COM + udostępnia trzy usługi administratorowi obiektu biznesowego, usuwając wiele zmartwień z programisty obiektów biznesowych:

- Zarządzanie transakcjami
- Zabezpieczenia

- Zestawianie zasobów

Zwykle używany jest COM + z kodem frontowym, który jest klientem COM do obiektów znajdujących się w COM +, a także usług zaplecza, takich jak baza danych, z łączeniem WebSphere MQ między obiektem biznesowym COM + i zaplecza.

Kodem frontowym może być program autonomiczny lub aktywna strona serwera (Active Server Page-ASP) obsługiwana przez serwer Microsoft Internet Information Server (IIS). Kod frontowy może być na tym samym komputerze co COM + i jego obiekty biznesowe, z połączeniem przez COM. Ewentualnie kod frontowy może znajdować się na innym komputerze, z połączeniem za pośrednictwem DCOM. Różnych klientów można używać do uzyskiwania dostępu do tego samego obiektu biznesowego COM + w różnych sytuacjach.

Kod zaplecza może znajdować się na tym samym komputerze, co COM + i jego obiekty biznesowe, lub na innym komputerze z połączeniem za pośrednictwem dowolnego z obsługiwanych protokołów WebSphere MQ .

Wygasanie globalnych jednostek pracy

Menedżer kolejek może być skonfigurowany w taki sposób, aby wygasał globalne jednostki pracy po wstępnie skonfigurowanym okresie nieaktywności.

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Aby włączyć to zachowanie, należy ustawić następujące zmienne środowiskowe:

- *AMQ_TRANSACTION_EXPIRY_RESCAN*= odstęp czasu ponawiania w milisekundach >
- *AMQ_XA_TRANSACTION_WAŻNOŚCI*= < limit czasu oczekiwania w milisekundach >

Ostrzeżenie: Zmienne środowiskowe mają wpływ tylko na transakcje, które znajdują się w stanie *Idle* (Bezczynny) w tabeli 6-4 specyfikacji XA. Oznacza to, że transakcje, które nie są powiązane z żadnym wątkiem aplikacji, ale dla których oprogramowanie zewnętrznego menedżera transakcji nie zostało jeszcze wywołane wywołaniem funkcji **xa_prepare** .

Zewnętrzne menedżery transakcji przechowują tylko dziennik transakcji, które zostały przygotowane, zatwierdzone lub wycofane. Jeśli zewnętrzny menedżer transakcji jest wyłączony z jakiegokolwiek powodu, na jego zwracanym przez niego zwrocie dyski są przygotowywane, zatwierdzane i wycofywane, ale wszystkie aktywne transakcje, które jeszcze zostały przygotowane, stają się osierocone. Aby tego uniknąć, należy ustawić wartość *AMQ_XA_TRANSACTION_EXPIRY* , aby umożliwić oczekiwanym odstępom między aplikacją wywołując wywołania interfejsu API MQI i zakończeniem transakcji, po przeprowadzeniu operacji transakcyjnych na innych menedżerach zasobów.

Aby zapewnić terminową procedurę czyszczącą po utracie ważności *AMQ_XA_TRANSACTION_WA* , ustaw wartość *AMQ_TRANSACTION_EXPIRY_RESCAN* na wartość niższą niż wartość *AMQ_XA_TRANSACTION_TERMIN* , najlepiej, aby rescan występuje więcej niż jeden raz w przedziale *AMQ_XA_TRANSACTION_WAŻNOŚCI* .

Dyspozycja jednostki odzyskiwania

Produkt WebSphere MQ for z/OS udostępnia jednostki rozdysponowania odzyskiwania. Ta opcja umożliwia skonfigurowanie, czy druga faza dwufazowych transakcji zatwierdzania może być sterowana, na przykład podczas odtwarzania, po nawiązaniu połączenia z innym menedżerem kolejek w tej samej grupie współużytkowania kolejki (QSG).

Uwaga: Ten temat jest również dostępny w produkcie IBM MQ Version 8.0 i nowszych wersjach. Nie można jednak przełączyć się do nowszej wersji, korzystając z pola listy "Zmień wersję" . Aby przejść do tematu w nowszej wersji, należy dokonać edycji numeru wersji w polu adresu URL w przeglądarce.

Produkt WebSphere MQ for z/OS V7.0.1 i nowszy obsługuje jednostkę dyspozycyjności odtwarzania.

Dyspozycja jednostki odzyskiwania

Jednostka rozporządzania odtwarzania jest powiązana z połączeniem aplikacji, a następnie z każdą rozpoczętą przez nią transakcjami. Istnieją dwie możliwe jednostki rozdysponowania.

- Jednostka grupowa dyspozycyjności odzyskiwania identyfikuje, że aplikacja transakcyjna jest logicznie połączona z grupą współużytkowania kolejki i nie ma powinowactwa do żadnego konkretnego menedżera kolejek. Wszystkie transakcje zatwierdzania dwufazowego, które zostały rozpoczęte, zakończone phase-1 procesu zatwierdzania, to znaczy są wątpliwe, mogą być zapytania i rozstrzygnięte, po połączeniu z dowolnym menedżerem kolejek w QSG. W scenariuszu odtwarzania oznacza to, że koordynator transakcji nie musi ponownie łączyć się z tym samym menedżerem kolejek, który może być niedostępny.
- Jednostka QMGR (jednostka odtwarzania) określa, że aplikacja ma bezpośrednie powinowactwo do menedżera kolejek, z którym jest połączona, a także wszystkie transakcje, które są uruchamiane, mają również takie dyspozycję.

  W scenariuszu odtwarzania koordynator transakcji musi ponownie nawiązać połączenie z tym samym menedżerem kolejek w celu sprawdzenia i rozstrzygnięcia transakcji wątpliwych, niezależnie od tego, czy menedżer kolejek należy do grupy współużytkowania kolejek.

Wybór języka programowania do użycia

W tej sekcji znajdują się informacje na temat języków programowania i środowisk obsługiwanych przez produkt IBM WebSphere MQ, a także niektóre zagadnienia dotyczące ich używania.

Produkt IBM WebSphere MQ zapewnia obsługę następujących języków proceduralnych programowania:

- C
- Visual Basic (tylko w systemach Windows)
- COBOL

Języki te korzystają z interfejsu kolejki komunikatów (MQI) w celu uzyskania dostępu do usług kolejkowania komunikatów. Więcej informacji na temat obsługi tych języków zawiera sekcja “Korzystanie z języków proceduralnych w produkcie WebSphere MQ” na stronie 80.

Produkt IBM WebSphere MQ zapewnia obsługę następujących elementów:

- .NET
- ActiveX
- C++
- Java
- JMS

W tych językach używany jest model obiektów produktu IBM WebSphere MQ , który udostępnia klasy udostępniające te same funkcje, co wywołania i struktury produktu WebSphere MQ , ale które są bardziej naturalnym sposobem programowania w środowisku obiektowym. Niektóre języki, w których używany jest model obiektów IBM WebSphere MQ , udostępniają dodatkowe funkcje, które nie są dostępne w interfejsie kolejki komunikatów (MQI). Więcej informacji na temat obsługi tych języków zawiera sekcja “Programowanie obiektowe w produkcie WebSphere MQ” na stronie 81.

Korzystanie z języków proceduralnych w produkcie WebSphere MQ

Szczegółowe informacje na temat pisania aplikacji w wybranym języku można znaleźć w poniższych linkach:

- “Kodowanie w języku C” na stronie 84
- “Kodowanie w języku Visual Basic” na stronie 89
- “Kodowanie w języku COBOL” na stronie 87

Przegląd interfejsu wywołań dla języków proceduralnych znajduje się w sekcji Opisy wywołań. Ten temat zawiera listę wywołań MQI, a każdy z wywołań pokazuje, jak należy zakodować wywołania w każdym z tych języków.

Produkt WebSphere MQ udostępnia pliki definicji danych, które ułatwiają pisanie aplikacji. Pełny opis znajduje się w sekcji “Pliki definicji danych produktu IBM WebSphere MQ” na stronie 82.

Jeśli możesz wybrać język, w którym chcesz zakodować programy, zastanów się nad maksymalną długością komunikatów, które będą przetwarzane przez programy. Jeśli programy będą przetwarzać tylko komunikaty o znanej maksymalnej długości, można je zakodować w dowolnym z obsługiwanych języków programowania. Jeśli jednak maksymalna długość komunikatów, które programy będą musiały być przetwarzane, nie będzie miała miejsca, język wybrany przez użytkownika będzie zależał od tego, czy używany jest program CICS, IMS, czy też aplikacja wsadowa:

IMS i wsadowe
Kod programów w języku C, PL/I lub asembler języka, aby korzystać z udogodnień tych języków oferują do uzyskania i zwolnienia arbitralne ilości pamięci. Alternatywnie można zakodować programy w języku COBOL, ale używać języków asemblera, języka PL/I lub C, aby uzyskać i zwolnić pamięć masową.

Program CICS
Kod programu w dowolnym języku obsługiwanym przez program CICS. Interfejs EXEC CICS udostępnia wywołania służące do zarządzania pamięcią, jeśli jest to konieczne.

Programowanie obiektowe w produkcie WebSphere MQ

Niektóre języki i środowiska programistyczne, które korzystają z modelu obiektów IBM WebSphere MQ , udostępniają dodatkowe funkcje, które nie są dostępne w interfejsie kolejki komunikatów (MQI). Szczegółowe informacje na temat klas, metod i właściwości udostępnianych przez model obiektu IBM WebSphere MQ zawiera sekcja “Model obiektu IBM WebSphere MQ” na stronie 89.

.NET
Informacje na temat kodowania programów .NET za pomocą klas WebSphere MQ .NET znajdują się w sekcji Używanie technologii .NET . Message Service Clients for C/C++ and .NET udostępniają aplikacyjny interfejs programistyczny (API) o nazwie XMS , który ma ten sam zestaw interfejsów co usługa Java Message Service (JMS). API.

C++
Produkt IBM WebSphere MQ udostępnia klasy języka C + + równoważne obiektom WebSphere MQ oraz niektóre dodatkowe klasy równoważne z typami danych tablicowych. Udostępnia ona wiele funkcji, które nie są dostępne w interfejsie MQI. Informacje Używanie języka C++ temat kodowania programów za pomocą modelu obiektów WebSphere MQ w języku C + +. Message Service Clients for C/C++ and .NET zawiera interfejs API o nazwie XMS , który zawiera ten sam zestaw interfejsów, co usługa Java Message Service (JMS). API.

Java
Informacje na temat kodowania programów za pomocą produktu WebSphere MQ Object Model w języku Java można znaleźć w sekcji Używanie języka Java . Informacje na temat różnic między klasami produktu IBM WebSphere MQ dla klas Java i IBM WebSphere MQ , które ułatwiają podjęcie decyzji o ich użyciu, można znaleźć w sekcji “Czy należy używać klas IBM WebSphere MQ dla klas Java lub IBM WebSphere MQ dla usługi JMS?” na stronie 91.

JMS
Produkt Websphere MQ udostępnia również klasy, które implementują specyfikację Java Message Service (JMS). Szczegółowe informacje na temat klas produktu Websphere MQ dla usługi JMS zawiera sekcja Używanie języka Java. Informacje na temat różnic między klasami produktu IBM WebSphere MQ dla języka Java i klas IBM WebSphere MQ , które ułatwiają podjęcie decyzji o ich użyciu, zawiera sekcja “Czy należy używać klas IBM WebSphere MQ dla klas Java lub IBM WebSphere MQ dla usługi JMS?” na stronie 91.

Message Service Clients for C/C++ and .NET udostępniają aplikacyjny interfejs programistyczny (API) o nazwie XMS , który ma ten sam zestaw interfejsów co usługa Java Message Service (JMS). API.

ActiveX

Produkt WebSphere MQ ActiveX jest powszechnie znany jako MQAX. Produkt MQAX jest częścią produktu WebSphere MQ for Windows.Support for ActiveX został ustabilizowany na poziomie produktu WebSphere MQ w wersji 6.0 . Aby wykorzystać funkcje wprowadzone w produkcie WebSphere MQ w wersji nowszej niż 6.0, należy rozważyć użycie platformy .NET. Informacje na temat kodowania programów za pomocą modelu obiektu WebSphere MQ (ActiveX) zawiera sekcja Używanie interfejsu Component Object Model Interface (WebSphere MQ Automation Classes for ActiveX) .

Pojęcia pokrewne

Przegląd techniczny

"Projektowanie aplikacji" na stronie 7

Produkt IBM WebSphere MQ udostępnia kilka sposobów tworzenia aplikacji w celu wysyłania i odbierania komunikatów, które są potrzebne do obsługi procesów biznesowych. Istnieje również możliwość tworzenia aplikacji do zarządzania menedżerami kolejek i powiązanymi zasobami.

"Pojęcia związane z projektowaniem aplikacji" na stronie 8

Do pisania aplikacji IBM WebSphere MQ można użyć wyboru języków proceduralnych lub obiektowych. Odsyłacze w tym temacie można znaleźć w informacjach dotyczących pojęć związanych z produktem IBM WebSphere MQ , które są przydatne dla programistów aplikacji.

Odsyłacze pokrewne

Informacje dodatkowe dotyczące programowania aplikacji

Pliki definicji danych produktu IBM WebSphere MQ

Produkt IBM WebSphere MQ udostępnia pliki definicji danych, które ułatwiają pisanie aplikacji.

Pliki definicji danych są również znane jako:

Język	Definicje danych
C	Włącz pliki lub pliki nagłówkowe
Visual Basic	Pliki modułów (tylko wersje 32-bitowe)
COBOL	Kopiuj pliki
Asembler	Makra
PL/I	Pliki INCLUDE

Pliki definicji danych ułatwiające pisanie wyjść kanału są opisane w sekcji WebSphere MQ COPY, header, include, and module files.

Pliki definicji danych, które ułatwiają pisanie programów zewnętrznych, są opisane w sekcji "Procedury zewnętrzne, wyjścia interfejsu API i usługi instalacyjne produktu WebSphere MQ" na stronie 383.

Pliki definicji danych obsługiwane w języku C + + znajdują się w sekcji Używanie języka C++.

Nazwy plików definicji danych mają przedrostek CMQ, a także przyrostek określony przez język programowania:

Przyrostek	Język
a	Język asembler
b	Visual Basic
c	C
l	COBOL (bez wartości inicjalizowanych)
p	PL/I
v	COBOL (z ustawionym domyślnym zestawem wartości)

Biblioteka instalacji

Nazwa **thlqual** jest kwalifikatorem wysokiego poziomu dla biblioteki instalacji w systemie z/OS.

W tym temacie przedstawiono opis plików definicji danych produktu WebSphere MQ w następujących pozycjach:

- “Pliki włączanych języków C” na stronie 83
- “Pliki modułu Visual Basic” na stronie 83
- “Pliki kopii COBOL” na stronie 83

Pliki włączanych języków C

Pliki włączanych produktów WebSphere MQ C są wymienione w sekcji Pliki nagłówkowe języka C. Są one instalowane w następujących katalogach lub bibliotekach:

Platforma	Katalog instalacyjny lub biblioteka
Platformy UNIX	*MQ_INSTALLATION_PATH*/inc/
Systemy Windows	*MQ_INSTALLATION_PATH*\Tools\c\include

gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowano produkt WebSphere MQ .

Uwaga: W przypadku platform UNIX pliki włączanych plików są dowiązane symbolicznie do produktu `/usr/include`.

Więcej informacji na temat struktury katalogów zawiera sekcja Planowanie obsługi systemu plików .

Pliki modułu Visual Basic

Produkt WebSphere MQ for Windows udostępnia cztery pliki modułu Visual Basic.

Są one wymienione w sekcji Pliki modułu Visual Basic i są zainstalowane w

```
MQ_INSTALLATION_PATH\Tools\Samples\VB\Include
```

Pliki kopii COBOL

W przypadku języka COBOL produkt WebSphere MQ udostępnia oddzielne pliki kopii zawierające stałe nazwane oraz dwa pliki kopii dla każdej z tych struktur.

Istnieją dwa pliki kopii dla każdej struktury, ponieważ każda z nich jest udostępniana zarówno z wartościami początkownymi, jak i bez nich:

- W sekcji WORKING-STORAGE SECTION w programie w języku COBOL należy użyć plików, które inicjują pola struktury, do wartości domyślnych. Struktury te są zdefiniowane w plikach kopii, których nazwy są przyrostowe z literą V (wartości).
- W sekcji LINKAGE w programie w języku COBOL należy używać struktur bez wartości początkowych. Struktury te są definiowane w plikach kopii, których nazwy są przyrostowe z literą L (powiązanie).

Pliki kopii w języku COBOL produktu WebSphere MQ są wymienione w sekcji Pliki kopii COBOL. Są one instalowane w następujących katalogach:

Platforma	Katalog instalacyjny lub biblioteka
Inne platformy UNIX	*MQ_INSTALLATION_PATH*/inc/
Windows	*MQ_INSTALLATION_PATH*\Tools\cobol\copybook (for Micro Focus COBOL) *MQ_INSTALLATION_PATH*\Tools\cobol\copybook\VAcobol (w przypadku produktu IBM VisualAge COBOL)

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

Dołącz do programu tylko te pliki, które są potrzebne. Należy to zrobić z jedną lub większą liczbę instrukcji COPY po deklaracji level-01 . Oznacza to, że w razie potrzeby można dołączyć wiele wersji struktur w programie. Należy zauważyć, że CMQV jest dużym plikiem.

Poniżej przedstawiono przykład kodu COBOL w celu uwzględnienia pliku kopii CMQMDV:

```
01 MQM-MESSAGE-DESCRIPTOR.
   COPY CMQMDV.
```

Każda deklaracja struktury rozpoczyna się od elementu level-01 . Można zadeklarować kilka instancji struktury, kodując deklarację level-01 , po której następuje instrukcja COPY, która ma zostać skopiowana do pozostałej części deklaracji struktury. Aby odwołać się do odpowiedniej instancji, należy użyć słowa kluczowego IN.

Poniżej przedstawiono przykład kodu COBOL w celu uwzględnienia dwóch instancji CMQMDV:

```
* Declare two instances of MQMD
 01  MY-CMQMD.
     COPY CMQMDV.
 01  MY-OTHER-CMQMD.
     COPY CMQMDV.
*
* Set MSGTYPE field in MY-OTHER-CMQMD
     MOVE MQMT-REQUEST TO MQMD-MSGTYPE IN MY-OTHER-CMQMD.
```

Wyrównaj struktury w granicach 4-bajtowych. W przypadku użycia instrukcji COPY w celu uwzględnienia struktury następującej po elemencie, który nie jest elementem level-01 , należy upewnić się, że struktura jest wielokrotnością 4-bajtów od początku elementu level-01 . Jeśli tego nie zrobisz, możesz zmniejszyć wydajność aplikacji.

Struktury są opisane w sekcji Typy danych używane w MQI. Opisy pól w strukturach przedstawiają nazwy pól bez przedrostka. W programach w języku COBOL należy poprzedzić nazwy pól nazwą struktury, po której następuje łącznik, tak jak to pokazano w deklaracjach języka COBOL. Pola w plikach kopii struktury są wstępnie ustalone w ten sposób.

Nazwy pól w deklaracjach w plikach kopii struktury są pisane wielkimi literami. Zamiast tego można użyć małych liter lub małych liter. Na przykład pole *StrucId* struktury MQGMO jest wyświetlane jako MQGMO-STRUCID w deklaracji języka COBOL i w pliku kopii.

Struktury przyrostków V są deklarowane przy użyciu wartości początkowych dla wszystkich pól, dlatego należy ustawić tylko te pola, w których wymagana wartość jest inna od wartości początkowej.

Kodowanie w języku C

Zwróć uwagę na informacje w poniższych sekcjach podczas kodowania programów WebSphere MQ w języku C.

Parametry wywołań MQI

Parametry, które są *tylko wejściowe* i typu MQHCONN, MQHOBJ, MQHMSG lub MQLONG, są przekazywane przez wartość. Dla wszystkich pozostałych parametrów *adres* parametru jest przekazywany przez wartość.

Nie wszystkie parametry, które są przekazywane przez adres, muszą być określone za każdym razem, gdy wywoływana jest funkcja. Jeśli określony parametr nie jest wymagany, jako parametr w wywołaniu funkcji można określić wskaźnik pusty, w miejsce adresu danych parametrów. Parametry, dla których jest to możliwe, są identyfikowane w opisach wywołań.

Żaden parametr nie jest zwracany jako wartość funkcji; w terminologii C oznacza to, że wszystkie funkcje zwracają wartość void.

Atrybuty tej funkcji są definiowane przez zmienną makra MQENTRY. Wartość tej zmiennej makra zależy od środowiska.

Parametry z niezdefiniowanym typem danych

Dla funkcji MQGET, MQPUT i MQPUT1 każdy z nich ma parametr *Buffer* , który ma niezdefiniowany typ danych. Ten parametr jest używany do wysyłania i odbierania danych komunikatu aplikacji.

Parametry tego sortowania są przedstawione w przykładach C jako tablice MQBYTE. Parametry można deklarować w ten sposób, ale zwykle bardziej wygodne jest zadeklarowanie ich jako struktury opisowej układu danych w komunikacie. Parametr funkcji jest zadeklarowany jako wskaźnik-do-void, a więc adres dowolnych danych może być określony jako parametr w wywołaniu funkcji.

Typy danych

Wszystkie typy danych są definiowane za pomocą instrukcji `typedef` .

Dla każdego typu danych zdefiniowany jest również odpowiedni typ danych wskaźnika. Nazwa typu danych wskaźnika to nazwa podstawowego lub strukturalnego typu danych poprzedzona literą P w celu oznaczenia wskaźnika. Atrybuty wskaźnika są definiowane za pomocą zmiennej makra MQPOINTER. Wartość tej zmiennej makra zależy od środowiska. Poniższy kod ilustruje sposób deklarowania typów danych wskaźników:

```
#define MQPOINTER                   /* depends on environment */
...
typedef MQLONG  MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;     /* pointer to MQMD */
```

Manipulowanie łańcuchami binarnymi

Łańcuchy danych binarnych są deklarowane jako jeden z typów danych MQBYTEn.

Podczas kopiowania, porównywania lub ustawiania pól tego typu należy używać funkcji C `memcpy`, `memcmp`lub `memset`:

```
#include <string.h>
#include "cmqc.h"

MQMD MyMsgDesc;

memcpy(MyMsgDesc.MsgId,           /* set "MsgId" field to nulls    */
       MQMI_NONE,                 /* ...using named constant      */
       sizeof(MyMsgDesc.MsgId));

memset(MyMsgDesc.CorrelId,        /* set "CorrelId" field to nulls */
       0x00,                      /* ...using a different method   */
       sizeof(MQBYTE24));
```

Nie należy używać funkcji łańcuchowych strcpy, strcmp, strncpy lub strncmp, ponieważ te funkcje nie działają poprawnie z danymi zadeklarowani jako MQBYTE24.

Manipulowanie łańcuchami znaków

Gdy menedżer kolejek zwraca dane znakowe do aplikacji, menedżer kolejek zawsze dopełnia dane znakowe spacjami do zdefiniowanej długości pola. Menedżer kolejek nie zwraca łańcuchów zakończonych znakiem o kodzie zero, ale można użyć ich w danych wejściowych. Dlatego przy kopiowaniu, porównywaniu lub konkatenowaniu takich łańcuchów należy użyć funkcji łańcuchowych strncpy, strncmp lub strncat.

Nie należy używać funkcji łańcuchowych, które wymagają, aby łańcuch został zakończony znakiem o wartości NULL (strcpy, strcmp i strcat). Ponadto nie należy używać funkcji strlen w celu określenia długości łańcucha. Zamiast tego należy użyć funkcji sizeof, aby określić długość pola.

Wartości początkowe dla struktur

Plik włączany <cmqc.h> definiuje różne zmienne makra, których można użyć w celu udostępnienia początkowych wartości struktur podczas deklarowania instancji tych struktur. Te zmienne makra mają nazwy w postaci MQxxx_DEFAULT, gdzie MQxxx reprezentuje nazwę struktury. Użyj ich w następujący sposób:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

W przypadku niektórych pól znakowych MQI definiuje konkretne wartości, które są poprawne (na przykład w przypadku pól *StrucId* lub w polu *Format* w strukturze MQMD). Dla każdej z poprawnych wartości dostępne są dwie zmienne makra:

- Jedna zmienna makra definiuje wartość jako łańcuch o długości, z wyłączeniem niejawnej wartości NULL, która jest dokładnie zgodna z zdefiniowaną długością pola. Na przykład symbol ¬ reprezentuje pusty znak:

  ```
  #define MQMD_STRUC_ID "MD¬¬"
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Użyj tego formularza z funkcjami memcpy i memcmp.
- Inna zmienna makra definiuje wartość jako tablicę typu char; nazwa tej zmiennej makra to nazwa formularza łańcucha w postaci przyrostka _ARRAY. Na przykład:

  ```
  #define MQMD_STRUC_ID_ARRAY 'M','D','¬','¬'
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Użyj tego formularza, aby zainicjować pole, gdy instancja struktury jest zadeklarowana z wartościami innymi niż te, które są udostępniane przez zmienną makra MQMD_DEFAULT.

Wartości początkowe dla struktur dynamicznych

Jeśli wymagana jest zmienna liczba instancji struktury, instancje są zwykle tworzone w głównej pamięci masowej uzyskanej dynamicznie przy użyciu funkcji calloc lub malloc.

Aby zainicjować pola w takich strukturach, zaleca się następujące techniki:

1. Zadeklaruj instancję struktury, używając odpowiedniej zmiennej makra MQxxx_DEFAULT w celu zainicjowania struktury. Ta instancja staje się *modelem* dla innych instancji:

   ```
   MQMD ModelMsgDesc = {MQMD_DEFAULT};
                                    /* declare model instance */
   ```

   Kod statyczny lub automatyczny słów kluczowych w deklaracji, aby nadać modelowi instancję statyczną lub dynamiczną, zgodnie z wymaganiami.
2. Użyj funkcji calloc lub malloc, aby uzyskać pamięć masową dla dynamicznej instancji struktury:

```
PMQMD InstancePtr;
InstancePtr = malloc(sizeof(MQMD));
                                  /* get storage for dynamic instance */
```

3. Użyj funkcji memcpy, aby skopiować instancję modelu do instancji dynamicznej:

```
memcpy(InstancePtr,&ModelMsgDesc,sizeof(MQMD));
                                  /* initialize dynamic instance */
```

Użyj z języka C++

W przypadku języka programowania C++ pliki nagłówkowe zawierają następujące dodatkowe instrukcje, które są uwzględniane tylko wtedy, gdy używany jest kompilator C++:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Kodowanie w języku COBOL

Należy zwrócić uwagę na informacje zawarte w poniższej sekcji podczas kodowania programów WebSphere MQ w języku COBOL.

Stałe nazwane

Nazwy stałych są wyświetlane jako część nazwy, która zawiera znak podkreślenia (_). W języku COBOL należy użyć znaku łącznika (-) w miejsce podkreślenia. Stałe, które mają wartości łańcuchowe, używają jednego znaku cudzysłowu (') jako separatora łańcucha. Aby kompilator akceptował ten znak, należy użyć opcji APOST kompilatora.

Plik kopii CMQV zawiera deklaracje stałych nazwanych jako elementy level-10 . Aby użyć stałych, należy zadeklarować jawnie element level-01 , a następnie użyć instrukcji COPY do skopiowania w deklaracjach stałych:

```
 WORKING-STORAGE SECTION.
 01  MQM-CONSTANTS.
     COPY CMQV.
```

Jednak ta metoda powoduje, że stałe zajmą pamięć masową w programie, nawet jeśli nie są one przywołane. Jeśli stałe są zawarte w wielu oddzielnych programach w obrębie tej samej jednostki wykonywania, wówczas będzie istnieć wiele kopii stałych; może to spowodować użycie znacznej ilości pamięci głównej. Aby tego uniknąć, należy dodać klauzulę GLOBAL do deklaracji level-01 :

```
* Declare a global structure to hold the constants
 01  MQM-CONSTANTS GLOBAL.
     COPY CMQV.
```

To przydziela pamięć tylko dla *jednego* zestawu stałych w jednostce uruchamiania. Stałe mogą jednak być przywoływani przez *dowolny* program w obrębie jednostki uruchamiania, a nie tylko program, który zawiera deklarację level-01 .

Zapewnianie wyrównania struktury

Należy zwrócić uwagę na zapewnienie, że struktury IBM WebSphere MQ , które są przekazywane do uruchamiania w wywołaniu programu MQ , muszą być wyrównane do granic słów. Granica słowa to 4 bajty dla procesów 32-bitowych, 8 bajtów dla procesów 64-bitowych i 16 bajtów dla procesów 128-bitowych (IBM i).

Jeśli to możliwe, wszystkie struktury produktu IBM WebSphere MQ należy umieścić razem, tak aby były wyrównane do granicy.

Kodowanie w języku pTAL

Podczas kodowania programów IBM WebSphere MQ w języku pTALnależy zapoznać się z informacjami zawartymi w poniższej sekcji.

HP Integrity NonStop Server

Definiowanie i inicjowanie struktur IBM WebSphere MQ

Definicje struktur pTAL dla struktur IBM WebSphere MQ mają nazwy kończące się łańcuchem ^DEF. Na przykład następujące deklaracje pTAL zostaną zakodowane w celu utworzenia struktury deskryptora komunikatu IBM WebSphere MQ (MQMD) i struktury opcji umieszczania komunikatu IBM WebSphere MQ (MQPMO).

```
STRUCT MYMD(MQMD^DEF);        ! Declare an MQMD structure
STRUCT MYPMO(MQPMO^DEF);      ! Declare an MQPMO structure
```

IBM WebSphere MQ udostępnia komendę pTAL DEFINE z nazwami, które kończą się łańcuchem ^DEFAULT , aby zainicjować struktury IBM WebSphere MQ z wartościami domyślnymi. Następujące instrukcje pTAL są kodowane w celu przypisania wartości domyślnych do zadeklarowanych struktur MQMD i MQPMO:

```
MQMD^DEFAULT(MYMD);           ! Assign default values to an MQMD structure
MQPMO^DEFAULT(MYPMO);         ! Assign default values to an MQPMO structure
```

Za pomocą podobnego kodu można deklarować i inicjować inne struktury IBM WebSphere MQ .

pTAL i CRE

Programy pTAL nie mogą zainicjować środowiska Common Runtime Environment i dlatego muszą być używane z główną procedurą w języku C lub COBOL.

Przykłady pTAL dostarczane z produktem IBM WebSphere MQ używają procedury wiersza głównego języka C o nazwie AMQSPTM0.C

Parametry z typem danych MQCHAR

Procedury MQGET, MQPUT i MQPUT1 mają parametr **Buffer** , który ma typ danych MQCHAR .EXT . Ten parametr służy do wysyłania i odbierania danych komunikatu aplikacji.

Parametry tego sortowania są wyświetlane w przykładach pTAL jako tablice łańcuchów. W ten sposób można zadeklarować parametry, ale zwykle wygodniej jest zadeklarować je jako strukturę opisującą układ danych w komunikacie. Parametr procedury jest zadeklarowany jako MQCHAR .EXT, ale adres wszystkich danych można określić jako parametr w wywołaniu procedury.

Manipulowanie łańcuchami znaków

Gdy menedżer kolejek zwraca dane znakowe do aplikacji, dane znakowe są zawsze dopełniane odstępami do zdefiniowanej długości pola. Menedżer kolejek nie zwraca łańcuchów zakończonych znakiem o kodzie zero, ale można ich użyć w danych wejściowych.

Kodowanie w języku Visual Basic

Należy zwrócić uwagę na informacje zawarte w poniższej sekcji podczas kodowania programów WebSphere MQ w języku Visual Basic.

Uwaga: Poza środowiskiem .NET wsparcie dla Visual Basic (VB) w produkcie WebSphere MQ zostało ustabilizowane na poziomie V6.0 . Większość nowych funkcji dodanych do produktu WebSphere MQ 7.0 lub nowszego nie jest dostępna dla aplikacji VB. Jeśli programowanie jest używane w produkcie VB.NET, należy użyć klas .NET produktu WebSphere MQ . Więcej informacji na ten temat zawiera sekcja Używanie technologii .NET.

Visual Basic jest obsługiwany tylko w systemie Windows.

Aby uniknąć niezamierzonego tłumaczenia danych binarnych przechodzących między Visual Basic i WebSphere MQ, należy użyć definicji MQBYTE zamiast MQSTRING. CMQB.BAS definiuje kilka nowych typów MQBYTE, które są odpowiednikami definicji bajtu w języku C i używają tych typów w strukturach produktu WebSphere MQ . Na przykład dla struktury MQMD (deskryptor komunikatu) wartość MsgId (identyfikator komunikatu) jest zdefiniowana jako MQBYTE24.

Element Visual Basic nie ma typu danych wskaźnika, dlatego odwołania do innych struktur danych produktu WebSphere MQ są przez przesunięcie, a nie wskaźnik. Zadeklaruj strukturę złożoną składającą się z dwóch struktur komponentów i określ strukturę złożoną w wywołaniu. Obsługa produktu WebSphere MQ dla języka Visual Basic zapewnia wywołanie MQCONNXAny, dzięki którym możliwe jest, aby aplikacje klienckie określały właściwości kanału w połączeniu klienta. Akceptuje on strukturę o nietypowym typie (MQCNOCD) w miejsce typowej struktury MQCNO.

Struktura MQCNOCD to struktura złożona składająca się z obiektu MQCNO, po którym następuje zmaterializowana tabela zapytania (MQCD). Ta struktura jest zadeklarowana w pliku nagłówkowych wyjść CMQXB. Użyj podprogramu MQCNOCD_DEFAULTS, aby zainicjować strukturę MQCNOCD. Udostępniono przykładowy proces tworzenia wywołań MQCONNX (amqscnxb.vbp).

Parametr MQCONNXAny ma te same parametry co parametr MQCONNX, z tą różnicą, że parametr *ConnectOpts* jest zadeklarowany jako typ dowolnego typu danych, a nie typ danych MQCNO. Umożliwia to akceptowanie przez funkcję struktury MQCNO lub struktury MQCNOCD. Ta funkcja jest zadeklarowana w głównym pliku nagłówkowy CMQB.

Model obiektu IBM WebSphere MQ

Model obiektu IBM WebSphere MQ składa się z klas, metod i właściwości. Te informacje umożliwiają zapoznanie się z każdym z tych pojęć.

Model obiektu IBM WebSphere MQ składa się z następujących elementów:

- *Klasy* reprezentujące pojęcia związane z produktem WebSphere MQ , takie jak menedżery kolejek, kolejki i komunikaty.
- *Metody* dla każdej klasy odpowiadającej wywołaniach MQI.
- *Properties* (Właściwości) dla każdej klasy odpowiadającej atrybutom obiektów WebSphere MQ .

Podczas tworzenia aplikacji produktu WebSphere MQ przy użyciu modelu obiektu WebSphere MQ tworzone są instancje tych klas w programie. Instancja klasy w programowaniu obiektowym jest nazywana *obiektem*. Gdy obiekt został utworzony, użytkownik wchodzi w interakcję z obiektem przez sprawdzenie lub ustawienie wartości właściwości obiektu (odpowiednik wywołania wywołania MQINQ lub MQSET) oraz przez wywołanie metody w odniesieniu do obiektu (odpowiednik wydania innych wywołań MQI).

W tych tematach opisano szczegółowo każdy z modeli obiektów WebSphere MQ :

- "Klasy" na stronie 90
- "odwołania do obiektów" na stronie 90
- "Kody powrotu" na stronie 91

Klasy

Model obiektów WebSphere MQ udostępnia następujący podstawowy zestaw klas.

Rzeczywista implementacja modelu różni się nieznacznie w zależności od różnych obsługiwanych środowisk obiektowych.

MQQueueManager
Obiekt klasy MQQueueManager reprezentuje połączenie z menedżerem kolejek. Ma metody do łączenia (), Disconnect (), Commit () i Backout () (odpowiednik MQCONN lub MQCONNX, MQDISC, MQCMIT i MQBACK). Ma ona właściwości odpowiadające atrybutom menedżera kolejek. Uzyskiwanie dostępu do właściwości atrybutu menedżera kolejek niejawnie łączy się z menedżerem kolejek, jeśli nie jest jeszcze połączony. Zniszczenie obiektu MQQueueManager w sposób niejawny rozłącza się z menedżerem kolejek.

MQQUEUE
Obiekt klasy MQQueue reprezentuje kolejkę. Posiada metody umieszczania () i Get () komunikatów do i z kolejki (odpowiedniki MQPUT i MQGET). Ma właściwości odpowiadające atrybutom kolejki. Uzyskanie dostępu do właściwości atrybutu kolejki lub wywołanie metody Put () lub Get () powoduje niejawne otwarcie kolejki (odpowiednik komendy MQOPEN). Zniszczenie obiektu MQQueue powoduje niejawne zamknięcie kolejki (odpowiednik komendy MQCLOSE).

Temat MQTopic
Obiekt klasy MQTopic reprezentuje temat. Posiada metody umieszczania (publikowania) i Get () (odbieranie lub subskrybowanie) komunikatów do i z tematu (równoważnego wywołania MQPUT i MQGET). Ma właściwości odpowiadające atrybutom tematu. Dostęp do obiektu MQTopic można uzyskać tylko w przypadku publikacji lub subskrypcji, a nie obu jednocześnie. W przypadku użycia do odbierania komunikatów obiekt MQTopic może zostać utworzony przy użyciu niezarządzanej lub zarządzanej subskrypcji oraz jako trwały lub nietrwały subskrybent-dla tych różniących się scenariuszy udostępniono wiele przeciążonych konstruktorów.

Komunikat MQMessage
Obiekt klasy MQMessage reprezentuje komunikat, który ma zostać umieszczony w kolejce lub został zwrócony z kolejki. Zawiera on bufor i hermetykuje zarówno dane aplikacji, jak i MQMD. Ma on właściwości odpowiadające pola MQMD oraz metody umożliwiające zapisywanie i odczytywanie danych użytkownika różnych typów (na przykład łańcuchów, długich liczb całkowitych, krótkich liczb całkowitych, pojedynczych bajtów) do buforu i z niego.

Opcje MQPutMessage
Obiekt klasy opcji MQPutMessagereprezentuje strukturę MQPMO. Ma właściwości odpowiadające polom MQPMO.

Opcje MQGetMessage
Obiekt klasy MQGetMessageOptions reprezentuje strukturę MQGMO. Ma właściwości odpowiadające polu MQGMO.

Proces MQProcess
Obiekt klasy MQProcess reprezentuje definicję procesu (używaną z wyzwalaniem). Zawiera właściwości reprezentujące atrybuty definicji procesu.

MQDistributionList
Obiekt klasy MQDistributionList reprezentuje listę dystrybucyjną (używaną do wysyłania wielu komunikatów za pomocą jednej operacji MQPUT). Zawiera ona listę obiektów elementów MQDistributionList.

Element MQDistributionList
Obiekt klasy Element MQDistributionListreprezentuje miejsce docelowe pojedynczej listy dystrybucyjnej. Obudowuje struktury MQOR, MQRR i MQPMR i ma właściwości odpowiadające danym dziedzinom tych struktur.

odwołania do obiektów

W programie WebSphere MQ , który korzysta z interfejsu MQI, program WebSphere MQ zwraca uchwyty połączeń i uchwyty obiektów do programu.

Te uchwyty muszą być przekazywane jako parametry w kolejnych wywołaniach produktu WebSphere MQ . W przypadku modelu obiektu WebSphere MQ uchwyty te są ukryte w programie użytkowym. Zamiast tego tworzenie obiektu z klasy powoduje, że odwołanie do obiektu jest zwracane do programu aplikacji. Jest to odwołanie do obiektu, które jest używane podczas wykonywania wywołań metod i dostępu do właściwości dla obiektu.

Kody powrotu

Wywołanie metody wywołania metody lub ustawienie wartości właściwości powoduje ustawienie zwracanych kodów powrotu.

Te kody powrotu są kodem zakończenia i kodem przyczyny, a same są właściwościami obiektu. Wartości kodu zakończenia i kodu przyczyny są takie same, jak wartości zdefiniowane dla interfejsu MQI, z pewnymi dodatkowymi wartościami specyficznymi dla środowiska obiektowego.

Czy należy używać klas IBM WebSphere MQ dla klas Java lub IBM WebSphere MQ dla usługi JMS?

Aplikacja Java może korzystać z klas IBM WebSphere MQ dla klas Java lub IBM WebSphere MQ dla usługi JMS w celu uzyskania dostępu do zasobów produktu IBM WebSphere MQ . Każde podejście ma swoje zalety.

Klasy produktu IBM WebSphere MQ dla języka Java obudowują interfejs kolejki komunikatów (Message Queue Interface-MQI), rodzimy interfejs API produktu IBM WebSphere MQ i używają tego samego modelu obiektowego co inne interfejsy obiektowe, natomiast klasy produktu IBM WebSphere MQ dla usługi Java Message Service implementuje interfejsy Sun Java Message Service (JMS).

Jeśli użytkownik jest zaznajomiony z produktem IBM WebSphere MQ w środowiskach innych niż Java, korzystając z języków proceduralnych lub obiektowych, można przenieść istniejącą wiedzę do środowiska Java przy użyciu klas IBM WebSphere MQ dla języka Java. Można również wykorzystać pełny zakres funkcji produktu IBM WebSphere MQ, z których nie wszystkie są dostępne w klasach produktu IBM WebSphere MQ dla usługi JMS.

Jeśli użytkownik nie zna produktu IBM WebSphere MQlub ma już doświadczenie w usłudze JMS, może okazać się łatwiejszy w użyciu znanego interfejsu API JMS w celu uzyskania dostępu do zasobów produktu IBM WebSphere MQ przy użyciu klas IBM WebSphere MQ dla usługi JMS. Usługa JMS jest również integralną częścią platformy Java Platform, Enterprise Edition (Java EE). Aplikacje Java EE mogą używać komponentów bean sterowanych komunikatami (message-driven bean-MDB) do asynchronicznej przetwarzania komunikatów, a komponenty MDB mogą przetwarzać tylko komunikaty JMS. Usługa JMS jest również standardowym mechanizmem dla środowiska Java EE , który umożliwia interakcję z asynchronicznymi systemami przesyłania komunikatów, takimi jak IBM WebSphere MQ. Każdy serwer aplikacji zgodny ze specyfikacją Java EE musi zawierać dostawcę JMS, dzięki temu można używać JMS do komunikacji między różnymi serwerami aplikacji. Można również używać portu aplikacji z jednego dostawcy JMS do innego bez żadnych zmian w aplikacji.

Projektowanie aplikacji produktu IBM WebSphere MQ

Po zdecydowaniu się, w jaki sposób aplikacje mogą korzystać z platform i środowisk, które są dostępne dla użytkownika, należy zdecydować, w jaki sposób korzystać z funkcji oferowanych przez produkt WebSphere MQ.

Podczas projektowania aplikacji IBM WebSphere MQ należy wziąć pod uwagę następujące pytania i opcje:

Rodzaj wniosku

Jaki jest cel Twojej aplikacji? Poniżej znajdują się odsyłacze do informacji na temat różnych typów aplikacji, które można utworzyć:

- Serwer
- Klient
- Publikowanie/subskrypcja

- Usługi WWW
- Procedury zewnętrzne, wyjścia funkcji API i usługi instalowalne

Ponadto można również napisać własne aplikacje, aby zautomatyzować administrowanie produktem IBM WebSphere MQ. Więcej informacji na ten temat zawiera sekcja Wprowadzenie do interfejsu WebSphere MQ Administration Interface (MQAI) i Automating administration tasks (Automatyzacja zadań administracyjnych).

Język programowania
Produkt IBM WebSphere MQ obsługuje wiele języków programowania proceduralnego i obiektowego w celu pisania aplikacji. Więcej informacji na ten temat zawiera sekcja “Wybór języka programowania do użycia” na stronie 80.

Aplikacje dla więcej niż jednej platformy

Czy Twoja aplikacja będzie działać na więcej niż jednej platformie? Czy masz strategię, aby przenieść się na inną platformę z tej, której używacie dzisiaj? Jeśli odpowiedź na którekolwiek z tych pytań brzmi "tak", należy się upewnić, że kod został zakodowany w sposób niezależny od platformy.

Jeśli używany jest kod C, kod w standardzie ANSI C. Należy użyć standardowej funkcji biblioteki C, a nie równoważnej funkcji specyficznej dla platformy, nawet jeśli funkcja specyficzna dla platformy jest szybsza lub bardziej wydajna. Wyjątkiem jest sytuacja, w której efektywność w kodzie ma wartość paramount, gdy kod ma być używany w obu sytuacjach za pomocą programu #ifdef. Na przykład:

```
#ifdef _AIX
     AIX specific code
#else
     generic code
#endif
```

Typy kolejek
Czy chcesz utworzyć kolejkę za każdym razem, gdy jest ona potrzebna, czy też chcesz korzystać z kolejek, które już zostały skonfigurowane? Czy chcesz usunąć kolejkę po zakończeniu jej używania, czy też będzie ona używana ponownie? Czy chcesz użyć kolejek aliasowych do niezależności aplikacji? Aby sprawdzić, jakie typy kolejek są obsługiwane, należy zapoznać się z kolejkami .

Korzystanie z klastrów menedżera kolejek
Użytkownik może chcieć skorzystać z uproszczonej administracji systemu oraz zwiększyć dostępność, skalowalność i równoważenie obciążenia, które są możliwe w przypadku korzystania z klastrów. Więcej informacji na ten temat zawiera sekcja Klastry menedżera kolejek .

Typy komunikatów
Można użyć datagramów dla prostych wiadomości, ale komunikaty żądania (w przypadku których oczekują odpowiedzi) na inne sytuacje. Do niektórych wiadomości można przypisać różne priorytety. Aby uzyskać więcej informacji na temat projektowania komunikatów, zobacz: “Projektowanie wiadomości” na stronie 94.

Korzystanie z przesyłania komunikatów w trybie publikowania/subskrypcji lub punkt z punktem
Za pomocą przesyłania komunikatów w trybie publikowania/subskrypcji aplikacja wysyłający wysyła informacje, które chce współużytkować w komunikacie IBM WebSphere MQ , do standardowego miejsca docelowego zarządzanego przez produkt IBM WebSphere MQ publikowania? Subskrybuj i umożliwia IBM WebSphere MQ obsługę dystrybucji tych informacji. Aplikacja docelowa nie musi wiedzieć nic o źródle otrzymywanych informacji, po prostu rejestruje zainteresowanie w jednym lub wielu tematach i otrzymuje te informacje, gdy jest ona dostępna. Więcej informacji na temat przesyłania komunikatów w trybie publikowania/subskrypcji można znaleźć w sekcji Wprowadzenie do przesyłania komunikatów w trybie publikowania/subskrypcji produktu IBM WebSphere MQ .

Za pomocą przesyłania komunikatów w trybie punkt z punktem aplikacja wysyłający wysyła komunikat do określonej kolejki, skąd wie, że aplikacja odbierający ją pobierze. Aplikacja odbierający pobiera komunikaty z określonej kolejki i działa na ich zawartość. Aplikacja często będzie działać zarówno jako nadawca, jak i odbiorca, wysyłając zapytanie do innej aplikacji i otrzymując odpowiedź.

Sterowanie programami IBM WebSphere MQ

Niektóre programy mogą być uruchamiane automatycznie lub gdy programy oczekują na nadejścia określonego komunikatu w kolejce (za pomocą opcji IBM WebSphere MQ *wyzwalanie* patrz “Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy” na stronie 338). Alternatywnie można uruchomić inną instancję aplikacji, gdy komunikaty w kolejce nie są przetwarzane wystarczająco szybko (za pomocą opcji IBM WebSphere MQ *zdarzeń instrumentacji* zgodnie z opisem w sekcji Zdarzenia instrumentacji).

Uruchamianie aplikacji na kliencie IBM WebSphere MQ

Pełny interfejs MQI jest obsługiwany w środowisku klienta. Dzięki temu niemal każda aplikacja produktu IBM WebSphere MQ może zostać zwolniona w celu uruchomienia na kliencie MQI produktu IBM WebSphere MQ . Powiąż aplikację z klientem MQI produktu IBM WebSphere MQ z biblioteką MQIC, a nie z biblioteką MQI.

Uwaga: Aplikacja działająca na kliencie produktu IBM WebSphere MQ może jednocześnie łączyć się z więcej niż jednym menedżerem kolejek lub używać nazwy menedżera kolejek z gwiazdką (*) w wywołaniu MQCONN lub MQCONNX. Zmień aplikację, jeśli chcesz połączyć się z bibliotekami menedżera kolejek zamiast bibliotek klienta, ponieważ ta funkcja nie będzie dostępna.

Więcej informacji zawiera sekcja “Uruchamianie aplikacji w środowisku klienta MQI produktu IBM WebSphere MQ” na stronie 368.

Wydajność aplikacji

Decyzje projektowe mogą mieć wpływ na wydajność aplikacji, a sugestie dotyczące zwiększenia wydajności aplikacji produktu IBM WebSphere MQ można znaleźć w sekcji “Projektowanie i wydajność aplikacji” na stronie 95.

Zaawansowane techniki IBM WebSphere MQ

W przypadku bardziej zaawansowanych aplikacji można użyć zaawansowanych technik IBM WebSphere MQ , takich jak korelowanie odpowiedzi, a także generowanie i wysyłanie informacji o kontekście produktu IBM WebSphere MQ . Więcej informacji na ten tematzawiera sekcja “Zaawansowane techniki IBM WebSphere MQ” na stronie 96.

Zabezpieczanie danych i zachowywanie integralności

Można użyć informacji o kontekście, które są przekazywane z komunikatem do przetestowania, że komunikat został wysłany z akceptowalnego źródła. W celu zapewnienia spójności danych z innymi zasobami można użyć obiektów syncwskazujących udostępnianych przez produkt IBM WebSphere MQ lub system operacyjny (szczegółowe informacje na ten temat zawiera sekcja “Zatwierdzanie i wycofywanie jednostek pracy” na stronie 331). Istnieje możliwość użycia funkcji *persistence* komunikatów produktu IBM WebSphere MQ w celu zapewnienia dostarczania ważnych komunikatów.

Testowanie aplikacji produktu IBM WebSphere MQ

Środowisko programowania aplikacji dla programów IBM WebSphere MQ nie różni się od tego w przypadku innych aplikacji, dlatego można używać tych samych narzędzi programistycznych, jak również do narzędzi śledzenia produktu IBM WebSphere MQ .

Obsługa wyjątków i błędów

Należy rozważyć sposób przetwarzania komunikatów, których nie można dostarczyć, a także sposób rozwiązywania sytuacji błędów, które są zgłaszane przez menedżera kolejek. W przypadku niektórych raportów konieczne jest ustawienie opcji raportu w tabeli MQPUT.

Pojęcia pokrewne

Przegląd techniczny produktu IBM WebSphere MQ

“Pojęcia związane z projektowaniem aplikacji” na stronie 8
Do pisania aplikacji IBM WebSphere MQ można użyć wyboru języków proceduralnych lub obiektowych. Odsyłacze w tym temacie można znaleźć w informacjach dotyczących pojęć związanych z produktem IBM WebSphere MQ , które są przydatne dla programistów aplikacji.

“Pisanie aplikacji kolejkowania” na stronie 199
Ta sekcja zawiera informacje na temat pisania aplikacji kolejkowania, łączenia się i rozłączania z menedżerem kolejek, publikowania/subskrypcji oraz obiektów otwierających i zamykających.

“Pisanie aplikacji klienckich” na stronie 360

Co należy wiedzieć, aby pisać aplikacje klienckie w produkcie WebSphere MQ.

“Używanie środowiska .NET” na stronie 573
Klasy produktu WebSphere MQ dla środowiska .NET umożliwiają programowi napisanego w środowisku programowania .NET nawiązanie połączenia z produktem WebSphere MQ jako klienta MQI produktu WebSphere MQ lub nawiązanie bezpośredniego połączenia z serwerem WebSphere MQ .

“Używanie języka C++” na stronie 644
ProduktWebSphere MQ udostępnia klasy języka C++ odpowiadające obiektom WebSphere MQ oraz niektóre dodatkowe klasy równoważne z typami danych tablicowych. Udostępnia ona wiele funkcji, które nie są dostępne w interfejsie MQI.

“Korzystanie z klas produktu WebSphere MQ dla usługi JMS” na stronie 734
Klasy WebSphere MQ classes for Java Message Service (klasy WebSphere MQ classes for JMS) są dostawcą JMS dostarczonym z produktem WebSphere MQ. Oprócz implementowania interfejsów zdefiniowanych w pakiecie javax.jms , klasy WebSphere MQ classes for JMS udostępniają dwa zestawy rozszerzeń do interfejsu API JMS.

“Korzystanie z klas produktu WebSphere MQ dla języka Java” na stronie 669
Klasy produktu WebSphere MQ dla języka Java umożliwiają korzystanie z produktu WebSphere MQ w środowisku Java. Aplikacja Java może korzystać z klas produktu WebSphere MQ dla języka Java lub klas WebSphere MQ dla usługi JMS w celu uzyskania dostępu do zasobów produktu WebSphere MQ .

“Korzystanie z interfejsu modelu obiektu komponentu (klasy automatyzacji produktu WebSphere MQ dla elementu ActiveX)” na stronie 1062
Klasy automatyzacji produktu WebSphere MQ dla ActiveX (MQAX) to komponenty ActiveX , które udostępniają klasy, których można użyć w aplikacji w celu uzyskania dostępu do produktu WebSphere MQ.

Projektowanie wiadomości

Należy wziąć pod uwagę aspekty podane w tych informacjach, które ułatwiają projektowanie komunikatów.

Komunikat jest tworzony w przypadku użycia wywołania MQI w celu umieszczenia komunikatu w kolejce. Jako dane wejściowe wywołania należy podać pewne informacje sterujące w *deskryptorze komunikatu* (MQMD) oraz dane, które mają zostać wysłane do innego programu. Ale na etapie projektowania należy wziąć pod uwagę następujące kwestie, ponieważ wpływają one na sposób tworzenia wiadomości:

Typ komunikatu, który ma zostać użyty

Czy projektujesz prostą aplikację, w której możesz wysłać wiadomość, a następnie nie podejmować dalszych działań? A może prosisz się o odpowiedź na pytanie? Jeśli zadajesz pytanie, możesz dołączyć do deskryptora komunikatu nazwę kolejki, na której chcesz otrzymać odpowiedź.

Czy chcesz, aby komunikaty żądania i odpowiedzi były synchroniczne? Oznacza to, że dla odpowiedzi zostanie ustawiony czas oczekiwania na odpowiedź na żądanie, a jeśli odpowiedź nie zostanie odezona w tym okresie, zostanie ona traktowana jako błąd.

A może wolisz pracować asynchronicznie, tak aby Twoje procesy nie musiały zależeć od występowania określonych zdarzeń, takich jak wspólne sygnały czasowe?

Inną kwestią jest to, czy masz wszystkie wiadomości wewnątrz jednostki pracy.

Przypisywanie różnych priorytetów do komunikatów

Do każdego komunikatu można przypisać wartość priorytetu, a następnie zdefiniować kolejkę w taki sposób, aby zachowała ona swoje komunikaty w kolejności ich priorytetu. W takim przypadku, gdy inny program pobierze komunikat z kolejki, zawsze otrzymuje on komunikat o najwyższym priorytecie. Jeśli kolejka nie utrzymuje swoich komunikatów w kolejności priorytetów, program pobierający komunikaty z kolejki pobierze je w kolejności, w jakiej zostały dodane do kolejki.

Programy mogą także wybierać komunikaty przy użyciu identyfikatora przypisanego do menedżera kolejek, gdy komunikat został umieszczony w kolejce. Alternatywnie można wygenerować własne identyfikatory dla każdego z komunikatów.

Wpływ zrestartowania menedżera kolejek na komunikaty
Menedżer kolejek zachowuje wszystkie komunikaty trwałe, odtwarzając je w razie potrzeby z plików dziennika produktu WebSphere MQ po zrestartowaniu. Nietrwałe komunikaty i tymczasowe kolejki dynamiczne nie są zachowywane. Wszystkie komunikaty, które nie mają być usuwane, muszą być zdefiniowane jako trwałe po ich utworzeniu. Podczas pisania aplikacji dla produktu WebSphere MQ for Windows lub WebSphere MQ w systemach UNIX and Linux należy się upewnić, że wiadomo, jak system został skonfigurowany w odniesieniu do przydziału plików dziennika, aby zmniejszyć ryzyko zaprojektowania aplikacji, która będzie uruchamiana w limitach plików dziennika.

Udzielanie informacji o sobie na rzecz odbiorcy wiadomości
Zwykle menedżer kolejek ustawia ID użytkownika, ale odpowiednio autoryzowane aplikacje mogą również ustawić to pole, dzięki czemu można uwzględnić własny identyfikator użytkownika i inne informacje, które program odbierający może wykorzystać do celów księgowych lub zabezpieczeń.

Ilość kolejek odbiorczy
Jeśli może być konieczne umieszczenie komunikatu w kilku kolejkach, można użyć listy dystrybucyjnejlub opublikować go w temacie.

Projektowanie i wydajność aplikacji

Istnieje wiele sposobów, w jaki kiepski projekt programu może mieć wpływ na wydajność. Może to być trudne do wykrycia, ponieważ program może się wydawać, że działa dobrze, ale wpływa na wydajność innych zadań. W tym temacie opisano kilka problemów specyficznych dla programów składających się na wywołania WebSphere MQ .

Oto kilka pomysłów, które pomogą Ci zaprojektować wydajne aplikacje:

- Zaprojektuj aplikację w taki sposób, aby przetwarzanie było wykonywane równolegle z czasem myślenia użytkownika:
  - Wyświetl panel i pozwól użytkownikowi na rozpoczęcie wpisywania w czasie, gdy aplikacja nadal jest inicjowana.
  - Uzyskaj dane, które są potrzebne równolegle z różnych serwerów.
- Należy utrzymywać otwarte połączenia i kolejki, jeśli ich ponowne użycie zamiast wielokrotnego otwierania i zamykania, łączenia i rozłączania jest wielokrotnie otwierane.
- Jednak aplikacja serwera, która jest umieszczana tylko w jednym komunikacie, powinna używać parametru MQPUT1.
- Menedżery kolejek są zoptymalizowane pod względem wielkości komunikatów o wielkości od 4 kB do 100 kB. Bardzo duże komunikaty są nieefektywne; prawdopodobnie lepiej jest wysłać 100 komunikatów o wielkości 1 MB każdy, niż jeden komunikat o wielkości 100 MB. Bardzo małe komunikaty są również nieefektywne. Menedżer kolejek wykonuje tę samą ilość pracy dla komunikatu jednobajtowego, co w przypadku komunikatu o wielkości 4 kB.
- Komunikaty należy przechowywać w jednostce pracy, dzięki czemu mogą być zatwierdzane lub wycofane jednocześnie.
- Użyj opcji nietrwałej dla komunikatów, które nie muszą być odtwarzalne.
- Jeśli konieczne jest wysłanie komunikatu do kilku kolejek docelowych, należy rozważyć użycie listy dystrybucyjnej.

Wpływ długości komunikatu

Ilość danych w komunikacie może mieć wpływ na wydajność aplikacji, która przetwarza komunikat. Aby uzyskać najlepszą wydajność z aplikacji, należy wysłać tylko istotne dane w komunikacie. Na przykład, w żądaniu obciążenia rachunku bankowego, jedyną informacją, która może być przekazywana z klienta do aplikacji serwera jest numer konta i kwota obciążenia.

Wpływ trwałości komunikatów

Komunikaty trwałe są zwykle rejestrowane. Protokołowanie komunikatów zmniejsza wydajność aplikacji, dlatego należy używać trwałych komunikatów tylko dla istotnych danych. Jeśli dane w komunikacie mogą zostać usunięte, jeśli menedżer kolejek zostanie zatrzymany lub nie powiedzie się, należy użyć nietrwałego komunikatu.

Wyszukiwanie konkretnego komunikatu

Wywołanie MQGET zwykle pobiera pierwszy komunikat z kolejki. Jeśli w deskryptorze komunikatu używane są komunikaty i identyfikatory korelacji (*MsgId* i *CorrelId*) w celu określenia konkretnego komunikatu, menedżer kolejek musi przeszukać kolejkę do momentu znalezienia tego komunikatu. Użycie wywołania MQGET w ten sposób wpływa na wydajność aplikacji.

Kolejki zawierające komunikaty o różnych długościach

Jeśli aplikacja nie może używać komunikatów o stałej długości, powiększaj i zmniejszaj bufory dynamicznie, tak aby odpowiadała typowi wielkości komunikatu. Jeśli aplikacja zgłosi wywołanie MQGET, które nie powiedzie się, ponieważ bufor jest za mały, zwracana jest wielkość danych komunikatu. Dodaj kod do aplikacji w taki sposób, aby bufor został odpowiednio zmieniony, a wywołanie MQGET zostało ponownie wysłane.

Uwaga: Jeśli atrybut *MaxMsgLength* nie zostanie ustawiony jawnie, jego wartością domyślną jest 4 MB, co może być bardzo nieefektywne, jeśli jest to używane do wywierania wpływu na wielkość buforu aplikacji.

Częstotliwość punktów synchronizacji

Programy, które emitują bardzo dużą liczbę wywołań MQPUT lub MQGET w punkcie synchronizacji, bez ich zatwierdzania, mogą powodować problemy z wydajnością. Przydzielone kolejki mogą zapełniać komunikaty, które są obecnie niedostępne, podczas gdy inne zadania mogą oczekiwać na pobranie tych komunikatów. Ma to wpływ na pamięć masową, a także na wątki związane z zadaniami, które próbują uzyskać komunikaty.

Użycie wywołania MQPUT1

Wywołania MQPUT1 należy używać tylko wtedy, gdy istnieje pojedynczy komunikat, który ma zostać umieszczony w kolejce. Jeśli chcesz umieścić więcej niż jeden komunikat, użyj wywołania MQOPEN, po którym następuje seria wywołań MQPUT i pojedyncze wywołanie MQCLOSE.

Liczba używanych wątków

W przypadku produktu WebSphere MQ for Windowsaplikacja może wymagać dużej liczby wątków. Dla każdego procesu menedżera kolejek przydzielana jest maksymalna dozwolona liczba wątków aplikacji.

Aplikacje mogą używać zbyt wielu wątków. Zastanów się, czy aplikacja bierze pod uwagę tę możliwość i że podejmuje działania, aby zatrzymać lub zgłosić ten rodzaj wystąpienia.

Zaawansowane techniki IBM WebSphere MQ

W przypadku prostej aplikacji produktu IBM WebSphere MQ należy zdecydować, które obiekty produktu WebSphere MQ mają być używane w aplikacji, oraz typy komunikatów, które mają być używane. W przypadku bardziej zaawansowanych aplikacji warto skorzystać z niektórych technik wprowadzonych w poniższych sekcjach.

Oczekiwanie na komunikaty

Program obsługujący kolejkę może czekać na komunikaty przez:

- Trwa oczekiwanie na nadejście komunikatu lub upłynie określony przedział czasu (patrz “Oczekiwanie na komunikaty” na stronie 275).
- Ustanawianie wyjścia wywołania zwrotnego, które ma być sterowane po nadejściu komunikatu; patrz “Asynchroniczne wykorzystanie komunikatów produktu IBM WebSphere MQ” na stronie 34.
- Wykonywanie okresowych wywołań w kolejce w celu sprawdzenia, czy komunikat został wysłany (*odpytywanie*). Zwykle nie jest to zalecane, ponieważ może mieć wpływ na wydajność.

Korelowanie odpowiedzi

W aplikacjach WebSphere MQ , gdy program odbiera komunikat, który żąda jego wykonania, program zwykle wysyła do requestera jeden lub więcej komunikatów odpowiedzi.

Aby pomóc requesterowi w powiązaniu tych odpowiedzi z oryginalnym żądaniem, aplikacja może ustawić pole *identyfikator korelacji* w deskryptorze każdego komunikatu. Następnie programy kopiują identyfikator komunikatu żądania do pola identyfikatora korelacji w ich komunikatach odpowiedzi.

Ustawianie i używanie informacji kontekstowych

Informacje o kontekście są używane do tworzenia powiązań komunikatów z użytkownikiem, który je wygenerował, a także do identyfikowania aplikacji, która wygenerował komunikat. Takie informacje są przydatne w celu zapewnienia bezpieczeństwa, rozliczania, kontroli i określania problemów.

Podczas tworzenia komunikatu można określić opcję, która żąda, aby menedżer kolejek powiąże domyślne informacje kontekstu z komunikatem.

Aby uzyskać więcej informacji na temat używania i ustawiania informacji o kontekście, zobacz: “Kontekst komunikatu” na stronie 39.

Automatyczne uruchamianie programów WebSphere MQ

Użyj programu WebSphere MQ *wyzwalania* , aby uruchomić program automatycznie, gdy komunikaty pojawiają się w kolejce.

Można ustawić warunki wyzwalacza w kolejce, tak aby program uruchamiał tę kolejkę:

- Za każdym razem, gdy komunikat pojawia się w kolejce
- Gdy pierwszy komunikat pojawia się w kolejce
- Gdy liczba komunikatów w kolejce osiągnie predefiniowaną liczbę

Więcej informacji na temat wyzwalania zawiera sekcja “Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy” na stronie 338. Wyzwalanie to tylko jeden ze sposobów automatycznego uruchamiania programu. Na przykład można uruchomić program automatycznie przy użyciu licznika czasu przy użyciu narzędzi innych niżWebSphere MQ .

WebSphere MQ umożliwia definiowanie obiektów usług w celu uruchomienia programów WebSphere MQ podczas uruchamiania menedżera kolejek. Patrz sekcja Obiekty usług.

Generowanie raportów produktu WebSphere MQ

W ramach aplikacji można zażądać następujących raportów:

- Raporty dotyczące wyjątków
- Sprawozdania z wygaśnięcia
- Raporty potwierdzenia w momencie przybycia (COA)
- Raporty potwierdzenia dostarczenia (COD)
- Raporty z powiadomieniem o działaniu pozytywnym (PAN)
- Raporty powiadomień o działaniu negatywnym (NAN)

Są one opisane w sekcji “Komunikaty raportów” na stronie 11.

Klastry i powinowactwa komunikatów

Przed rozpoczęciem korzystania z klastrów z wieloma definicjami dla tej samej kolejki należy sprawdzić aplikacje, aby sprawdzić, czy istnieją jakieś wymagające wymiany powiązanych komunikatów.

W obrębie klastra komunikat może być kierowany do dowolnego menedżera kolejek, który udostępnia instancję odpowiedniej kolejki. Dlatego też logika aplikacji z powinowactwa komunikatów może być zdenerwowana.

Na przykład mogą być używane dwa aplikacje, które polegają na serii komunikatów przepływających między nimi w formie pytań i odpowiedzi. Ważne może być, aby wszystkie pytania były wysyłane do tego samego menedżera kolejek oraz aby wszystkie odpowiedzi były wysyłane z powrotem do innego menedżera kolejek. W takiej sytuacji ważne jest, aby procedura zarządzania obciążeniem nie wysyłała komunikatów do żadnego menedżera kolejek, który tak się składa, że tylko w tym przypadku jest hostem instancji odpowiedniej kolejki.

Jeśli to możliwe, usuń powinowactwa. Usunięcie powinowactwa komunikatów zwiększa dostępność i skalowalność aplikacji.

Więcej informacji na ten temat zawiera sekcja Obsługa powinowactw komunikatów .

Przykładowe programy produktu WebSphere MQ

Ta kolekcja tematów zawiera informacje na temat przykładowych programów WebSphere MQ na różnych platformach.

- “Przykładowe programy dla platform rozproszonych” na stronie 99

Pojęcia pokrewne

“Pojęcia związane z projektowaniem aplikacji” na stronie 8
Do pisania aplikacji IBM WebSphere MQ można użyć wyboru języków proceduralnych lub obiektowych. Odsyłacze w tym temacie można znaleźć w informacjach dotyczących pojęć związanych z produktem IBM WebSphere MQ , które są przydatne dla programistów aplikacji.

“Wybór języka programowania do użycia” na stronie 80
W tej sekcji znajdują się informacje na temat języków programowania i środowisk obsługiwanych przez produkt IBM WebSphere MQ, a także niektóre zagadnienia dotyczące ich używania.

“Projektowanie aplikacji produktu IBM WebSphere MQ” na stronie 91
Po zdecydowaniu się, w jaki sposób aplikacje mogą korzystać z platform i środowisk, które są dostępne dla użytkownika, należy zdecydować, w jaki sposób korzystać z funkcji oferowanych przez produkt WebSphere MQ.

“Pisanie aplikacji kolejkowania” na stronie 199
Ta sekcja zawiera informacje na temat pisania aplikacji kolejkowania, łączenia się i rozłączania z menedżerem kolejek, publikowania/subskrypcji oraz obiektów otwierających i zamykających.

“Pisanie aplikacji klienckich” na stronie 360
Co należy wiedzieć, aby pisać aplikacje klienckie w produkcie WebSphere MQ.

“Korzystanie z usług Web Service w produkcie WebSphere MQ” na stronie 972
Aplikacje produktu IBM WebSphere MQ dla usług Web Services można tworzyć przy użyciu transportu IBM WebSphere MQ dla protokołu SOAP lub mostu IBM WebSphere MQ dla protokołu HTTP.

“Zapisywanie aplikacji publikowania/subskrypcji” na stronie 285
Rozpocznij pisanie aplikacji WebSphere MQ publikowania/subskrypcji.

“Budowanie aplikacji IBM WebSphere MQ” na stronie 439
Ta sekcja zawiera informacje na temat budowania aplikacji produktu IBM WebSphere MQ na różnych platformach.

“Obsługa błędów programu” na stronie 561
Te informacje wyjaśniają błędy związane z wywołaniami MQI aplikacji, gdy wywołuje połączenie, lub gdy jego komunikat jest dostarczany do miejsca docelowego.

Przykładowe programy dla platform rozproszonych

W tej sekcji opisano przykładowe programy dostarczane z produktem IBM WebSphere MQ, napisane w językach C i COBOL. Przykłady demonstrują typowe zastosowania interfejsu kolejki komunikatów (Message Queue Interface-MQI).

Przykłady nie mają na celu demonstrować ogólnych technik programowania, dlatego niektóre sprawdzanie błędów, które może być uwzględnione w programie produkcyjnym, jest pomijane. Te przykłady są jednak odpowiednie do użycia jako podstawa dla własnych programów kolejkowania komunikatów.

Kod źródłowy dla wszystkich przykładów jest dostarczany wraz z produktem; źródło to zawiera komentarze, które wyjaśniają techniki kolejkowania komunikatów demonstrowe w programach.

Programy przykładowe C++: Opis przykładowych programów dostępnych w języku C + + znajduje się w sekcji Używanie języka C++ .

Nazwy przykładów rozpoczynają się od przedrostka `amq`. Czwarty znak wskazuje język programowania i kompilator tam, gdzie jest to konieczne.

s	język C
0	Język COBOL w kompilatorach IBM i Micro Focus
i	Język COBOL tylko w kompilatorach IBM
m	Tylko język COBOL na kompilatorach Micro Focus

Ósmy znak kodu wykonywalnego wskazuje, czy próbka działa w trybie powiązania lokalnego, czy w trybie klienta. Jeśli nie ma ósmego znaku, to próbka działa w trybie powiązań lokalnych. Jeśli ósmy znak jest 'c', to próbka jest uruchamiana w trybie klienta. Aby skonfigurować menedżera kolejek w celu akceptowania połączeń klienckich, należy zapoznać się z informacjami szczegółowymi w sekcji “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113 .

Aby dowiedzieć się więcej na temat przykładowych programów, należy użyć poniższych odsyłaczy:

- “Funkcje demonstrowe w przykładowych programach” na stronie 100
- “Programy przykładowe publikowania/subskrypcji” na stronie 139
- “Przykładowe programy umieszczania” na stronie 144
- “Przykładowy program listy dystrybucyjnej” na stronie 131
- “Przeglądanie przykładowych programów” na stronie 119
- “Przykładowy program przeglądarki” na stronie 120
- “Pobieranie przykładowych programów” na stronie 133
- “Przykładowe programy Message Message” na stronie 145
- “Przykładowe programy żądania” na stronie 151
- “Przykładowe programy Inquire” na stronie 138
- “Właściwości Inquire przykładowego programu Message Handle” na stronie 138
- “Przykładowe programy” na stronie 155
- “Programy przykładowe Echo” na stronie 132
- “Przykładowy program do konwersji danych” na stronie 123
- “Przykładowe programy wyzwalające” na stronie 159
- “Przykładowy program do umieszczania w pamięci asynchronicznej” na stronie 119
- “Przykłady koordynacji bazy danych” na stronie 123
- “Przykład transakcji CICS” na stronie 122
- “Próbki TUXEDO” na stronie 160
- “Przykład procedury obsługi kolejki niedostarczanych komunikatów” na stronie 131

- “Przykładowy program Connect” na stronie 122
- “Przykładowy program obsługi wyjścia funkcji API” na stronie 117
- “Korzystanie z wyjścia zabezpieczeń SSPI w systemach Windows” na stronie 173
- “Uruchamianie przykładów przy użyciu kolejek zdalnych” na stronie 174
- “Przykładowy program monitorowania kolejki klastra (AMQSCLM)” na stronie 174
- “Przykładowy program do wyszukiwania punktów końcowych połączenia (CEPL)” na stronie 184

Funkcje demonstrowe w przykładowych programach

Kolekcja tabel, w których przedstawiono techniki demonstrowe przez programy przykładowe WebSphere MQ .

Wszystkie przykładowe kolejki są otwierane i zamykane przy użyciu wywołań MQOPEN i MQCLOSE, dlatego te techniki nie są wymienione oddzielnie w tabelach. Zapoznaj się z nagłówkiem, w którym znajduje się interesowana platforma.

Przykłady dla systemów UNIX and Linux

W tym temacie przedstawiono techniki pokazanego przez programy przykładowe dla produktu WebSphere MQ w systemach UNIX and Linux .

Patrz “Przygotowywanie i uruchamianie przykładowych programów w systemach UNIX” na stronie 115 , aby dowiedzieć się, gdzie są przechowywane przykładowe programy dla produktu WebSphere MQ w systemach UNIX i Linux .

Tabela 14 na stronie 100 Tabela zawiera listę dostępnych plików źródłowych języka C i języka COBOL oraz informacje o tym, czy plik wykonywalny serwera czy klienta jest dołączony.

Tabela 14. Produkt WebSphere MQ w przykładowych programach UNIX and Linux demonstrujących użycie interfejsu MQI (C i COBOL)

Technika	**C (źródło) (“1” na stronie 103)**	**COBOL (źródło) (“2” na stronie 103)**	**Serwer (kod wykonywalny C)**	**Klient (kod wykonywalny C) (“3” na stronie 103)**
Korzystanie z interfejsu publikowania/ subskrybowania	amqspuba amqssuba amqssbxa	brak próby	amqspub amqssub amqssbx	brak próby
Umieszczanie komunikatów przy użyciu wywołania MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Umieszczanie pojedynczego komunikatu przy użyciu wywołania MQPUT1	amqsinqa amqsecha	amqminqx amqmechx amqiinqx amqiechx	amqsinq amqsech	amqsechc
Umieszczanie komunikatów na liście dystrybucyjnej (“4” na stronie 103)	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Odpowiadanie na komunikat żądania	amqsinqa	amqminqx amqiinqx	amqsinq	brak próby
Pobieranie komunikatów (bez oczekiwania)	amqsgbr0	amq0gbr0	amqsgbr	brak próby
Pobieranie komunikatów (czekaj z limitem czasu)	amqsget0	amq0get0	amqsget	amqsgetc
Pobieranie komunikatów (nieograniczony czas oczekiwania)	amqstrg0	brak próby	amqstrg	amqstrgc

Tabela 14. Produkt WebSphere MQ w przykładowych programach UNIX and Linux demonstrujących użycie interfejsu MQI (C i COBOL) (kontynuacja)

Technika	**C (źródło) (“1” na stronie 103)**	**COBOL (źródło) (“2” na stronie 103)**	**Serwer (kod wykonywalny C)**	**Klient (kod wykonywalny C) (“3” na stronie 103)**
Pobieranie komunikatów (z konwersją danych)	amqsecha	brak próby	amqsech	brak próby
Umieszczanie komunikatów referencyjnych w kolejce (“4” na stronie 103)	amqsprma	brak próby	amqsprm	amqsprmc
Pobieranie komunikatów referencyjnych z kolejki (“4” na stronie 103)	amqsgrma	brak próby	amqsgrm	amqsgrmc
Wyjście odwołania kanału komunikatów (“4” na stronie 103)	amqsqrma amqsxrma	brak próby	amqsxrm	brak próby
Przeglądanie pierwszych 20 znaków wiadomości	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Przeglądanie pełnych komunikatów	amqsbcg0	brak próby	amqsbcg	amqsbcgc
Korzystanie z współużytkowanej kolejki wejściowej	amqsinqa	amqminqx amqiinqx	amqsinq	amqsinqc
Korzystanie z wyłącznej kolejki wejściowej	amqstrg0	amq0req0	amqstrg	amqstrgc
Korzystanie z wywołania MQINQ	amqsinqa	amqminqx amqiinqx	amqsinq	brak próby
Korzystanie z wywołania MQSET	amqsseta	amqmsetx amqisetx	amqsset	amqssetc
Korzystanie z kolejki odpowiedzi	amqsreq0	amq0req0	amqsreq	amqsreqc
Żądanie wyjątków komunikatów	amqsreq0	amq0req0	amqsreq	brak próby
Akceptowanie obciętej wiadomości	amqsgbr0	amq0gbr0	amqsgbr	brak próby
Korzystanie z przetłumaczanej nazwy kolejki	amqsgbr0	amq0gbr0	amqsgbr	brak próby
Wyzwalanie procesu	amqstrg0	brak próby	amqstrg	amqstrgc
Używanie konwersji danych	(“5” na stronie 103)	brak próby	brak próby	brak próby
WebSphere MQ (koordynowanie menedżerów baz danych zgodnych z interfejsem XA) z dostępem do jednej bazy danych przy użyciu języka SQL	amqsxas0.sqc DB2 amqsxas0.ec Informix	amq0xas0.sqb	brak próby	brak próby
WebSphere MQ (koordynowanie menedżerów baz danych zgodnych z interfejsem XA) z dostępem do dwóch baz danych przy użyciu języka SQL	amqsxag0.c amqsxab0.sqc amqsxaf0.sqc	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	brak próby	brak próby
Transakcja CICS (“6” na stronie 103)	amqscic0.ccs	brak próby	amqscic0	brak próby
Transakcja Encina (“4” na stronie 103)	amqsxae0	brak próby	amqsxae0	brak próby
Transakcja TUXEDO w celu umieszczania komunikatów (“7” na stronie 103)	amqstxpx	brak próby	brak próby	brak próby

Tabela 14. Produkt WebSphere MQ w przykładowych programach UNIX and Linux demonstrujących użycie interfejsu MQI (C i COBOL) (kontynuacja)

Technika	**C (źródło) (“1” na stronie 103)**	**COBOL (źródło) (“2” na stronie 103)**	**Serwer (kod wykonywalny C)**	**Klient (kod wykonywalny C) (“3” na stronie 103)**
Transakcja TUXEDO w celu pobrania komunikatów (“7” na stronie 103)	amqstxgx	brak próby	brak próby	brak próby
Serwer dla TUXEDO (“7” na stronie 103)	amqstxsx	brak próby	brak próby	brak próby
procedura obsługi kolejki niedostarczonych komunikatów	Katalog `./tools/c/Samples/dlq` (“8” na stronie 103)	brak próby	amqsdlq	brak próby
Z poziomu klienta MQI: umieszczanie komunikatu	brak próby	brak próby	brak próby	amqsputc
Z poziomu klienta MQI: uzyskiwanie komunikatu	brak próby	brak próby	brak próby	amqsgetc
Nawiąże połączenie z menedżerem kolejek przy użyciu produktu MQCONNX	amqscnxc	brak próby	brak próby	amqscnxc
Korzystanie z wyjść funkcji API	amqsaxe0	brak próby	amqsaxe	brak próby
Wyjście równoważenia obciążenia klastra	amqswlm0	brak próby	amqswlm	brak próby
Asynchroniczne umieszczanie komunikatów i pobieranie statusu za pomocą wywołania MQSTAT	amqsapt0	brak próby	amqsapt	amqsaptc
Klienty z możliwością ponownego nawiązania połączenia	amqsphac amqsghac amqsmhac	brak próby	bez zastosowania	amqsphac amqsghac amqsmhac
Używanie konsumentów komunikatów do asynchronicznego korzystania z komunikatów z wielu kolejek	amqscbf0	brak próby	amqscbf	amqscbfc
Określanie informacji o połączeniu SSL/TLS na serwerze MQCONNX	amqssslc	brak próby	bez zastosowania	amqssslc

Tabela 14. Produkt WebSphere MQ w przykładowych programach UNIX and Linux demonstrujących użycie interfejsu MQI (C i COBOL) (kontynuacja)

Technika	C (źródło) (“1” na stronie 103)	COBOL (źródło) (“2” na stronie 103)	Serwer (kod wykonywalny C)	Klient (kod wykonywalny C) (“3” na stronie 103)

Uwagi:

1. Wersja wykonywalna przykładów klienta MQI produktu WebSphere MQ współużytkuje to samo źródło, co przykłady, które działają w środowisku serwera.
2. Skompiluj programy zaczynające się od 'amqm' z kompilatorem Micro Focus COBOL, zaczynające się 'amqi' z kompilatorem języka COBOL IBM , a także z tymi, które rozpoczynają się od 'amq0'.
3. Wykonywalne wersje przykładów klienta MQI produktu WebSphere MQ nie są dostępne w produkcie WebSphere MQ for HP-UX.
4. Obsługiwane tylko w produkcie WebSphere MQ for AIX, WebSphere MQ for HP-UXi WebSphere MQ tylko dla systemu Solaris.
5. W produkcie WebSphere MQ for AIX, WebSphere MQ for HP-UX, and WebSphere MQ for Solaris ten program jest nazywany `amqsvfc0.c`
6. Produkt CICS jest obsługiwany tylko przez produkt WebSphere MQ dla systemów AIX i WebSphere MQ wyłącznie dla systemu HP-UX .
7. Produkt TUXEDO nie jest obsługiwany przez produkt WebSphere MQ for Linux w systemie System p.
8. Źródło dla procedury obsługi kolejki niedostarczonych komunikatów składa się z kilku plików i znajduje się w oddzielnym katalogu.

Szczegółowe informacje na temat wsparcia dla systemów UNIX and Linux są dostępne na stronie wymagań systemowych produktu WebSphere MQ pod adresem Wymagania systemowe produktu IBM WebSphere MQ.

Przykłady dla klienta IBM WebSphere MQ dla produktu HP Integrity NonStop Server

W tym temacie przedstawiono techniki pokazanego przez programy przykładowe dla klienta IBM WebSphere MQ w systemach HP Integrity NonStop Server .

Tabela 15 na stronie 103W tabeli przedstawiono przykładowe programy przykładowe źródłowe C, COBOL i pTAL .

Tabela 15. IBM WebSphere MQ w przykładowych programach HP Integrity NonStop Server demonstrujących użycie języka C, COBOL i pTAL

Technika	C				COBOL		pTAL	
	OSS (źródło)	OSS (plik wykonywalny)	Guardian (Źródło)	Guardian (plik wykonywalny)	OSS (źródło)	Guardian (Źródło)	OSS (źródło)	Guardian (Źródło)
Korzystanie z interfejsu publikowania/subskrybowania	amqspuba.c amqssbxa.c amqssuba.c amqspse0.c	amqspubc amqssbxc amqssubc	MQSPUBC MQSSBXC MQSSUBC	AMQSPUBC AMQSSBXC AMQSSUBC	amq0pub0.cbl amq0sub0.cbl	MQSPUBL MQSSUBL	amqtpub0.tal amqtsub0.tal	MQSPUBT MQSSUBT

Tabela 15. IBM WebSphere MQ w przykładowych programach HP Integrity NonStop Server demonstrujących użycie języka C, COBOL i pTAL (kontynuacja)

Technika	**C**				**COBOL**		**pTAL**	
Umieszczanie komunikatów przy użyciu wywołania MQPUT	amqsput0.c	amqsputc	MQSPUTC	amqsputc	amq0put0.cbl	MQSPUTL	amqtput0.tal	MQSPUTT
Umieszczanie pojedynczego komunikatu przy użyciu wywołania MQPUT1	amqsecha.c	amqsechc	MQSECHC	AMQSECHC			amqtech0.tal	MQSECHT
Umieszczanie komunikatów na liście dystrybucyjnej	amqsptl0.c	amqsptlc	MQSPTLC	AMQSPTLC	amq0ptl0.cbl	MQSPTLL		
Odpowiadanie na komunikat żądania	amqsinqa.c	amqsinqc	MQSINQC	AMQSINQC				
Pobieranie komunikatów (bez oczekiwania)	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		
Pobieranie komunikatów (czekaj z limitem czasu)	amqsget0.c	amqsgetc	MQSGETC	AMQSGETC	amq0get0.cbl	MQSGETL	amqtget0.tal	MQSGETT

Tabela 15. IBM WebSphere MQ w przykładowych programach HP Integrity NonStop Server demonstrujących użycie języka C, COBOL i pTAL (kontynuacja)

Technika	**C**				**COBOL**		**pTAL**	
Pobieranie komunikatów (nieograniczony czas oczekiwania)	amqstrg0.c	amqstrgc	MQSTRGC,	AMQSTRGC				
Pobieranie komunikatów (z konwersją danych)	amqsecha.c	amqsechc	MQSECHC	AMQSECHC				
Umieszczanie komunikatów odniesienia w kolejce	amqsprma.c	amqsprmc	MQSPRMC	Kontekst monitorowania AMQSPRMC				
Pobieranie komunikatów odwołania z kolejki	amqsgrma.c	amqsgrmc	MQSGRMC	AMQSGRMC				
Wyjście odwołania kanału komunikatów	amqsqrma.c amqsxrma.c		MQSQRMC MQSXRMC					
Przeglądanie pierwszych 20 znaków wiadomości	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		
Przeglądanie pełnych komunikatów	amqsbcg0.c	amqsbcgc	MQSBCGC	AMQSBCGC				

Tabela 15. IBM WebSphere MQ w przykładowych programach HP Integrity NonStop Server demonstrujących użycie języka C, COBOL i pTAL (kontynuacja)

Technika	**C**				**COBOL**		**pTAL**	
Korzystanie z współużytkowanej kolejki wejściowej	amqsinqa.c	amqsinqc	MQSINQC	MQSINQC				
Korzystanie z wyłącznej kolejki wejściowej	amqstrg0.c	amqstrgc	MQSTRGC,	AMQSTRGC				
Korzystanie z wywołania MQINQ	amqsinqa.c	amqsinqc	MQSINQC	AMQSINQC				
Korzystanie z wywołania MQSET	amqsseta.c	amqssetc	MQSSETC	AMQSSETC				
Korzystanie z kolejki odpowiedzi	amqsreq0.c	amqsreqc	MQSREQC	AMQSREQC	amq0req0.cbl	MQSREQL		
Żądanie wyjątków komunikatów	amqsreq0.c	amqsreqc	MQSREQC	AMQSREQC	amq0req0.cbl	MQSREQL		
Akceptowanie obciętej wiadomości	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		
Korzystanie z przetłumaczanej nazwy kolejki	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		

Tabela 15. IBM WebSphere MQ w przykładowych programach HP Integrity NonStop Server demonstrujących użycie języka C, COBOL i pTAL (kontynuacja)

Technika	**C**				**COBOL**		**pTAL**	
Wyzwalanie procesu	amqstrg0.c	amqstrgc	MQSTRGC,	AMQSTRGC				
Używanie konwersji danych	amqsvfc0.c							
Procedura obsługi kolejki niedostarczanych komunikatów (1)	Katalog ./samp/dlq							
Nawiązywanie połączenia z menedżerem kolejek przy użyciu produktu MQCONNX	amqscnxc.c	amqscnxc	MQSCNXC					
Korzystanie z wyjść funkcji API	amqsaxe0.c amqsaem0.c							
Wyjście równoważenia obciążenia klastra	amqswlm0.c		MQSWLMC					
Monitor kolejki klastra	amqsclma.c							

Tabela 15. IBM WebSphere MQ w przykładowych programach HP Integrity NonStop Server demonstrujących użycie języka C, COBOL i pTAL (kontynuacja)

Technika	**C**				**COBOL**		**pTAL**	
Asynchroniczne umieszczanie komunikatów i pobieranie statusu za pomocą wywołania MQSTAT	amqsapt0.c	amqsaptc	MQSAPTC	MQSAPTC				
Klienty z możliwością ponownego nawiązania połączenia	amqsghac.c amqsmhac.c amqsphac.c	amqsghac amqsmhac amqsphac	MQSGHAC MQSMHAC MQSPHAC MQSFHAC -MQSFHAC	AMQSGHAC AMQSMHAC AMQSPHAC AMQSFHAC				
Korzystanie z konsumentów komunikatów w celu asynchronicznego korzystania z komunikatów z wielu kolejek	amqscbf0.c	amqscbfc						
Określanie informacji o połączeniu SSL/TLS na serwerze MQCONNX	amqsssslc.c	amqsssslc	MQSSSLC	AMQSSSLC				

Tabela 15. IBM WebSphere MQ w przykładowych programach HP Integrity NonStop Server demonstrujących użycie języka C, COBOL i pTAL (kontynuacja)

Technika	**C**				**COBOL**		**pTAL**	
Śledzenie działania	amqsact0.c	amqsactc	MQSACTC	AMQSACTC				
Właściwości komunikatu	amqsiqma.c amqsstma.c	amqsiqmc amqsstmc	MQSIQMC MQSSTMC	AMQSIQMC AMQSSTMC				
Serwer komend	amqsstop.c		MQSSTOC					
Zdarzenia dziennika	amqslog0.c	amqslogc	MQSLOGC	AMQSLOGC				
Rozliczenia	amqsmon0.c	amqsmonc	MQSMONC	AMQSMONC				
Interfejs administracyjny	amqsaicq.c amqsaiem.c amqsailq.c							
Przykład podstawowej funkcji języka C na potrzeby wywoływania komendy pTAL			MQSPTMC					

Uwagi:

1. Źródło dla procedury obsługi kolejki niedostarczonych komunikatów składa się z kilku plików i znajduje się w oddzielnym katalogu.
2. Informacje na temat tworzenia aplikacji dla klienta IBM WebSphere MQ na platformie HP Integrity NonStop Server można znaleźć pod adresem:
   - “Budowanie aplikacji w systemie HP Integrity NonStop Server” na stronie 446
     - “Przygotowywanie programów w języku C w programie HP Integrity NonStop Server” na stronie 448
     - “Przygotowywanie programów w języku COBOL” na stronie 449
     - “Przygotowywanie programów pTAL” na stronie 450

Przykłady dla produktu IBM WebSphere MQ for Windows

W tym temacie przedstawiono techniki pokazanego przez programy przykładowe dla systemu IBM WebSphere MQ dla systemu Windows.

Tabela 16 na stronie 110Tabela zawiera listę dostępnych plików źródłowych języka C i języka COBOL oraz informacje o tym, czy plik wykonywalny serwera czy klienta jest dołączony.

Tabela 16. IBM WebSphere MQ dla przykładowych programów Windows demonstrujących użycie interfejsu MQI (C i COBOL)

Technika	**C (źródło)**	**COBOL (źródło)**	**Serwer (kod wykonywalny C)**	**Klient (kod wykonywalny C)**
Korzystanie z interfejsu publikowania/ subskrybowania	amqspuba amqssuba amqssbxa	brak próby	amqspub amqssub amqssbx	brak próby
Umieszczanie komunikatów przy użyciu wywołania MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Umieszczanie pojedynczego komunikatu przy użyciu wywołania MQPUT1	amqsinqa amqsecha	amqminq2 amqmech2 amqiinq2 amqiech2	amqsinq amqsech	amqsinqc amqsechc
Umieszczanie komunikatów na liście dystrybucyjnej	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Odpowiadanie na komunikat żądania	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Pobieranie komunikatów (bez oczekiwania)	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Pobieranie komunikatów (czekaj z limitem czasu)	amqsget0	amq0get0	amqsget	amqsgetc
Pobieranie komunikatów (nieograniczony czas oczekiwania)	amqstrg0	brak próby	amqstrg	amqstrgc
Pobieranie komunikatów (z konwersją danych)	amqsecha	brak próby	amqsech	amqsechc
Umieszczanie komunikatów odniesienia w kolejce	amqsprma	brak próby	amqsprm	amqsprmc
Pobieranie komunikatów odwołania z kolejki	amqsgrma	brak próby	amqsgrm	amqsgrmc
Wyjście odwołania kanału komunikatów	amqsqrma amqsxrma	brak próby	amqsxrm	brak próby
Przeglądanie pierwszych 20 znaków wiadomości	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Przeglądanie pełnych komunikatów	amqsbcg0	brak próby	amqsbcg	amqsbcgc
Korzystanie z współużytkowanej kolejki wejściowej	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Korzystanie z wyłącznej kolejki wejściowej	amqstrg0	amq0req0	amqstrg	amqstrgc
Korzystanie z wywołania MQINQ	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Korzystanie z wywołania MQSET	amqsseta	amqmset2 amqiset2	amqsset	amqssetc

Tabela 16. IBM WebSphere MQ dla przykładowych programów Windows demonstrujących użycie interfejsu MQI (C i COBOL) (kontynuacja)

Technika	**C (źródło)**	**COBOL (źródło)**	**Serwer (kod wykonywalny C)**	**Klient (kod wykonywalny C)**
Korzystanie z wywołania MQINQMP	amqsiqma	brak próby	brak próby	brak próby
Korzystanie z kolejki odpowiedzi	amqsreq0	amq0req0	amqsreq	amqsreqc
Żądanie wyjątków komunikatów	amqsreq0	amq0req0	amqsreq	amqsreqc
Akceptowanie obciętej wiadomości	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Korzystanie z przetłumaczanej nazwy kolejki	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Wyzwalanie procesu	amqstrg0	brak próby	amqstrg	amqstrgc
Używanie konwersji danych	amqsvfc0	brak próby	brak próby	brak próby
WebSphere MQ (koordynowanie menedżerów baz danych zgodnych z interfejsem XA) z dostępem do jednej bazy danych przy użyciu języka SQL	amqsxas0.sqc DB2 amqsxas0.ec Informix	amq0xas0.sqb	brak próby	brak próby
WebSphere MQ (koordynowanie menedżerów baz danych zgodnych z interfejsem XA) z dostępem do dwóch baz danych przy użyciu języka SQL	amqsxag0.c amqsxab0.sqc DB2 amqsxaf0.sqc DB2	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	brak próby	brak próby
Transakcja TUXEDO w celu umieszczania komunikatów	amqstxpx	brak próby	brak próby	brak próby
Transakcja TUXEDO w celu pobrania komunikatów	amqstxgx	brak próby	brak próby	brak próby
Serwer dla TUXEDO	amqstxsx	brak próby	brak próby	brak próby
procedura obsługi kolejki niedostarczonych komunikatów	Katalog `./tools/c/Samples/dlq` (“1” na stronie 112)	brak próby	amqsdlq	brak próby
Z klienta MQI produktu WebSphere MQ umieszczając komunikat	brak próby	brak próby	brak próby	amqsputc
Uzyskiwanie komunikatu z klienta MQI produktu WebSphere MQ	brak próby	brak próby	brak próby	amqsgetc
Nawiąże połączenie z menedżerem kolejek przy użyciu produktu MQCONNX	amqscnxc	brak próby	brak próby	amqscnxc
Korzystanie z wyjść funkcji API	amqsaxe0	brak próby	amqsaxe	brak próby
Równoważenie obciążenia klastra	amqswlm0	brak próby	amqswlm	brak próby
Procedury bezpieczeństwa SSPI	amqsspin	brak próby	amqrspin.dll	amqrspin.dll

Tabela 16. IBM WebSphere MQ dla przykładowych programów Windows demonstrujących użycie interfejsu MQI (C i COBOL) (kontynuacja)

Technika	C (źródło)	COBOL (źródło)	Serwer (kod wykonywalny C)	Klient (kod wykonywalny C)
Asynchroniczne umieszczanie komunikatów i pobieranie statusu za pomocą wywołania MQSTAT	amqsapt0	brak próby	amqsapt	amqsaptc
Klienty z możliwością ponownego nawiązania połączenia	amqsphac amqsghac amqsmhac	brak próby	Nie dotyczy	amqsphac amqsghac amqsmhac
Używanie konsumentów komunikatów do asynchronicznego korzystania z komunikatów z wielu kolejek	amqscbf0	brak próby	amqscbf	amqscbfc
Określanie informacji o połączeniu SSL/TLS na serwerze MQCONNX	amqsssslc	brak próby	bez zastosowania	amqsssslc

Uwagi:

1. Źródło dla procedury obsługi kolejki niedostarczonych komunikatów składa się z kilku plików i znajduje się w oddzielnym katalogu.

Wizualne podstawowe przykłady dla produktu IBM WebSphere MQ dla systemu Windows

W tym temacie przedstawiono techniki pokazanego przez programy przykładowe Visual Basic dla systemu IBM WebSphere MQ dla systemu Windows.

Tabela 17 na stronie 112 przedstawia techniki pokazanego przez program IBM WebSphere MQ dla przykładowych programów Windows .

Projekt może zawierać kilka plików. Po otwarciu projektu w języku Visual Basic, pozostałe pliki są ładowane automatycznie. Nie są dostępne żadne programy wykonywalne.

Wszystkie przykładowe projekty, z wyjątkiem mqtrivc.vbp, są skonfigurowane do pracy z serwerem IBM WebSphere MQ . Aby dowiedzieć się, jak zmienić przykładowe projekty, aby pracować z klientami IBM WebSphere MQ , należy zapoznać się z “Przygotowywanie programów Visual Basic w systemie Windows” na stronie 475.

Tabela 17. IBM WebSphere MQ dla przykładowych programów Windows demonstrujących użycie interfejsu MQI (Visual Basic)

Technika	Nazwa pliku projektu
Umieszczanie komunikatów przy użyciu wywołania MQPUT	amqsputb.vbp
Pobieranie komunikatów za pomocą wywołania MQGET	amqsgetb.vbp
Przeglądanie kolejki przy użyciu wywołania MQGET	amqsbcgb.vbp
Prosty przykład MQGET i MQPUT (klient)	mqtrivc.vbp
Prosty przykład MQGET i MQPUT (serwer)	mqtrivs.vbp
Umieszczanie i pobieranie łańcuchów i struktur zdefiniowanych przez użytkownika za pomocą MQPUT i MQGET	strings.vbp
Używanie struktur PCF do uruchamiania i zatrzymywania kanału	pcfsamp.vbp
Tworzenie kolejki przy użyciu interfejsu MQAI	amqsaicq.vbp

Tabela 17. IBM WebSphere MQ dla przykładowych programów Windows demonstrujących użycie interfejsu MQI (Visual Basic) (kontynuacja)	
Technika	**Nazwa pliku projektu**
Wyświetlanie kolejek menedżera kolejek za pomocą interfejsu MQAI	amqsailq.vbp
Monitorowanie zdarzeń za pomocą interfejsu MQAI	amqsaiem.vbp

Przygotowywanie i uruchamianie przykładowych programów

Skonfiguruj menedżer kolejek w taki sposób, aby bezpiecznie akceptować przychodzące żądania połączeń z aplikacji działających w trybie klienta.

Zanim rozpoczniesz

Upewnij się, że menedżer kolejek już istnieje i został uruchomiony. Określ, czy rekordy uwierzytelniania kanału są już włączone w następujący sposób:

```
DISPLAY QMGR CHLAUTH
```

To zadanie oczekuje, że włączone są rekordy uwierzytelniania kanału. Jeśli jest to menedżer kolejek używany przez innych użytkowników i aplikacje, zmiana tego ustawienia będzie mieć wpływ na wszystkich innych użytkowników i aplikacje. Jeśli menedżer kolejek nie korzysta z rekordów uwierzytelniania kanału, krok “4” na stronie 113 można zastąpić alternatywną metodą uwierzytelniania (na przykład wyjście bezpieczeństwa), które ustawia wartość MCAUSER na wartość *non-privileged-user-id* (identyfikator użytkownika bez uprawnień uprzywilejowanych), który zostanie uzyskany w kroku “1” na stronie 113.

Należy wiedzieć, która nazwa kanału ma być używana przez aplikację w taki sposób, aby aplikacja mogła używać kanału. Należy również wiedzieć, które obiekty, na przykład kolejki lub tematy, mają być używane przez aplikację, dzięki czemu aplikacja będzie mogła je używać.

O tym zadaniu

To zadanie tworzy nieuprawnionego ID użytkownika, który ma być używany dla aplikacji klienckiej, która łączy się z menedżerem kolejek. Dostęp do aplikacji klienckiej jest udzielany tylko w celu użycia kanału, którego potrzebuje, oraz kolejki, której potrzebuje, korzystając z tego identyfikatora użytkownika.

Procedura

1. Uzyskaj identyfikator użytkownika w systemie, na którym jest uruchomiony menedżer kolejek. W przypadku tego zadania ten identyfikator użytkownika nie może być uprzywilejowanym użytkownikiem administracyjnym. Ten identyfikator użytkownika będzie miał uprawnienia, w ramach którego połączenie klienta zostanie uruchomione w menedżerze kolejek.
2. Uruchom program nasłuchujący z następującymi komendami, w których:

   qmgr to nazwa menedżera kolejek
   nnnn jest wybranym numerem portu

   a) W systemach UNIX i Windows :

   ```
   runmqlsr -t tcp -m qmgr -p nnnn
   ```

3. Jeśli aplikacja korzysta z systemu SYSTEM.DEF.SVRCONN , a następnie ten kanał jest już zdefiniowany. Jeśli aplikacja używa innego kanału, utwórz ją, wydając komendę MQSC:

   ```
   DEFINE CHANNEL('channel-name') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Channel for use by sample programs')
   ```

   nazwa-kanału jest nazwą kanału.
4. Utwórz regułę uwierzytelniania kanału, zezwalając na korzystanie z kanału tylko przez adres IP systemu klienckiego za pomocą komendy MQSC:

```
SET CHLAUTH('channel-name') TYPE(ADDRESSMAP) ADDRESS('client-machine-IP-address') +
MCAUSER('non-privileged-user-id')
```

nazwa-kanału jest nazwą kanału.

adres_IP jest adresem IP systemu klienckiego użytkownika.

Jeśli przykładowa aplikacja kliencka działa na tym samym komputerze, co menedżer kolejek, należy użyć adresu IP '127.0.0.1', jeśli aplikacja ma nawiązać połączenie przy użyciu 'localhost'. Jeśli zostanie nawiązane połączenie kilku różnych komputerów klienckich, można użyć wzorca lub zakresu zamiast pojedynczego adresu IP. Szczegółowe informacje na ten temat zawiera sekcja Ogólne adresy IP .

identyfikator-użytkownika bez uprawnień jest identyfikatorem użytkownika uzyskanym w kroku “1” na stronie 113

5. Jeśli aplikacja korzysta z systemu SYSTEM.DEFAULT.LOCAL.QUEUE , a następnie ta kolejka jest już zdefiniowana. Jeśli aplikacja używa innej kolejki, utwórz ją, wydając komendę MQSC:

```
DEFINE QLOCAL('queue-name') DESCR('Queue for use by sample programs')
```

nazwa-kolejki jest nazwą kolejki.

6. Przyznaj dostęp do połączenia z menedżerem kolejek i sprawdź, czy menedżer kolejek:

   a) W przypadku systemów UNIX iWindows komenda MQSC jest wydawaana przez następujące komendy:

```
SET AUTHREC OBJTYPE(QMGR) PRINCIPAL('non-privileged-user-id') +
AUTHADD(CONNECT, INQ)
```

identyfikator-użytkownika bez uprawnień jest identyfikatorem użytkownika uzyskanym w kroku “1” na stronie 113

7. Jeśli aplikacja jest aplikacją typu punkt z punktem, to znaczy, że korzysta z kolejek, nadając uprawnienia do uzyskiwania informacji oraz umieszczając i pobierając komunikaty przy użyciu kolejki przy użyciu identyfikatora użytkownika, który ma być używany, wydając komendy MQSC:

   a) W przypadku systemów UNIX i Windows komenda MQSC jest wydawaana przez następujące komendy:

```
SET AUTHREC PROFILE('queue-name') OBJTYPE(QUEUE) +
PRINCIPAL('non-privileged-user-id') AUTHADD(PUT, GET, INQ, BROWSE)
```

nazwa-kolejki jest nazwą kolejki.

identyfikator-użytkownika bez uprawnień jest identyfikatorem użytkownika uzyskanym w kroku “1” na stronie 113

8. Jeśli aplikacja jest aplikacją publikowania/subskrypcji, to znaczy, że korzysta z tematów, przyznaj dostęp, aby zezwolić na publikowanie i subskrybowanie tematu przy użyciu identyfikatora użytkownika, który ma być używany, wydając komendy MQSC:

   a) W przypadku systemów UNIX i Windows komenda MQSC jest wydawaana przez następujące komendy:

```
SET AUTHREC PROFILE('SYSTEM.BASE.TOPIC') OBJTYPE(TOPIC) +
PRINCIPAL('non-privileged-user-id') AUTHADD(PUB, SUB)
```

identyfikator-użytkownika bez uprawnień jest identyfikatorem użytkownika uzyskanym w kroku “1” na stronie 113

Spowoduje to nadanie *nieuprawnionego identyfikatora użytkownika* dostępu do dowolnego tematu w drzewie tematów. Można również zdefiniować obiekt tematu przy użyciu produktu **DEFINE TOPIC** i nadać dostęp tylko do części drzewa tematów, do której odwołuje się ten obiekt tematu. Szczegółowe informacje na ten temat zawiera sekcja Kontrolowanie dostępu użytkowników do tematów .

Co dalej

Aplikacja kliencka może teraz połączyć się z menedżerem kolejek i umieścić lub pobrać komunikaty przy użyciu kolejki.

Zadania pokrewne

Nadawanie dostępu do obiektu WebSphere MQ w systemach UNIX lub Linux oraz w systemie Windows

Odsyłacze pokrewne

USTAW WARTOŚĆ CHLAUTH

Zdefiniowanie kanału

DEFINE QLOCAL

SET AUTHREC

Przygotowywanie i uruchamianie przykładowych programów w systemach UNIX

Tabela 18. Informacje o tym, gdzie znaleźć przykłady dla produktu WebSphere MQ w systemach UNIX and Linux

Zawartość	**Katalog**
Pliki źródłowe	*MQ_INSTALLATION_PATH*/samp
pliki źródłowe procedur obsługi kolejki niedostarczonych komunikatów	*MQ_INSTALLATION_PATH*/samp/dlq
Pliki wykonywalne	*MQ_INSTALLATION_PATH*/samp/bin
MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .	

Przykładowe pliki produktu WebSphere MQ w systemach UNIX and Linux znajdują się w katalogach wymienionych w sekcji Tabela 18 na stronie 115 , jeśli wartości domyślne zostały użyte w czasie instalacji. Aby uruchomić przykłady, należy użyć dostarczonych wersji kodu wykonywalnego lub skompilować wersje źródłowe tak, jak w przypadku innych aplikacji, używając kompilatora ANSI. Informacje na temat tego sposobu można znaleźć w sekcji “Uruchamianie przykładowych programów” na stronie 116.

Przygotowywanie i uruchamianie przykładowych programów w systemach Windows

Tabela 19. Gdzie znaleźć przykłady dla produktu WebSphere MQ for Windows

Zawartość	**Katalog**
Kod źródłowy C	*MQ_INSTALLATION_PATH*\Tools\C\Samples
Kod źródłowy dla przykładu procedury obsługi niedostarczonych komunikatów	*MQ_INSTALLATION_PATH*\Tools\C\Samples\DLQ
Kod źródłowy COBOL	*MQ_INSTALLATION_PATH*\Tools\Cobol \ Przykłady
Pliki wykonywalne C [1]	*MQ_INSTALLATION_PATH*\ Tools\C\Samples \ Bin (32-bitowe wersje) *MQ_INSTALLATION_PATH*\ Tools\C\Samples\Bin64 (wersje 64-bitowe)
Przykładowe pliki MQSC	*MQ_INSTALLATION_PATH*\Tools\MQSC\Samples
Kod źródłowy Visual Basic	*MQ_INSTALLATION_PATH*\Tools\VB\SampVB6
Przykłady .NET	*MQ_INSTALLATION_PATH*\Tools\dotnet \ Przykłady

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

Uwaga:

1. Wersje 64-bitowe są dostępne dla niektórych plików wykonywalnych w języku C.

Przykładowe pliki produktu WebSphere MQ for Windows znajdują się w katalogach wymienionych w sekcji Tabela 19 na stronie 115 , jeśli wartości domyślne zostały użyte w czasie instalacji. Wartością domyślną napędu instalacyjnego jest < c: >. Aby uruchomić przykłady, należy użyć dostarczonych wersji kodu wykonywalnego lub skompilować wersje źródłowe tak, jak w przypadku innych aplikacji produktu WebSphere MQ for Windows . Informacje na temat sposobu wykonania tej czynności zawiera sekcja “Uruchamianie przykładowych programów” na stronie 116.

Uruchamianie przykładowych programów

Należy rozważyć użycie tego tematu podczas uruchamiania programów przykładowych na różnych platformach.

Przed uruchomieniem dowolnego z przykładowych programów należy utworzyć menedżer kolejek i skonfigurować definicje domyślne. Jest to wyjaśnione w sekcji Administrowanie.

Na platformach Windows, UNIX i Linux

W przypadku przykładów wymagany jest zestaw kolejek do pracy. Aby utworzyć zestaw, należy użyć własnych kolejek lub uruchomić przykładowy plik MQSC amqscos0.tst .

Aby wykonać tę czynności w systemach UNIX and Linux , wpisz:

- `runmqsc QManagerName <amqscos0.tst >/tmp/sampobj.out`

Sprawdź plik `sampobj.out` , aby upewnić się, że nie wystąpiły żadne błędy.

W tym celu w systemach Windows wpisz:

- `runmqsc QManagerName  <amqscos0.tst > sampobj.out`

Sprawdź plik `sampobj.out` , aby upewnić się, że nie wystąpiły żadne błędy. Ten plik znajduje się w bieżącym katalogu.

Teraz można uruchomić aplikacje przykładowe. Wprowadź nazwę przykładowej aplikacji, po której następują dowolne parametry, na przykład:

- `amqsput myqueue qmanagername`

gdzie `myqueue` jest nazwą kolejki, w której mają zostać umieszczone komunikaty, a `qmanagername` jest menedżerem kolejek, który jest właścicielem produktu `myqueue`.

Zapoznaj się z opisem poszczególnych próbek, aby uzyskać informacje na temat parametrów, których oczekuje każdy z nich.

Długość nazwy kolejki

W przypadku przykładowych programów w języku COBOL, gdy nazwy kolejek są przekazywane jako parametry, należy podać 48 znaków, dopełnianie pustych znaków, jeśli to konieczne. Wszystkie inne znaki niż 48 znaków powodują, że program nie powiedzie się z kodem przyczyny 2085.

Sprawdź, czy przykłady, Ustaw i Echo

W przypadku przykładów Inquire, Set i Echo, przykładowe definicje wyzwalają wersje C tych przykładów.

Aby wersje w języku COBOL były zmieniane, należy zmienić definicje procesów:

- SYSTEM.SAMPLE.INQPROCESS
- SYSTEM.SAMPLE.SETPROCESS
- SYSTEM.SAMPLE.ECHOPROCESS

W systemach Windowssystemy UNIX and Linux to robią, edytując plik `amqscos0.tst` i zmieniając nazwy plików wykonywalnych w języku C na nazwy plików wykonywalnych języka COBOL przed użyciem komendy **runmqsc** , jak to pokazano wcześniej.

Przykładowy program obsługi wyjścia funkcji API

Przykładowe wyjście interfejsu API generuje dane śledzenia MQI w pliku określonym przez użytkownika z przedrostkiem zdefiniowanym w zmiennej środowiskowej MQAPI_TRACE_LOGFILE.

Więcej informacji na temat wyjść funkcji API zawiera sekcja “Pisanie i kompilowanie wyjść funkcji API” na stronie 397.

Źródło
amqsaxe0.c

Binarny
amqsaxe

Konfigurowanie dla wyjścia przykładu

1. Dodaj następujące informacje do pliku qm.ini .

   Platformy inne niż Windows

   ```
   ApiExitLocal:
      Sequence=100
      Function=EntryPoint
      Module=MQ_INSTALLATION_PATH/samp/bin/amqsaxe
      Name=SampleApiExit
   ```

   gdzie *MQ_INSTALLATION_PATH* oznacza katalog, w którym zainstalowano produkt IBM WebSphere MQ .

   Windows

   ```
   ApiExitLocal:
      Sequence=100
      Function=EntryPoint
      Module=MQ_INSTALLATION_PATH\Tools\c\Samples\bin\amqsaxe
      Name=SampleApiExit
   ```

   gdzie *MQ_INSTALLATION_PATH* oznacza katalog, w którym zainstalowano produkt IBM WebSphere MQ .

2. Ustaw zmienną środowiskową

   ```
   MQAPI_TRACE_LOGFILE=/tmp/MqiTrace
   ```

3. Uruchom aplikację.

   Pliki wyjściowe są tworzone w katalogu /tmp z nazwami takimi jak: `MqiTrace.<pid>.<tid>.log`

Przykładowy program konsumpcji asynchronicznej

Przykładowy program amqscbf demonstruje użycie komendy MQCB i MQCTL w celu asynchronicznego korzystania z komunikatów z wielu kolejek.

amqscbf jest dostarczany jako kod źródłowy C, a klient binarny i plik wykonywalny serwera na platformach Windows, UNIX and Linux .

Program jest uruchamiany z poziomu wiersza komend i przyjmuje następujące parametry opcjonalne:

```
Usage: [Options] <Queue Name> { <Queue Name> }
  where Options are:
    -m <Queue Manager Name>
    -o <Open options>
    -r <Reconnect Type>
```

```
       d Reconnect Disabled
       r Reconnect
       m Reconnect Queue Manager
```

Podaj więcej niż jedną nazwę kolejki, aby odczytać komunikaty z wielu kolejek (maksymalnie dziesięć kolejek jest obsługiwanych przez próbkę).

Uwaga: Opcja *Typ ponownego połączenia* jest poprawna tylko dla programów klienckich.

Przykład

W przykładzie przedstawiono komendę amqscbf run as a server program reading one message from QL1 and then being zatrzymany.

Użyj programu WebSphere MQ Explorer, aby umieścić komunikat testowy w produkcie QL1. Zatrzymaj program, naciskając klawisz Enter.

```
C:\>amqscbf QL1
Sample AMQSCBF0 start

Press enter to end
Message Call (9 Bytes) :
Message 1

Sample AMQSCBF0 end
```

Co demonstruje amqscbf

W przykładzie pokazano, jak odczytywać komunikaty z wielu kolejek w kolejności ich przybycia. Wymagałoby to dużo większej liczby kodów przy użyciu synchronicznego wywołania MQGET. W przypadku wykorzystania asynchronicznego, odpytywanie nie jest wymagane, a zarządzanie wątkami i pamięcią masową jest wykonywane przez produkt WebSphere MQ. Przykład "realny świat" musiałby poradzić sobie z błędami; w przykładowych błędach wypisuje się na konsolę.

Przykładowy kod składa się z następujących kroków:

1. Zdefiniuj pojedynczą funkcję zwrotną wykorzystania komunikatów,

   ```
   void MessageConsumer(MQHCONN      hConn,
                        MQMD    * pMsgDesc,
                        MQGMO   * pGetMsgOpts,
                        MQBYTE  * Buffer,
                        MQCBC   * pContext)
   { ... }
   ```

2. Połącz się z menedżerem kolejek,

   ```
   MQCONNX(QMName,&cno,&Hcon,&CompCode,&CReason);
   ```

3. Otwórz kolejki wejściowe i powiąż każdą z nich za pomocą funkcji zwrotnej `MessageConsumer` ,

   ```
    MQOPEN(Hcon,&od,O_options,&Hobj,&OpenCode,&Reason);
    cbd.CallbackFunction = MessageConsumer;
    MQCB(Hcon,MQOP_REGISTER,&cbd,Hobj,&md,&gmo,&CompCode,&Reason);
   ```

   `cbd.CallbackFunction` nie musi być ustawiany dla każdej kolejki; jest to pole tylko wejściowe. Możliwe jest jednak powiązanie innej funkcji zwrotnej z każdą kolejką.

4. Rozpoczęcie korzystania z komunikatów,

   ```
   MQCTL(Hcon,MQOP_START,&ctlo,&CompCode,&Reason);
   ```

5. Poczekaj, aż użytkownik kliknie klawisz Enter, a następnie zaprzestanie korzystania z komunikatów,

   ```
   MQCTL(Hcon,MQOP_STOP,&ctlo,&CompCode,&Reason);
   ```

6. W końcu rozłącz się z menedżerem kolejek,

```
MQDISC(&Hcon,&CompCode,&Reason);
```

Przykładowy program do umieszczania w pamięci asynchronicznej

Sekcja zawiera informacje na temat uruchamiania przykładu amqsapt oraz projektu przykładowego programu Asynchroniczne umieszczanie przykładu.

Program przykładowy wstawiany asynchronicznie umieszcza komunikaty w kolejce przy użyciu wywołania asynchronicznego MQPUT, a następnie pobiera informacje o statusie za pomocą wywołania MQSTAT. Nazwę tego programu można znaleźć w sekcji “Funkcje demonstrowe w przykładowych programach” na stronie 100 na różnych platformach.

Uruchamianie przykładu amqsapt

Ten program przyjmuje maksymalnie 6 parametrów:

1. Nazwa kolejki docelowej (wymagane)
2. Nazwa menedżera kolejek (opcjonalnie)
3. Opcje otwarcia (opcjonalnie)
4. Opcje zamknięcia (opcjonalnie)
5. Nazwa docelowego menedżera kolejek (opcjonalnie)
6. Nazwa kolejki dynamicznej (opcjonalnie)

Jeśli menedżer kolejek nie jest określony, amqsapt łączy się z domyślnym menedżerem kolejek.

Projekt przykładowego programu Asynchronicznego Put

Program korzysta z wywołania MQOPEN z dostarczonym opcjami wyjścia lub z opcjami MQOO_OUTPUT i MQOO_FAIL_IF_QUIESCING, aby otworzyć kolejkę docelową w celu umieszczania komunikatów.

Jeśli nie jest możliwe otwarcie kolejki, program wyświetli komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN. Aby program był prosty, w tym i w kolejnych wywołaniach MQI program używa wartości domyślnych dla wielu opcji.

Dla każdego wiersza danych wejściowych program odczytuje tekst do buforu i używa wywołania MQPUT z opcją MQPMO_ASYNC_RESPONSE w celu utworzenia komunikatu datagramu zawierającego tekst tego wiersza i asynchronicznie umieszczonego go w kolejce docelowej. Program jest kontynuowany do momentu osiągnięcia końca danych wejściowych lub wywołanie MQPUT nie powiedzie się. Jeśli program dociera do końca danych wejściowych, zamyka kolejkę za pomocą wywołania MQCLOSE.

Następnie program wysyła wywołanie MQSTAT, zwracając strukturę MQSTS, a następnie wyświetla komunikaty zawierające liczbę pomyślnie umieszczonych komunikatów, liczbę komunikatów wysłanych z ostrzeżeniem oraz liczbę niepowodzeń.

Przeglądanie przykładowych programów

W przykładowych programach przeglądania komunikaty są przeglądane w kolejce przy użyciu wywołania MQGET.

Nazwy tych programów można znaleźć w sekcji “Funkcje demonstrowe w przykładowych programach” na stronie 100 .

Projekt przykładowego programu przeglądania

Program otwiera kolejkę docelową przy użyciu wywołania MQOPEN z opcją MQOO_BROWSE. Jeśli nie jest możliwe otwarcie kolejki, program wyświetli komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN.

Dla każdego komunikatu w kolejce program korzysta z wywołania MQGET w celu skopiowania komunikatu z kolejki, a następnie wyświetla dane zawarte w komunikacie. Wywołanie MQGET używa następujących opcji:

MQGMO_BROWSE_NEXT
Po wywołaniu MQOPEN kursor przeglądania jest ustawiony logicznie przed pierwszym komunikatem w kolejce, dlatego ta opcja powoduje zwrócenie komunikatu ***pierwszy*** , gdy wywołanie jest wykonywane po raz pierwszy.

MQGMO_NO_WAIT
Program nie czeka, jeśli w kolejce nie ma żadnych komunikatów.

Komunikat MQGMO_ACCEPT_TRUNCATED_MSG
Wywołanie MQGET określa bufor o stałej wielkości. Jeśli komunikat jest dłuższy niż ten bufor, w programie zostanie wyświetlony obcięty komunikat wraz z ostrzeżeniem, że komunikat został obcięty.

Program demonstruje sposób, w jaki należy wyczyścić pola *MsgId* i *CorrelId* struktury MQMD po każdym wywołaniu MQGET, ponieważ wywołanie ustawia te pola na wartości zawarte w komunikacie, który pobiera. Usunięcie zaznaczenia tych pól oznacza, że kolejne wywołania MQGET pobierają komunikaty w kolejności, w jakiej komunikaty są przechowywane w kolejce.

Program będzie kontynuował działanie na końcu kolejki; wywołanie MQGET zwraca kod przyczyny MQRC_NO_MSG_AVAILABLE, a program wyświetli komunikat ostrzegawczy. Jeśli wywołanie MQGET nie powiedzie się, w programie zostanie wyświetlony komunikat o błędzie zawierający kod przyczyny.

Następnie program zamknie kolejkę przy użyciu wywołania MQCLOSE.

Systemy UNIX, Linux i Windows

Warto rozważyć użycie tego tematu podczas uczenia się o programach przykładowych w systemach UNIX, Linux i Windows .

Wersja C programu przyjmuje 2 parametry

1. Nazwa kolejki źródłowej (niezbędne)
2. Nazwa menedżera kolejek (opcjonalnie)

Jeśli menedżer kolejek nie jest określony, łączy się z domyślnym menedżerem kolejek. Na przykład wprowadź jedną z następujących opcji:

- `amqsgbr myqueue qmanagername`
- `amqsgbrc myqueue qmanagername`
- `amq0gbr0 myqueue`

gdzie `myqueue` to nazwa kolejki, z której będą wyświetlane komunikaty, a `qmanagername` to menedżer kolejek, który jest właścicielem produktu `myqueue`.

If you omit the `qmanagername`, when running the C sample, it assumes that the default queue manager owns the queue.

Wersja COBOL nie ma żadnych parametrów. Jest on połączony z domyślnym menedżerem kolejek i po uruchomieniu zostanie wyświetlony monit:

```
Please enter the name of the target queue
```

Wyświetlane są tylko pierwsze 50 znaków każdego komunikatu, po którym następuje `- - - truncated` , gdy jest to przypadek.

Przykładowy program przeglądarki

Przykładowy program przeglądarki odczytuje i zapisuje zarówno deskryptor komunikatu, jak i pola treści komunikatu dla wszystkich komunikatów znajdujących się w kolejce.

Przykładowy program jest napisany jako program narzędziowy, nie tylko po to, aby zademonstrować technikę. Nazwy tych programów można znaleźć w sekcji “Funkcje demonstrowe w przykładowych programach” na stronie 100 .

Ten program przyjmuje następujące parametry:

1. Nazwa kolejki źródłowej
2. Nazwa menedżera kolejek
3. Opcjonalny parametr dla właściwości.

Pierwsze dwa parametry wejściowe dla tego programu są obowiązkowe. Na przykład uruchom program w jeden z następujących sposobów:

- `amqsbcg myqueue qmanagername`
- `amqsbcgc myqueue qmanagername`

gdzie `myqueue` to nazwa kolejki, w której mają być przeglądane komunikaty, a `qmanagername` to menedżer kolejek, który jest właścicielem produktu `myqueue`.

Odczytuje on każdy komunikat z kolejki i zapisuje następujące informacje w celu wyjścia standardowego wyjścia:

- Sformatowane pola deskryptora komunikatu
- Dane komunikatu (zrzucone w formacie szesnastkowym i, gdzie jest to możliwe, format znakowy)

Dopuszczalne wartości parametru właściwości to:

Wartość	**Zachowanie**
0	Zachowanie domyślne, tak jak w przypadku systemu V6. Właściwości, które są dostarczane do aplikacji, są zależne od atrybutu kolejki *PropertyControl* , z którego pochodzi komunikat.

1	Uchwyt komunikatu jest tworzony i używany razem z MQGET. Właściwości komunikatu, z wyjątkiem tych, które znajdują się w deskryptorze komunikatu (lub rozszerzeniu), są wyświetlane w podobny sposób do deskryptora komunikatu. Na przykład:
	`****Message properties****`
	`  <property name> : <property value>`
	Lub jeśli żadne właściwości nie są dostępne:
	`****Message properties****`
	`  None`
	Wartości liczbowe są wyświetlane przy użyciu `printf`, wartości łańcuchowe są otaczane w pojedynczych cudzysłowach, a łańcuchy bajtowe są otoczone znakiem X i apostrofami, co dla deskryptora komunikatu.

2	Określono parametr MQGMO_NO_PROPERTIES, tak aby zwracane były tylko właściwości deskryptora komunikatu.
3	Określono wartość MQGMO_PROPERTIES_FORCE_MQRFH2 , tak aby wszystkie właściwości zostały zwrócone w danych komunikatu.
4	Właściwość MQGMO_PROPERTIES_COMPATIBILITY jest określona, tak aby wszystkie właściwości mogły zostać zwrócone w zależności od tego, czy właściwość w wersji 6 jest włączona, w przeciwnym razie właściwości są usuwane.

Program jest ograniczony do drukowania pierwszych 65535 znaków wiadomości i kończy się niepowodzeniem z powodu *obciętego komunikatu* , jeśli zostanie odczytany dłuższy komunikat.

Przykład danych wyjściowych tego programu narzędziowego znajduje się w publikacji Administrowanie .

Przykład transakcji CICS

Dostępny jest przykładowy program transakcyjny CICS o nazwie amqscic0.ccs dla kodu źródłowego i amqscic0 dla wersji wykonywalnej. Transakcje można tworzyć przy użyciu standardowych narzędzi CICS .

Szczegółowe informacje na temat komend wymaganych dla danej platformy zawiera sekcja “Budowanie aplikacji IBM WebSphere MQ” na stronie 439 .

Transakcja odczytuje komunikaty z kolejki transmisji SYSTEM.SAMPLE.CICS.WORKQUEUE w domyślnym menedżerze kolejek i umieszcza je w kolejce lokalnej, której nazwa znajduje się w nagłówku przesyłania komunikatu. Wszystkie niepowodzenia są wysyłane do kolejki SYSTEM.SAMPLE.CICS.DLQ.

Uwaga: Aby utworzyć te kolejki i przykładowe kolejki wejściowe, można użyć przykładowego skryptu MQSC amqscic0.tst .

Przykładowy program Connect

Przykładowy program Connect umożliwia eksplorowanie wywołania MQCONNX i jego opcji z poziomu klienta. Ten przykład łączy się z menedżerem kolejek przy użyciu wywołania MQCONNX, zapytanie o nazwę menedżera kolejek za pomocą wywołania MQINQ i wyświetlenie go. Dowiedz się również o uruchomieniu przykładu amqscnxc.

Uwaga: Przykładowy program Connect jest przykładowym klientem. Można go skompilować i uruchomić na serwerze, ale funkcja ma znaczenie tylko na kliencie, a tylko pliki wykonywalne klienta są dostarczane.

Uruchamianie przykładu amqscnxc

Składnia wiersza komend programu Connect jest następująca:

```
amqscnxc [-x ConnName [-c SvrconnChannelName]] [QMgrName]
```

Parametry są opcjonalne, a ich kolejność nie jest istotna, z wyjątkiem parametru QMgrName, który, o ile został określony, musi być ostatni. Parametry są następujące:

ConnName
Nazwa połączenia TCP/IP menedżera kolejek serwera

Nazwa SvrconnChannel
Nazwa kanału połączenia z serwerem

QMgrName
Nazwa docelowego menedżera kolejek

Jeśli nazwa połączenia TCP/IP nie zostanie określona, MQCONNX zostanie wystawiony z *ClientConnPtr* ustawionym na NULL. Jeśli zostanie określona nazwa połączenia TCP/IP, ale nie jest to kanał połączenia z serwerem (odwrócenie nie jest dozwolone), to przykład użyje nazwy SYSTEM.DEF.SVRCONN. Jeśli docelowy menedżer kolejek nie zostanie określony, próbka łączy się z tym, który menedżer kolejek będzie nasłuchiwać przy danej nazwie połączenia TCP/IP.

Uwaga: Jeśli jako jedyny parametr zostanie wprowadzony znak zapytania lub jeśli zostaną wprowadzone niepoprawne parametry, zostanie wyświetlony komunikat z wyjaśnieniem sposobu korzystania z programu.

Jeśli próbka zostanie uruchomiona bez opcji wiersza komend, do określenia informacji o połączeniu zostanie użyta zawartość zmiennej środowiskowej MQSERVER. (W tym przykładzie wartość MQSERVER jest ustawiona na `SYSTEM.DEF.SVRCONN/TCP/machine.site.company.com`.) Zostaną wyświetlone dane wyjściowe podobne do następujących:

```
Sample AMQSCNXC start
Connecting to the default queue manager
with no client connection information specified.
Connection established to queue manager machine
```

```
Sample AMQSCNXC end
```

Jeśli uruchamiasz przykład i podaj nazwę połączenia TCP/IP i nazwę kanału połączenia z serwerem, ale nie ma nazwy docelowego menedżera kolejek, tak jak to:

```
amqscnxc -x machine.site.company.com -c SYSTEM.ADMIN.SVRCONN
```

Domyślna nazwa menedżera kolejek jest używana, a dane wyjściowe są podobne do następujących:

```
Sample AMQSCNXC start
Connecting to the default queue manager
using the server connection channel SYSTEM.ADMIN.SVRCONN
on connection name machine.site.company.com.
Connection established to queue manager MACHINE

Sample AMQSCNXC end
```

W przypadku uruchomienia przykładu podaj nazwę połączenia TCP/IP i nazwę docelowego menedżera kolejek, tak jak to:

```
amqscnxc -x machine.site.company.com MACHINE
```

Wyświetlane są dane wyjściowe podobne do następujących:

```
Sample AMQSCNXC start
Connecting to queue manager MACHINE
using the server connection channel SYSTEM.DEF.SVRCONN
on connection name machine.site.company.com.
Connection established to queue manager MACHINE

Sample AMQSCNXC end
```

Przykładowy program do konwersji danych

Przykładowy program konwersji danych jest szkieletem procedury wyjścia konwersji danych. Informacje na temat projektowania przykładu konwersji danych.

Nazwy tych programów można znaleźć w sekcji “Funkcje demonstrowe w przykładowych programach” na stronie 100 .

Projekt próby konwersji danych

Każda procedura wyjścia konwersji danych przekształca pojedynczy nazwany format komunikatu. Ten szkielet jest przeznaczony jako opakowanie dla fragmentów kodu generowanych przez program narzędziowy do generowania danych wyjściowych konwersji danych.

Program narzędziowy tworzy jeden fragment kodu dla każdej struktury danych; kilka takich struktur tworzy format, więc do tego szkieletu dodawane są kilka fragmentów kodu w celu utworzenia procedury do konwersji danych w całym formacie.

Następnie program sprawdza, czy konwersja jest sukcesem lub niepowodzeniem, a następnie zwraca wartości wymagane do programu wywołującego.

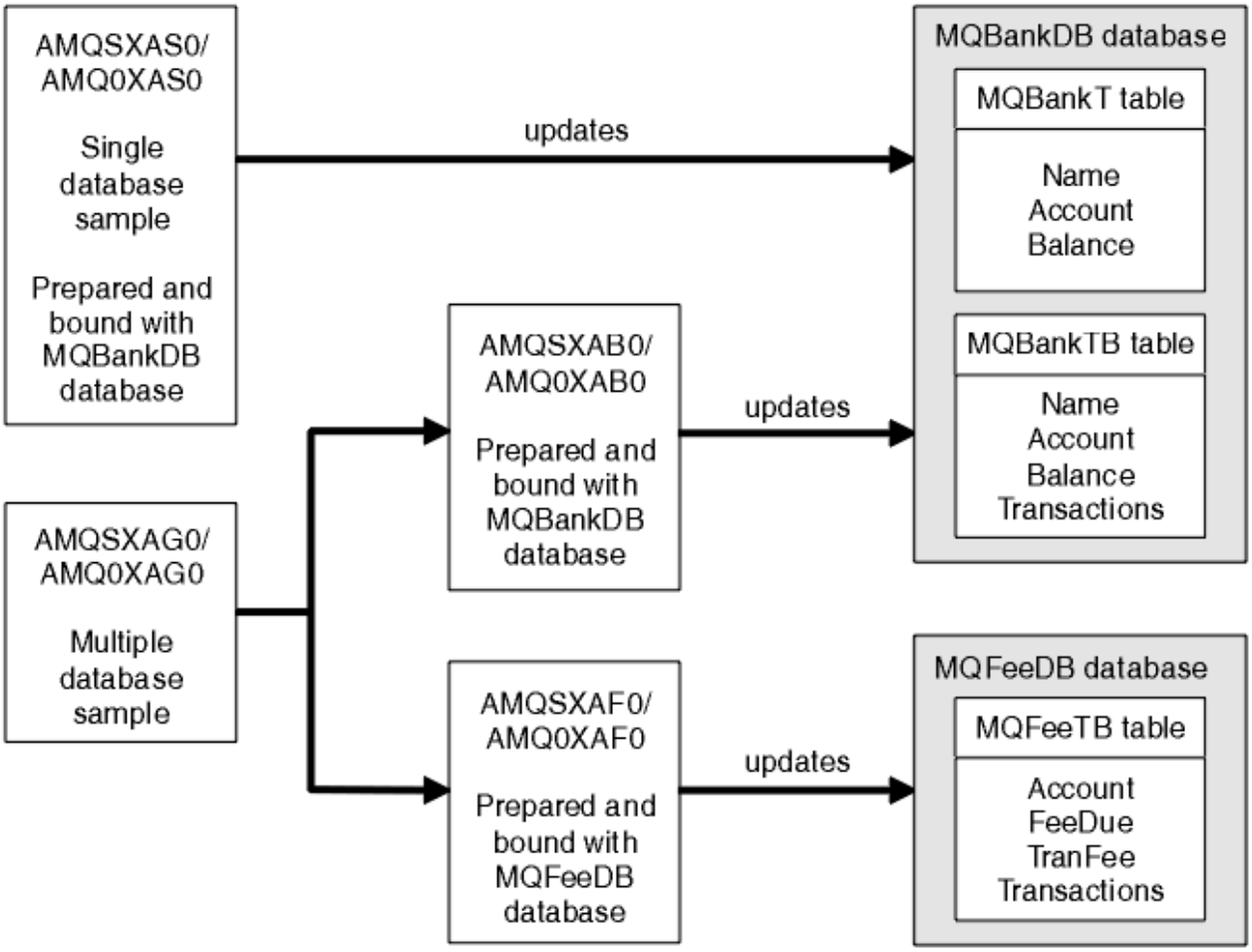
Przykłady koordynacji bazy danych

Udostępniono dwie próbki, które demonstrują sposób, w jaki produkt WebSphere MQ może koordynować zarówno aktualizacje produktu WebSphere MQ , jak i aktualizacje bazy danych w ramach tej samej jednostki pracy.

Są to następujące przykłady:

1. AMQSXAS0 (w języku C) lub AMQ0XAS0 (w języku COBOL), które aktualizuje pojedynczą bazę danych w jednostce pracy produktu WebSphere MQ .
2. AMQSXAG0 (w języku C) lub AMQ0XAG0 (w języku COBOL), AMQSXAB0 (w języku C) lub AMQ0XAB0 (w języku COBOL) i AMQSXAF0 (w języku C) lub AMQ0XAF0 (w języku COBOL), które razem aktualizują dwie bazy danych w ramach jednostki pracy produktu WebSphere MQ , pokazując, w jaki sposób można uzyskać dostęp do wielu baz danych. Te przykłady są dostarczane w celu wyświetlenia użycia wywołań MQBEGIN, mieszanych SQL i WebSphere MQ oraz miejsca, gdzie i kiedy należy nawiązać połączenie z bazą danych.

Rysunek 18 na stronie 124 pokazuje, w jaki sposób udostępnione przykłady służą do aktualizowania baz danych:



Rysunek 18. Przykłady koordynacji bazy danych

Programy odczytają komunikat z kolejki (w punkcie synchronizacji), a następnie, korzystając z informacji znajdujących się w komunikacie, uzyskają odpowiednie informacje z bazy danych i aktualizują je. Następnie zostanie wydrukowany nowy status bazy danych.

Logika programu jest następująca:

1. Użyj nazwy kolejki wejściowej z argumentu programu
2. Nawiąże połączenie z domyślnym menedżerem kolejek (lub opcjonalnie dostarczonym nazwą w języku C) przy użyciu komendy MQCONN.
3. Otwórz kolejkę (za pomocą komendy MQOPEN) dla danych wejściowych, gdy nie występują niepowodzenia.
4. Uruchamianie jednostki pracy przy użyciu komendy MQBEGIN
5. Pobierz następny komunikat (za pomocą komendy MQGET) z kolejki w punkcie synchronizacji

6. Pobierz informacje z baz danych
7. Aktualizacja informacji z baz danych
8. Zatwierdź zmiany za pomocą MQCMIT
9. Wydrukuj zaktualizowane informacje (brak dostępnych komunikatów zliczanych jako niepowodzenie, a pętla kończy się)
10. Zamknij kolejkę za pomocą komendy MQCLOSE
11. Rozłącz się z kolejką za pomocą MQDISC

W próbkach używane są kursory SQL, dzięki czemu odczyty z baz danych (czyli wiele instancji) są blokowane w czasie przetwarzania komunikatu, co pozwala na jednoczesne uruchamianie wielu instancji tych programów. Kursory są jawnie otwierane, ale niejawnie zamykane przy użyciu wywołania MQCMIT.

Pojedyncza próbka bazy danych (AMQSXAS0 lub AMQ0XAS0) nie zawiera instrukcji SQL CONNECT, a połączenie z bazą danych jest niejawnie wykonywane przez produkt WebSphere MQ przy użyciu wywołania MQBEGIN. Przykładowa baza danych (AMQSXAG0 lub AMQ0XAG0, AMQSXAB0 lub AMQ0XAB0oraz AMQSXAF0 lub AMQ0XAF0) zawiera instrukcje SQL CONNECT, ponieważ niektóre produkty bazodanowe zezwalają na tylko jedno aktywne połączenie. Jeśli nie jest to przypadek dla produktu bazodanowego lub jeśli użytkownik uzyskuje dostęp do pojedynczej bazy danych w wielu produktach bazodanowych, instrukcje SQL CONNECT mogą zostać usunięte.

Przykłady są przygotowywane razem z produktem bazodanowym IBM DB2 , dlatego konieczne może być zmodyfikowanie ich w celu pracy z innymi produktami bazodanowymi.

Sprawdzanie błędów SQL używa podprogramów w UTIL.C i CHECKERR.CBL dostarczane przez DB2. Muszą one zostać skompilowane lub zastąpione przed kompilacją i dowiązowaniem.

Uwaga: Jeśli używane jest źródło Micro Focus COBOL, CHECKERR.MFC dla sprawdzania błędów SQL, należy zmienić ID programu na wielkie, czyli CHECKERR, dla AMQ0XAS0 , aby poprawnie dowiązali się.

Tworzenie baz danych i tabel

Utwórz bazy danych i tabele przed kompilacją przykładów.

Aby utworzyć bazy danych, należy użyć zwykłej metody dla produktu bazy danych, na przykład:

```
  DB2 CREATE DB MQBankDB
  DB2 CREATE DB MQFeeDB
```

Utwórz tabele przy użyciu instrukcji SQL w następujący sposób:

W języku C:

```
EXEC SQL CREATE TABLE MQBankT(Name         VARCHAR(40) NOT NULL,
                              Account      INTEGER     NOT NULL,
                              Balance      INTEGER     NOT NULL,
                              PRIMARY KEY (Account));

EXEC SQL CREATE TABLE MQBankTB(Name         VARCHAR(40) NOT NULL,
                               Account      INTEGER     NOT NULL,
                               Balance      INTEGER     NOT NULL,
                               Transactions INTEGER,
                               PRIMARY KEY (Account));

EXEC SQL CREATE TABLE MQFeeTB(Account      INTEGER     NOT NULL,
                              FeeDue       INTEGER     NOT NULL,
                              TranFee      INTEGER     NOT NULL,
                              Transactions INTEGER,
                              PRIMARY KEY (Account));
```

W języku COBOL:

```
 EXEC SQL CREATE TABLE
   MQBankT(Name      VARCHAR(40) NOT NULL,
           Account   INTEGER     NOT NULL,
           Balance   INTEGER     NOT NULL,
```

```
           PRIMARY KEY (Account))
   END-EXEC.

EXEC SQL CREATE TABLE
  MQBankTB(Name      VARCHAR(40) NOT NULL,
           Account   INTEGER     NOT NULL,
           Balance   INTEGER     NOT NULL,
           Transactions INTEGER,
           PRIMARY KEY (Account))
  END-EXEC.

EXEC SQL CREATE TABLE
  MQFeeTB(Account      INTEGER     NOT NULL,
          FeeDue       INTEGER     NOT NULL,
          TranFee      INTEGER     NOT NULL,
          Transactions INTEGER,
          PRIMARY KEY (Account))
  END-EXEC.
```

Wprowadź dane do tabel przy użyciu instrukcji SQL w następujący sposób:

```
EXEC SQL INSERT INTO MQBankT VALUES ('Mr Fred Bloggs',1,0);
EXEC SQL INSERT INTO MQBankT VALUES ('Mrs S Smith',2,0);
EXEC SQL INSERT INTO MQBankT VALUES ('Ms Mary Brown',3,0);
.
EXEC SQL INSERT INTO MQBankTB VALUES ('Mr Fred Bloggs',1,0,0);
EXEC SQL INSERT INTO MQBankTB VALUES ('Mrs S Smith',2,0,0);
EXEC SQL INSERT INTO MQBankTB VALUES ('Ms Mary Brown',3,0,0);
.
EXEC SQL INSERT INTO MQFeeTB VALUES (1,0,50,0);
EXEC SQL INSERT INTO MQFeeTB VALUES (2,0,50,0);
EXEC SQL INSERT INTO MQFeeTB VALUES (3,0,50,0);
.
```

Uwaga: W przypadku języka COBOL należy używać tych samych instrukcji SQL, ale END_EXEC na końcu każdego wiersza.

Prekompilowanie, kompilowanie i łączenie próbek

Ta sekcja zawiera informacje na temat prekompilacji, kompilowania i łączenia przykładów w językach C i COBOL.

Prekompiluj pliki .SQC (w języku C) i .SQB (w języku COBOL) i powiąże je z odpowiednią bazą danych w celu utworzenia plików .C lub .CBL. W tym celu należy użyć typowej metody dla produktu bazodanowego.

Wstępne kompilowanie w języku C

```
db2 connect to MQBankDB
db2 prep AMQSXAS0.SQC
db2 connect reset

db2 connect to MQBankDB
db2 prep AMQSXAB0.SQC
db2 connect reset

db2 connect to MQFeeDB
db2 prep AMQSXAF0.SQC
db2 connect reset
```

Prekompilowanie w języku COBOL

```
db2 connect to MQBankDB
db2 prep AMQ0XAS0.SQB bindfile target ibmcob
db2 bind AMQ0XAS0.BND
db2 connect reset

db2 connect to MQBankDB
db2 prep AMQ0XAB0.SQB bindfile target ibmcob
db2 bind AMQ0XAB0.BND
```

```
db2 connect reset

db2 connect to MQFeeDB
db2 prep AMQ0XAF0.SQB bindfile target ibmcob
db2 bind AMQ0XAF0.BND
db2 connect reset
```

Kompilowanie i łączenie

Następujące przykładowe komendy używają symboli *<DB2TOP>* i *MQ_INSTALLATION_PATH*. *<DB2TOP>* reprezentuje katalog instalacyjny dla produktu DB2 . *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .

- W systemie AIXścieżka do katalogu jest następująca:

  ```
  /usr/lpp/db2_05_00
  ```

- W systemach HP-UX i Solaris ścieżka do katalogu jest następująca:

  ```
  /opt/IBMdb2/V5.0
  ```

- W systemach Windows ścieżka katalogu zależy od ścieżki wybranej podczas instalowania produktu. Jeśli wybrano ustawienia domyślne, ścieżka jest następująca:

  ```
  c:\sqllib
  ```

Uwaga: Przed wprowadzeniem komendy link w systemach Windows należy upewnić się, że zmienna środowiskowa LIB zawiera ścieżki do bibliotek DB2 i WebSphere MQ .

Skopiuj następujące pliki do katalogu tymczasowego:

- Plik `amqsxag0.c` z poziomu instalacji produktu WebSphere MQ

  Uwaga: Plik ten można znaleźć w następujących katalogach:

  – W systemach UNIX and Linux :

  ```
  MQ_INSTALLATION_PATH/samp/xatm
  ```

  – W systemach Windows :

  ```
  MQ_INSTALLATION_PATH\tools\c\samples\xatm
  ```

- Pliki `.c` , które zostały uzyskane przez wstępne kompilowanie plików źródłowych `.sqc` , `amqsxas0.sqc, amqsxaf0.sqc` i `amqsxab0.sqc`
- Pliki `util.c` i `util.h` z instalacji DB2 .

  Uwaga: Pliki te można znaleźć w katalogu:

  ```
  <DB2TOP>/samples/c
  ```

Zbuduj pliki obiektów dla każdego pliku `.c` przy użyciu następującej komendy kompilatora dla używanej platformy:

- AIX

  ```
  xlc_r -IMQ_INSTALLATION_PATH/inc -I
  <DB2TOP>/include -c -o
   <FILENAME>.o <FILENAME>.c
  ```

- HP-UX

```
cc -Aa +z -IMQ_INSTALLATION_PATH/inc -I
<DB2TOP>/include -c -o
 <FILENAME>.o <FILENAME>.c
```

- Solaris

```
cc -Aa -KPIC -mt -IMQ_INSTALLATION_PATH
/inc -I<DB2TOP>/include -c -o
 <FILENAME>.o <FILENAME>.c
```

- Systemy Windows

```
cl /c /IMQ_INSTALLATION_PATH\tools\c\include /I
<DB2TOP>\include
 <FILENAME>.c
```

Utwórz plik wykonywalny amqsxag0 za pomocą następującej komendy łącza dla używanej platformy:

- AIX

```
xlc_r -H512 -T512 -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib
 -lmqm util.o amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- HP-UX wersja 11i

```
ld -E -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib -lmqm -lc -lpthread -lcl
 /lib/crt0.o util.o amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- Solaris

```
cc -mt -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib
 -lmqm -lthread -lsocket -lc -lnsl -ldl util.o
  amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- Systemy Windows

```
link util.obj amqsxaf0.obj amqsxab0.obj amqsxag0.obj mqm.lib db2api.lib
 /out:amqsxag0.exe
```

Utwórz plik wykonywalny produktu amqsxas0 przy użyciu następujących komend kompilacji i dowiązania dla używanej platformy:

- AIX

```
xlc_r -H512 -T512 -L<DB2TOP>/lib -ldb2
 -LMQ_INSTALLATION_PATH/lib -lmqm util.o amqsxas0.o -o amqsxas0
```

- HP-UX wersja 11i

```
ld -E -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib -lmqm -lc -lpthread
 -lcl /lib/crt0.o util.o amqsxas0.o -o amqsxas0
```

- Solaris

```
cc -mt -L<DB2TOP>/lib -ldb2-LMQ_INSTALLATION_PATH/lib
 -lqm -lthread -lsocket -lc -lnsl -ldl util.o
  amqsxas0.o -o amqsxas0
```

- Systemy Windows

```
link util.obj amqsxas0.obj mqm.lib db2api.lib /out:amqsxas0.exe
```

Dodatkowe informacje

Jeśli użytkownik pracuje w systemie AIX lub HP-UX i chce uzyskać dostęp do bazy danych Oracle, należy użyć kompilatora xlc_r i utworzyć odsyłacz do pliku libmqm_r.a.

Uruchamianie przykładów

Informacje zawarte w tej sekcji umożliwiają poznanie sposobu konfigurowania menedżera kolejek przed uruchomieniem przykładów koordynacji bazy danych w językach C i COBOL.

Przed uruchomieniem przykładów należy skonfigurować menedżer kolejek z użyciem używanego produktu bazodanowego. Informacje na temat sposobu wykonania tej czynności zawiera sekcja “Scenariusz 1: Menedżer kolejek wykonuje koordynację” na stronie 43.

Poniższe tytuły zawierają informacje na temat sposobu uruchamiania przykładów w językach C i COBOL:

- “Próbki C” na stronie 129
- “Przykłady języka COBOL” na stronie 130

Próbki C

Komunikaty muszą być w następującym formacie, aby można było odczytać z kolejki:

```
UPDATE Balance change=nnn WHERE Account=nnn
```

Komenda AMQSPUT może być używana do umieszczania komunikatów w kolejce.

Przykłady koordynacji bazy danych przyjmują dwa parametry:

1. Nazwa kolejki (wymagane)
2. Nazwa menedżera kolejek (opcjonalnie)

Zakładając, że utworzono i skonfigurowano menedżer kolejek dla pojedynczej próbki bazy danych o nazwie singDBQM, z kolejką o nazwie singDBQ, przyrost konta Freda Bloggsa o 50 jest zwiększany w następujący sposób:

```
AMQSPUT singDBQ singDBQM
```

Następnie klucz w następującym komunikacie:

```
UPDATE Balance change=50 WHERE Account=1
```

W kolejce można umieścić wiele komunikatów.

```
AMQSXAS0 singDBQ singDBQM
```

Następnie zostanie wydrukowany zaktualizowany status konta pana Freda Bloggsa.

Zakładając, że menedżer kolejek został utworzony i skonfigurowany dla przykładu z wieloma bazami danych o nazwie multDBQM, w kolejce o nazwie multDBQ, należy zmniejszyć konto Mary Brown o 75 w następujący sposób:

```
AMQSPUT multDBQ multDBQM
```

Następnie klucz w następującym komunikacie:

```
UPDATE Balance change=-75 WHERE Account=3
```

W kolejce można umieścić wiele komunikatów.

```
AMQSXAG0 multDBQ multDBQM
```

Następnie zostanie wydrukowany zaktualizowany status konta Mary Brown.

Przykłady języka COBOL

Komunikaty muszą być w następującym formacie, aby można było odczytać z kolejki:

```
UPDATE Balance change=snnnnnnnn WHERE Account=nnnnnnnn
```

W przypadku prostoty `Balance change` musi być liczbą 8-znakową podpisaną, a `Account` musi być ośmioznakowym numerem.

W celu umieszczenia komunikatów w kolejce można użyć przykładowego programu AMQSPUT.

Przykłady nie przyjmują żadnych parametrów i korzystają z domyślnego menedżera kolejek. Można go skonfigurować w taki sposób, aby uruchamiał tylko jedną z przykładów w dowolnym momencie. Zakładając, że domyślny menedżer kolejek został skonfigurowany dla pojedynczej próbki bazy danych, z kolejką o nazwie singDBQ, należy zwiększyć konto Freda Bloggsa o 50, wykonując następujące czynności:

```
AMQSPUT singDBQ
```

Następnie klucz w następującym komunikacie:

```
UPDATE Balance change=+00000050 WHERE Account=00000001
```

W kolejce można umieścić wiele komunikatów:

```
AMQ0XAS0
```

Wpisz nazwę kolejki:

```
singDBQ
```

Następnie zostanie wydrukowany zaktualizowany status konta pana Freda Bloggsa.

Zakładając, że domyślny menedżer kolejek został skonfigurowany dla wielokrotnej bazy danych, w kolejce o nazwie multDBQ, należy zmniejszyć konto Mary Brown o 75 w następujący sposób:

```
AMQSPUT multDBQ
```

Następnie klucz w następującym komunikacie:

```
UPDATE Balance change=-00000075 WHERE Account=00000003
```

W kolejce można umieścić wiele komunikatów:

```
AMQ0XAG0
```

Wpisz nazwę kolejki:

```
multDBQ
```

Następnie zostanie wydrukowany zaktualizowany status konta Mary Brown.

Przykład procedury obsługi kolejki niedostarczanych komunikatów

Podana jest przykładowa procedura obsługi kolejki niedostarczonych komunikatów, która jest nazwą pliku wykonywalnego amqsdlq. Jeśli chcesz, aby procedura obsługi kolejki niedostarczonych komunikatów różniła się od RUNMQDLQ, źródło próbki jest dostępne do użycia jako podstawa.

Przykład jest podobny do procedury obsługi niedostarczonych komunikatów udostępnianych w produkcie, ale śledzenie i raportowanie błędów są różne. Istnieją dwie zmienne środowiskowe dostępne dla użytkownika:

ODQ_TRACE
Ustaw wartość YES lub yes, aby włączyć śledzenie

ODQ_MSG
Służy do ustawiania nazwy pliku zawierającego komunikaty o błędach i komunikaty informacyjne. Podany plik nosi nazwę amqsdlq.msg.

Zmienne te muszą być znane w środowisku za pomocą komend **export** lub **set** , w zależności od platformy. Śledzenie jest wyłączone za pomocą komendy **unset** .

Użytkownik może zmodyfikować plik komunikatów o błędach amqsdlq.msg, aby dostosować go do własnych wymagań. Przykład przedstawia komunikaty do wyjścia standardowego, ***nie*** do pliku dziennika błędów produktu WebSphere MQ .

Podręcznik Administrowanie lub *System Management Guide* dla używanej platformy wyjaśnia, w jaki sposób działa program obsługi niedostarczonych komunikatów i jak go uruchomić.

Przykładowy program listy dystrybucyjnej

Przykład listy dystrybucyjnej amqsptl0 zawiera przykład umieszczania komunikatu w kilku kolejkach komunikatów. Jest on oparty na przykładzie MQPUT amqsput0.

Uruchamianie przykładu Lista dystrybucyjna, amqsptl0

Przykład listy dystrybucyjnej jest uruchamiany w podobny sposób do przykładów umieszczania.

Przyjmuje ona następujące parametry:

- Nazwy kolejek
- Nazwy menedżerów kolejek

Te wartości są wprowadzane jako pary. Na przykład:

```
amqsptl0 queue1 qmanagername1 queue2 qmanagername2
```

Kolejki są otwierane za pomocą komendy MQOPEN, a komunikaty są umieszczane w kolejkach za pomocą komendy MQPUT. Kody przyczyny są zwracane, jeśli żadna z nazw kolejek lub menedżerów kolejek nie została rozpoznana.

Należy pamiętać o definiowaniu kanałów między menedżerami kolejek, tak aby komunikaty mogły przepływać między nimi. Przykładowy program nie robi tego dla Ciebie.

Projekt przykładu listy dystrybucyjnej

Rekordy komunikatów (put Message Records-MQPMRs) określają atrybuty komunikatów dla każdego miejsca docelowego. Przykład udostępnia wartości dla produktów *MsgId* i *CorrelId*, a te nadpisują wartości określone w strukturze MQMD.

Pole *PutMsgRecFields* w strukturze MQPMO wskazuje, które pola są obecne w strukturze MQPMR:

```
MQLONG PutMsgRecFields=MQPMRF_MSG_ID + MQPMRF_CORREL_ID;
```

Następnie próbka przydziela rekordy odpowiedzi i rekordy obiektów. Rekordy obiektów (MQORs) wymagają co najmniej jednej pary nazw i parzystej liczby nazw, tj. *ObjectName* i *ObjectQMgrName*.

Następny etap obejmuje łączenie się z menedżerami kolejek za pomocą MQCONN. Próbka próbuje połączyć się z menedżerem kolejek powiązanym z pierwszą kolejką w tabeli MQOR. Jeśli ta próba nie powiedzie się, przechodzi ona przez rekordy obiektów z kolei. Użytkownik jest informowany, jeśli nie jest możliwe nawiązanie połączenia z dowolnym menedżerem kolejek i wyjście z programu.

Kolejki docelowe są otwierane za pomocą komendy MQOPEN, a komunikat jest umieszczany w tych kolejkach za pomocą komendy MQPUT. Wszelkie problemy i niepowodzenia są zgłaszane w rekordach odpowiedzi (MQRRs).

W końcu kolejki docelowe są zamykane za pomocą komendy MQCLOSE, a program rozłącza się z menedżerem kolejek za pomocą MQDISC. Te same rekordy odpowiedzi są używane dla każdego wywołania, w którym znajdują się *CompCode* i *Reason*.

Programy przykładowe Echo

Przykładowe programy Echo echo to komunikat z kolejki komunikatów do kolejki odpowiedzi.

Nazwy tych programów można znaleźć w sekcji “Funkcje demonstrowe w przykładowych programach” na stronie 100 .

Programy te są przeznaczone do uruchamiania jako programy wyzwalane.

W systemach UNIX, Linux i Windows ich jedyną zmienną wejściową jest struktura MQTMC2 (komunikat wyzwalacza), która zawiera nazwę kolejki docelowej i menedżera kolejek. W wersji COBOL używany jest domyślny menedżer kolejek.

Jeśli definicja została ustawiona poprawnie, najpierw uruchom program AMQSERV4 w jednym zadaniu, a następnie uruchom komendę AMQSREQ4 w innym. Można użyć komendy AMQSTRG4 zamiast AMQSERV4, ale ewentualne opóźnienia w składaniu zadań mogą spowodować, że śledzenie zdarzeń będzie mniej proste.

Aby wysłać komunikaty do kolejki SYSTEM.SAMPLE.ECHO. Przykładowe programy Echo wysyłają komunikat odpowiedzi zawierający dane zawarte w komunikacie żądania do kolejki odpowiedzi określonej w komunikacie żądania.

Projektowanie przykładowych programów Echo

Program otwiera kolejkę nazwaną w strukturze komunikatu wyzwalacza, która została przekazana podczas jej uruchamiania. (Dla jasności zadzwonimy do tej *kolejki żądań*.) Program korzysta z wywołania MQOPEN, aby otworzyć tę kolejkę dla współużytkowanych danych wejściowych.

Program korzysta z wywołania MQGET w celu usunięcia komunikatów z tej kolejki. To wywołanie używa opcji MQGMO_ACCEPT_TRUNCATED_MSG, MQGMO_CONVERT i MQGMO_WAIT, z interwałem oczekiwania na 5 sekund. Program testuje deskryptor każdego komunikatu, aby sprawdzić, czy jest to komunikat żądania. Jeśli tak nie jest, program usuwa komunikat i wyświetla komunikat ostrzegawczy.

Dla każdego wiersza danych wejściowych program odczytuje tekst do buforu i korzysta z wywołania MQPUT1 w celu umieszczenia komunikatu żądania zawierającego tekst tego wiersza w kolejce odpowiedzi.

Jeśli wywołanie MQGET nie powiedzie się, program umieszcza komunikat raportu w kolejce odpowiedzi, ustawiając pole *Feedback* w deskryptorze komunikatu na kod przyczyny zwrócony przez komendę MQGET.

Jeśli w kolejce żądań nie pozostały żadne komunikaty, program zamknie tę kolejkę i rozłącza się z menedżerem kolejek.

Pobieranie przykładowych programów

Program pobierania próbek pobiera komunikaty z kolejki za pomocą wywołania MQGET.

Nazwy tych programów można znaleźć w sekcji “Funkcje demonstrowe w przykładowych programach” na stronie 100 .

Projekt przykładowego programu Get

Program otwiera kolejkę docelową przy użyciu wywołania MQOPEN z opcją MQOO_INPUT_AS_Q_DEF. Jeśli nie można otworzyć kolejki, w programie wyświetlany jest komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN.

Dla każdego komunikatu w kolejce program korzysta z wywołania MQGET w celu usunięcia komunikatu z kolejki, a następnie wyświetla dane zawarte w komunikacie. Wywołanie MQGET używa opcji MQGMO_WAIT, określając wartość *WaitInterval* 15 sekund, co powoduje, że program oczekuje na ten okres, jeśli w kolejce nie ma komunikatu. Jeśli przed upływem tego okresu nie zostanie wyświetlony żaden komunikat, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_NO_MSG_AVAILABLE.

Program demonstruje sposób, w jaki należy usunąć pola *MsgId* i *CorrelId* struktury MQMD po każdym wywołaniu MQGET, ponieważ wywołanie ustawia te pola na wartości zawarte w komunikacie, który pobiera. Usunięcie zaznaczenia tych pól oznacza, że kolejne wywołania MQGET pobierają komunikaty w kolejności, w jakiej komunikaty są przechowywane w kolejce.

Wywołanie MQGET określa bufor o stałej wielkości. Jeśli komunikat jest dłuższy niż ten bufor, wywołanie nie powiedzie się, a program zostanie zatrzymany.

Program będzie kontynuowany do czasu, aż wywołanie MQGET zwróci kod przyczyny MQRC_NO_MSG_AVAILABLE lub wywołanie MQGET nie powiedzie się. Jeśli wywołanie nie powiedzie się, w programie zostanie wyświetlony komunikat o błędzie zawierający kod przyczyny.

Następnie program zamknie kolejkę przy użyciu wywołania MQCLOSE.

Uruchamianie próbek amqsget i amqsgetc

Każdy z tych programów ma dwa parametry:

1. Nazwa kolejki źródłowej (wymagane)
2. Nazwa menedżera kolejek (opcjonalnie)

Jeśli menedżer kolejek nie jest określony, amqsget łączy się z domyślnym menedżerem kolejek, a amqsgetc łączy się z menedżerem kolejek identyfikowany przez zmienną środowiskową lub plik definicji kanału klienta.

Aby uruchomić te programy, wprowadź jedną z następujących opcji:

- `amqsget myqueue qmanagername`
- `amqsgetc myqueue qmanagername`

gdzie `myqueue` to nazwa kolejki, z której program otrzyma komunikaty, a `qmanagername` to menedżer kolejek, który jest właścicielem produktu `myqueue`.

Jeśli parametr `qmanagername`zostanie pominięty, programy przyjmują wartość domyślną lub, w przypadku klienta MQI, menedżera kolejek identyfikowanego przez zmienną środowiskową lub plik definicji kanału klienta.

Przykładowe programy o wysokiej dostępności

Programy przykładowe o wysokiej dostępności w systemach **amqsghac**, **amqsphac**i **amqsmhac** używają zautomatyzowanego ponownego połączenia klienta w celu zademonstrowania odtwarzania po awarii menedżera kolejek. Program **amqsfhac** sprawdza, czy menedżer kolejek korzystający z sieciowej pamięci masowej zachowuje integralność danych po awarii.

Programy **amqsghac**, **amqsphac**i **amqsmhac** są uruchamiane z wiersza komend i mogą być używane w kombinacji w celu zademonstrowania ponownego połączenia po awarii jednej instancji menedżera kolejek z wieloma instancjami.

Alternatywnie można użyć przykładów **amqsghac**, **amqsphac**i **amqsmhac** , aby zademonstrować ponowne połączenie klienta z menedżerami kolejek z pojedynczą instancją, zwykle skonfigurowanymi w grupie menedżerów kolejek.

Aby zachować prosty przykład, co ułatwia konfigurowanie, wyświetlane są przykładowe programy łączące się ponownie z menedżerem kolejek z pojedynczą instancją, który jest uruchamiany, zatrzymywany, a następnie restartowany ponownie (patrz sekcja "Konfigurowanie menedżera kolejek i sterowanie nim" na stronie 136).

Aby sprawdzić integralność systemu plików, należy użyć opcji **amqsfhac** równolegle z opcją **amqmfsck** . Więcej informacji na ten temat zawiera sekcja **amqmfsck** (sprawdzanie systemu plików) i sekcja Weryfikowanie zachowania współużytkowanego systemu plików .

amqsphac ***queueName*** **[*qMgrNazwa*]**

- **amqsphac** jest aplikacją IBM WebSphere MQ MQI client . Umieszcza sekwencję komunikatów w kolejce z dwusekundowym opóźnieniem między każdym komunikatem i wyświetla zdarzenia wysłane do jego procedury obsługi zdarzeń.
- Żaden punkt synchronizacji nie jest używany do umieszczania komunikatów w kolejce.
- Można ponownie nawiązać połączenie z dowolnym menedżerem kolejek w tej samej grupie menedżerów kolejek.

amqsghac ***queueName*** **[*qMgrNazwa*]**

- **amqsghac** jest aplikacją IBM WebSphere MQ MQI client . Pobiera komunikaty z kolejki i wyświetla zdarzenia wysłane do procedury obsługi zdarzeń.
- Żaden punkt synchronizacji nie jest używany do pobierania komunikatów z kolejki.
- Można ponownie nawiązać połączenie z dowolnym menedżerem kolejek w tej samej grupie menedżerów kolejek.

amqsmhac -s ***sourceQueueNazwa*** **-t** ***targetQueueNazwa*** **[-m** ***qMgrNazwa*****] [-w** ***waitInterval*****]**

- **amqsmhac** jest aplikacją IBM WebSphere MQ MQI client . Kopiuje on komunikaty z jednej kolejki do drugiej z domyślnym odstępem czasu oczekiwania wynoszącym 15 minut po ostatnim odebranym komunikacie, zanim program zakończy działanie.
- Komunikaty są kopiowane w ramach punktu synchronizacji.
- Ponowne połączenie można nawiązać tylko z tym samym menedżerem kolejek.

amqsfhac ***QueueManagerName QueueName SideQueueName InTransactionCount RepeatCount*** **(0|1|2)**

- **amqsfhac** jest aplikacją IBM WebSphere MQ MQI client . Sprawdza on, czy menedżer kolejek z wieloma instancjami programu IBM WebSphere MQ korzystający z sieciowej pamięci masowej, takiej jak NAS lub klastrowy system plików, utrzymuje integralność danych. Wykonaj kroki opisane w sekcji Weryfikowanie zachowania współużytkowanego systemu plików , aby uruchomić komendę **amqsfhac** .
- Opcja `MQCNO_RECONNECT_Q_MGR` jest używana podczas nawiązywania połączenia z menedżerem kolejek *QueueManager*. Po przełączeniu menedżera kolejek następuje automatyczne ponowne nawiązanie połączenia.
- Umieszcza ona *InTransactionCount***RepeatCount* trwałe komunikaty w kolejce *QueueName* , podczas którego menedżer kolejek może przełączać się awaryjnie dowolną liczbę razy. Program **amqsfhac** ponownie nawiązuje połączenie z menedżerem kolejek za każdym razem i kontynuuje działanie. Test ma na celu upewnienie się, że żadne komunikaty nie zostaną utracone.
- *InTransactionCount* komunikaty są umieszczane w każdej transakcji. Transakcja jest powtarzana *RepeatCount* razy. Jeśli w ramach transakcji wystąpi niepowodzenie, program **amqsfhac** wycofa

zmiany i wprowadzi ponownie transakcję, gdy program **amqsfhac** ponownie nawiąże połączenie z menedżerem kolejek.

- Umieszcza również komunikaty w kolejce bocznikowania *SideQueueName*. Do sprawdzenia, czy wszystkie komunikaty zostały pomyślnie zatwierdzone lub wycofane z kolejki *QueueName* , używany jest parametr *SideQueueName* . Jeśli wykryje niespójność, zapisuje komunikat o błędzie.
- Zmiana poziomu śledzenia danych wyjściowych z **amqsfhac** przez ustawienie ostatniego parametru na `(0|1|2)`.

  0
  Najmniejszy wynik.

  1
  Wyjście pośredniczące.

  2
  Większość danych wyjściowych.

Konfigurowanie połączenia klienckiego

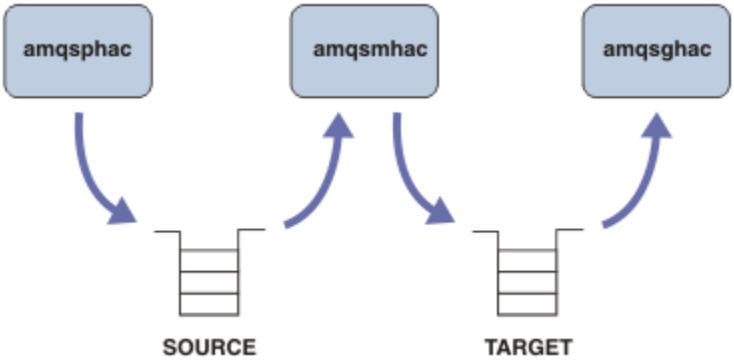
Aby uruchomić przykłady, należy skonfigurować kanał połączenia klienta i serwera. Procedura weryfikacji klienta wyjaśnia sposób konfigurowania środowiska testowego klienta. Patrz sekcja Weryfikowanie instalacji klienta.

Alternatywnie można użyć konfiguracji podanej w poniższym przykładzie.

Przykład użycia amqsghac, amqsphaci amqsmhac

W tym przykładzie przedstawiono klienty z możliwością ponownego nawiązania połączenia przy użyciu menedżera kolejek z pojedynczą instancją.

Komunikaty są umieszczane w kolejce SOURCE przez **amqsphac**, przesyłane do TARGET przez **amqsmhac**i pobierane z TARGET przez **amqsghac**; patrz Rysunek 19 na stronie 135.



Rysunek 19. Przykłady klienta z możliwością ponownego połączenia

Wykonaj poniższe kroki, aby uruchomić przykłady.

1. Utwórz plik `hasamples.tst` zawierający komendy:

```
DEFINE QLOCAL(SOURCE) REPLACE
DEFINE QLOCAL(TARGET) REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)   TRPTYPE(TCP) +
       MCAUSER(MUSR_MQADMIN) REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) +
       CONNAME('LOCALHOST(2345)') QMNAME(QM1) REPLACE
ALTER  LISTENER(SYSTEM.DEFAULT.LISTENER.TCP) TRPTYPE(TCP) +
       PORT(2345)
START  LISTENER(SYSTEM.DEFAULT.LISTENER.TCP)
START  CHANNEL(CHANNEL1)
```

2. Wpisz następujące komendy w wierszu komend:

a. `crtmqm QM1`

b. `strmqm QM1`

c. `runmqsc QM1 < hasamples.tst`

3. Ustaw zmienną środowiskową **MQCHLLIB** na ścieżkę do pliku definicji kanału klienta `AMQCLCHL.TAB` , na przykład `SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc`.
4. Otwórz trzy nowe okna z ustawionym **MQCHLLIB** ; na przykład w systemie Windowswpisz **start** trzy razy w poprzednim wierszu komend, uruchamiając każdy program w jednym z okien. Patrz krok “5” na stronie 137 w sekcji “Konfigurowanie menedżera kolejek i sterowanie nim” na stronie 136).
5. Wpisz komendę `endmqm -r -p QM1` , aby zatrzymać menedżer kolejek, a następnie zezwolić klientom na ponowne nawiązanie połączenia.
6. Wpisz komendę `strmqm QM1` , aby zrestartować menedżer kolejek.

Wyniki uruchomienia przykładów **amqsghac**, **amqsphac**i **amqsmhac** w systemie Windows zostały przedstawione w poniższych przykładach.

Konfigurowanie menedżera kolejek i sterowanie nim

1. Utwórz menedżer kolejek.

```
C:\>crtmqm QM1
WebSphere MQ queue manager created.
Directory 'C:\IBM\MQ\MQ7\Data\qmgrs\QM1' created.
Creating or replacing default objects for QM1.
Default objects statistics : 67 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

   Zapamiętaj katalog danych, aby później ustawić zmienną **MQCHLLIB** .

2. Uruchom menedżer kolejek.

```
C:\>strmqm QM1
WebSphere MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
WebSphere MQ queue manager 'QM1' started.
```

3. Utwórz kolejki i kanały, zmodyfikuj port nasłuchiwania oraz uruchom program nasłuchujący i kanał.

```
C:\>runmqsc QM1 < hasamples.tst
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.

     1 : DEFINE QLOCAL(SOURCE) REPLACE
AMQ8006: WebSphere MQ queue created.
     2 : DEFINE QLOCAL(TARGET) REPLACE
AMQ8006: WebSphere MQ queue created.
     3 : DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)   TRPTYPE(TCP) MCAUSER(MUSR_MQADMIN)
REPLACE
AMQ8014: WebSphere MQ channel created.
     4 : DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) CONNAME('LOCALHOST(2345)')
QMNAME(QM1) REPLACE
AMQ8014: WebSphere MQ channel created.
     5 : ALTER LISTENER(SYSTEM.DEFAULT.LISTENER.TCP) TRPTYPE(TCP) PORT(2345)
AMQ8623: WebSphere MQ listener changed.
     6 : START LISTENER(SYSTEM.DEFAULT.LISTENER.TCP)
AMQ8021: Request to start WebSphere MQ Listener accepted.
     7 : START CHANNEL(CHANNEL1)
AMQ8018: Start WebSphere MQ channel accepted.
7 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

4. Udostępnij klientom tabelę kanałów klienta.

Użyj katalogu danych zwróconego przez komendę **crtmqm** w kroku “1” na stronie 136i dodaj do niego katalog `@ipcc` , aby ustawić zmienną **MQCHLLIB** .

```
C:\>SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc
```

5. Uruchamianie przykładowych programów w innych oknach

```
C:\>start amqsphac SOURCE QM1
C:\>start amqsmhac -s SOURCE -t TARGET -m QM1
C:\>start amqsghac TARGET QM1
```

6. Zakończ działanie menedżera kolejek i zrestartuj go ponownie.

```
C:\>endmqm -r -p QM1
Waiting for queue manager 'QM1' to end.
WebSphere MQ queue manager 'QM1' ending.
WebSphere MQ queue manager 'QM1' ended.

C:\>strmqm QM1
WebSphere MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
WebSphere MQ queue manager 'QM1' started.
```

amqsphac

```
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
message <Message 2>
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnectedmessage
<Message 3>
message <Message 4>
message <Message 5>
```

amqsmhac

```
Sample AMQSMHA0 start
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
No more messages.
Sample AMQSMHA0 end
C:\>
```

amqsghac

```
Sample AMQSGHAC start
message <Message 1>
message <Message 2>
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
message <Message 3>
message <Message 4>
message <Message 5>
```

Zadania pokrewne

Weryfikowanie zachowania współużytkowanego systemu plików

Odsyłacze pokrewne

amqmfsck (sprawdzanie systemu plików)

Przykładowe programy Inquire

Przykładowe programy Inquire pytają o niektóre atrybuty kolejki przy użyciu wywołania MQINQ.

Nazwy tych programów można znaleźć w sekcji "Funkcje demonstrowe w przykładowych programach" na stronie 100 .

Programy te są przeznaczone do uruchamiania jako programy wyzwalane, więc ich jedynym wejściem jest struktura MQTMC2 (komunikat wyzwalacza) dla systemów IBM i, Windowsi UNIX and Linux . Ta struktura zawiera nazwę kolejki docelowej z atrybutami, które mają zostać zapytane. W wersji C używana jest również nazwa menedżera kolejek. W wersji COBOL używany jest domyślny menedżer kolejek.

Aby proces wyzwalający działał, należy upewnić się, że przykładowy program Inquire, który ma być używany, jest wyzwalany przez komunikaty przychodzące do kolejki SYSTEM.SAMPLE.INQ. W tym celu należy podać nazwę przykładowego programu Inquire, który ma być używany w polu *ApplicId* definicji procesu SYSTEM.SAMPLE.INQPROCESS. Kolejka przykładowa ma typ wyzwalacza FIRST; jeśli przed uruchomieniem przykładu żądania znajdują się już komunikaty w kolejce, próbka nie jest wyzwalana przez wysyłane komunikaty.

Jeśli definicja została ustawiona poprawnie:

- W przypadku systemów UNIX, Linux i Windows uruchom program **runmqtrm** w jednej sesji, a następnie uruchom program `amqsreq` w innej sesji.

Za pomocą przykładowych programów żądania można wysyłać komunikaty żądań, z których każdy zawiera tylko nazwę kolejki, do kolejki SYSTEM.SAMPLE.INQ. W przypadku każdego komunikatu żądania programy przykładowe Inquire wysyłają komunikat odpowiedzi zawierający informacje o kolejce określonej w komunikacie żądania. Odpowiedzi są wysyłane do kolejki odpowiedzi określonej w komunikacie żądania.

Projekt przykładowego programu Inquire

Program otwiera kolejkę nazwaną w strukturze komunikatu wyzwalacza, która została przekazana podczas jej uruchamiania. (Dla jasności zadzwonimy do tej *kolejki żądań*.) Program korzysta z wywołania MQOPEN, aby otworzyć tę kolejkę dla współużytkowanych danych wejściowych.

Program korzysta z wywołania MQGET w celu usunięcia komunikatów z tej kolejki. To wywołanie używa opcji MQGMO_ACCEPT_TRUNCATED_MSG i MQGMO_WAIT, z interwałem oczekiwania 5 sekund. Program testuje deskryptor każdego komunikatu, aby sprawdzić, czy jest to komunikat żądania. Jeśli tak nie jest, program usuwa komunikat i wyświetla komunikat ostrzegawczy.

Dla każdego komunikatu żądania usuniętego z kolejki żądań program odczytuje nazwę kolejki (którą nazywamy *kolejką docelową*) zawartą w danych, a następnie otwiera tę kolejkę przy użyciu wywołania MQOPEN z opcją MQOO_INQ. Następnie program korzysta z wywołania MQINQ w celu sprawdzenia wartości atrybutów *InhibitGet*, *CurrentQDepth*i *OpenInputCount* w kolejce docelowej.

Jeśli wywołanie MQINQ powiedzie się, program korzysta z wywołania MQPUT1 w celu umieszczenia komunikatu odpowiedzi w kolejce odpowiedzi. Ten komunikat zawiera wartości trzech atrybutów.

Jeśli wywołanie MQOPEN lub MQINQ nie powiodło się, program korzysta z wywołania MQPUT1 w celu umieszczenia komunikatu raportu w kolejce odpowiedzi. W polu *Feedback* deskryptora komunikatu tego komunikatu raportu jest to kod przyczyny zwracany przez wywołanie MQOPEN lub MQINQ, w zależności od tego, który błąd nie powiódł się.

Po wywołaniu wywołania MQINQ program zamknie kolejkę docelową przy użyciu wywołania MQCLOSE.

Jeśli w kolejce żądań nie pozostały żadne komunikaty, program zamknie tę kolejkę i rozłącza się z menedżerem kolejek.

Właściwości Inquire przykładowego programu Message Handle

AMQSIQMA to przykładowy program w języku C do sprawdzania właściwości uchwytu komunikatu z kolejki komunikatów. Jest to przykład użycia wywołania funkcji API MQINQMP.

W tym przykładzie tworzony jest uchwyt komunikatu i umieszcza go w polu MsgHandle w strukturze MQGMO. Następnie próbka pobiera jeden komunikat i pobiera i drukuje wszystkie właściwości, z którymi uchwyt komunikatu został zapełniony.

```
C:\Program Files\IBM\WebSphere MQ\tools\c\Samples\Bin >amqsiqm Q QM1
Sample AMQSIQMA start
property name <MyProp> value <MyValue>
message text <Hello world!>
Sample AMQSIQMA end
```

Programy przykładowe publikowania/subskrypcji

Przykładowe programy publikowania/subskrypcji demonstrują sposób użycia funkcji publikowania i subskrypcji w produkcie WebSphere MQ.

Istnieją trzy przykładowe programy w języku C ilustrujące sposób programu do interfejsu publikowania/subskrybowania produktu WebSphere MQ . Istnieją przykłady języka C, które korzystają ze starszych interfejsów, a są to przykłady Java. Przykłady Java używają interfejsu publikowania/subskrybowania produktu WebSphere MQ w pliku com.ibm.mq.jar i interfejsu publikowania/subskrypcji JMS w klasie com.ibm.mqjms. Przykłady JMS nie są uwzględnione w tym temacie.

C

Znajdź przykładowy publikator `amqspub` w folderze przykładów produktu C . Uruchom go z dowolną nazwą tematu, którą lubisz jako pierwszy, a następnie opcjonalną nazwą menedżera kolejek. Na przykład: `amqspub mytopic QM3` . Istnieje również wersja klienta o nazwie `amqspubc`. Jeśli zostanie wybrana opcja uruchamiania wersji klienta, należy najpierw zapoznać się z informacjami szczegółowymi “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113 .

Publikator łączy się z domyślnym menedżerem kolejek i odpowiada na dane wyjściowe, `target topic is mytopic` . Każdy wiersz wprowadzany do tego okna od teraz jest publikowany w produkcie `mytopic`.

Otwórz inne okno komend w tym samym katalogu, a następnie uruchom program subskrybenta `amqssub`, podając tę samą nazwę tematu i opcjonalną nazwę menedżera kolejek. Na przykład: `amqssub mytopic QM3`.

Subskrybent odpowiada za pomocą danych wyjściowych `Calling MQGET : 30 seconds wait time`. Od tej pory w danych wyjściowych subskrybenta pojawiają się wiersze, które wpisujesz do publikatora.

Uruchom innego subskrybenta w innym oknie komend i obserwuj oba subskrybenty, które otrzymują publikacje.

Pełną dokumentację parametrów, w tym opcje ustawień, można znaleźć w przykładowym kodzie źródłowym. Wartości w polu opcji subskrybenta zostały opisane w następującym temacie: Opcje (MQLONG).

Istnieje inny przykładowy subskrybent `amqssbx`, który oferuje dodatkowe opcje subskrypcji jako przełączniki wiersza komend.

Wpisz `amqssbx -d mysub -t mytopic -k` , aby wywołać subskrybenta przy użyciu trwałych subskrypcji, które są zachowywane po zakończeniu działania subskrybenta.

Przetestuj subskrypcję, publikując inny element przy użyciu publikatora. Poczekaj 30 sekund na zakończenie działania subskrybenta. Opublikuj kilka elementów w tym samym temacie. Zrestartuj subskrybenta. Ostatni element opublikowany w czasie, gdy subskrybent nie był uruchomiony, jest wyświetlany przez subskrybent natychmiast po jego zrestartowaniu.

Wcześniejsza wersja

Istnieje dodatkowy zestaw przykładów języka C, które demonstrują komendy w kolejce. Niektóre z tych przykładów zostały oryginalnie wysłane jako część pakietu MQ0C Supportpac. Możliwości demonstrować są w pełni obsługiwane, ze względu na kompatybilność.

Zniechęcamy Cię do korzystania z interfejsu komend w kolejce. Jest on znacznie bardziej złożony niż interfejs API publikowania/subskrypcji i nie ma przekonujących powodów funkcjonalnych do programowania złożonych komend w kolejce. Jednak podejście w kolejce może być bardziej odpowiednie, być może dlatego, że korzystasz już z interfejsu, lub ponieważ środowisko programistyczne ułatwia budowanie złożonego komunikatu i wywoła ogólną operację MQPUT, a nie konstruowanie różnych wywołań MQSUB.

Dodatkowe przykłady znajdują się w podkatalogu pubsub w folderze `samples` .

W produkcie Tabela 20 na stronie 140znajduje się sześć typów przykładów.

Tabela 20. Kategorie wcześniejszych przykładowych programów w języku C publikowania/subskrypcji

Kategoria	Programy	Komentarze
RFH1	`amqssr1a.c` `amqspr1a.c`	Prosty przykład publikowania/subskrypcji zbudowany przy użyciu komunikatów w formacie RFH1 .
RFH2	`amqssr2a.c` `amqspr2a.c`	Prosty przykład publikowania/subskrypcji zbudowany przy użyciu komunikatów w formacie RFH2 .
Przykłady MQAI	`amqsppca.c` `amqsspca.c`	Prosty przykład publikowania/subskrypcji zbudowany za pomocą komend PCF i interfejsu komend MQAI.
MA0C Wyniki usługi przy użyciu RFH1	`amqsgama.c` `amqsresa.c`	Usługa wyników zbudowana przy użyciu nagłówków RFH1 1. Wymaga kolejek zdefiniowanych w `amqsgama.tst` i `amqsresa.tst` 2. Produkt `amqsresa` musi być uruchomiony przed `amqsgama`
MA0C Wyniki usługi przy użyciu RFH2	`amqsgr2a.c` `amqsrr2a.c`	Usługa wyników zbudowana przy użyciu nagłówków RFH2 1. Wymaga kolejek zdefiniowanych w `amqsgama.tst` i `amqsresa.tst` 2. Produkt `amqsresa` musi być uruchomiony przed `amqsgama`
Przykład publikowania/ subskrybowania wyjścia routingu	`amqspsra.c`	Demonstruje, jak zmienić miejsce docelowe kolejki lub menedżera kolejek dla komunikatu publikowania/subskrypcji w wyjściu routingu.

Java

Przykładowy język Java `MQPubSubApiSample.java` łączy publikatora i subskrybentów w jednym programie. Jego źródłowe i skompilowane pliki klas znajdują się w folderze przykładów produktu `wmqjava` .

Jeśli zostanie wybrana opcja uruchamiania w trybie klienta, należy najpierw zapoznać się z informacjami szczegółowymi “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113 .

Uruchom przykład z wiersza komend za pomocą komendy Java, jeśli jest skonfigurowane środowisko Java. Można również uruchomić przykład z poziomu obszaru roboczego WebSphere MQ Explorer Eclipse , który zawiera już skonfigurowane środowisko robocze programistyczne Java.

W celu jego uruchomienia może być konieczna zmiana niektórych właściwości programu przykładowego. Można to zrobić, podając parametry dla maszyny JVM lub edytując źródło.

Instrukcje zawarte w sekcji “Uruchamianie przykładu Java MQPubSubApiSample” na stronie 141 przedstawiają sposób uruchamiania przykładu z obszaru roboczego Eclipse .

Uruchamianie przykładu Java MQPubSubApiSample

Jak uruchomić MQPubSubApiSample przy użyciu narzędzi programistycznych Java z poziomu platformy Eclipse .

Zanim rozpoczniesz

Otwórz środowisko robocze Eclipse . Utwórz nowy katalog obszaru roboczego i wybierz go. Zamknij okno powitalne.

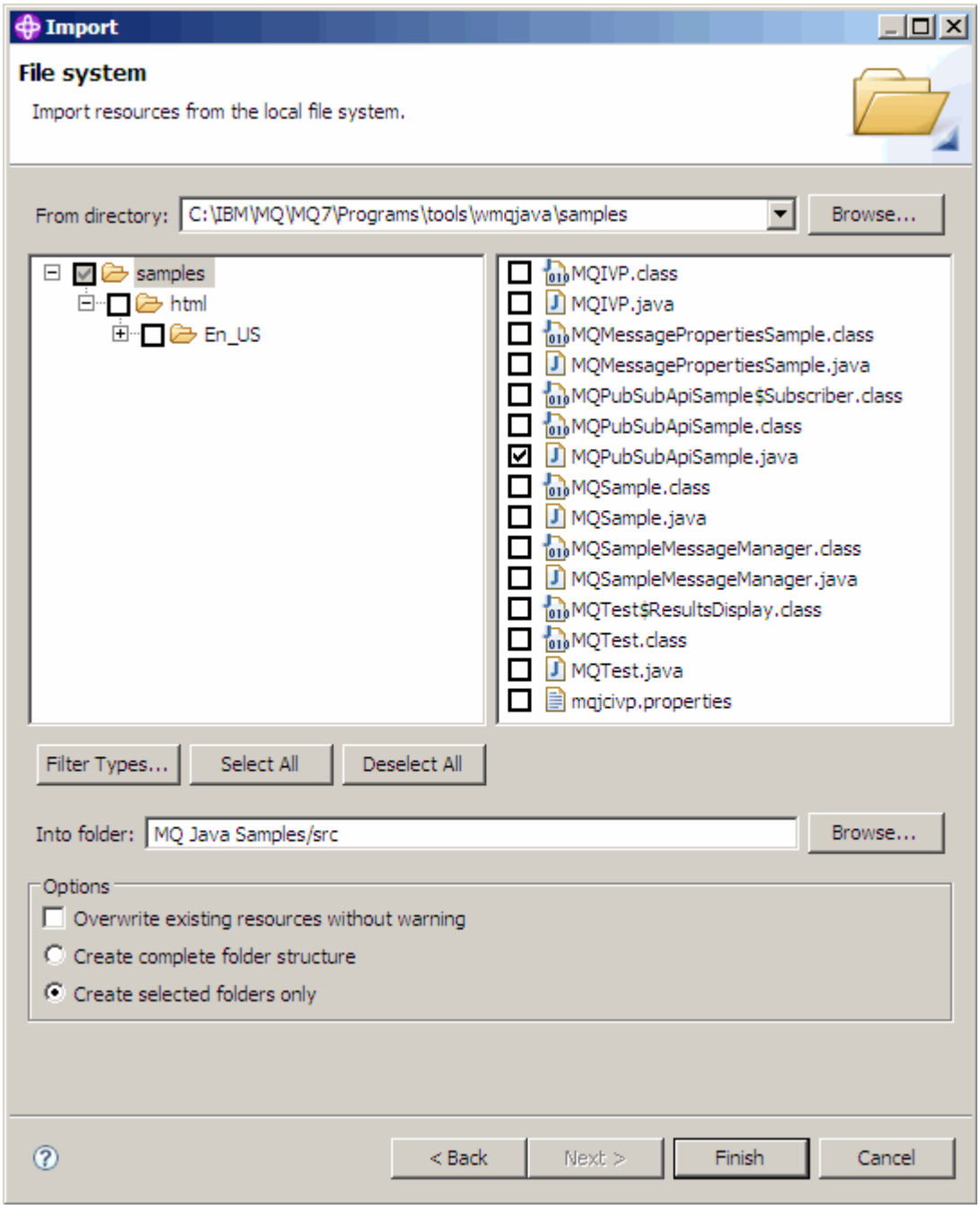
Wykonaj kroki opisane w sekcji “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113 przed uruchomieniem jako klient.

O tym zadaniu

Przykładowy program Java publish/subscribe to program Java klienta MQI produktu WebSphere MQ . Przykład jest uruchamiany bez modyfikacji przy użyciu domyślnego menedżera kolejek nasłuchiwania na porcie 1414. Zadanie opisuje ten prosty przypadek i zawiera ogólne informacje na temat sposobu udostępniania parametrów i modyfikowania przykładu w celu dopasowania do różnych konfiguracji produktu WebSphere MQ . Przykład ten jest ilustrowany w systemie Windows. Ścieżki do plików będą się różnić na innych platformach.

Procedura

1. Importowanie przykładowych programów Java
   a) W środowisku roboczym kliknij opcję **Okna** > **Otwórz perspektywę** > **Inne** > **Java** , a następnie kliknij przycisk **OK**.
   b) Przejdź do widoku **Eksplorator pakietów** .
   c) Kliknij prawym przyciskiem myszy białą przestrzeń w widoku **Eksplorator pakietów** . Kliknij opcję **Nowy** > **Projekt Java**.
   d) W polu **`Project name`** wpisz `MQ Java Samples`. Kliknij przycisk **Dalej**.
   e) Na panelu **`Java Settings`** przejdź do karty **Libraries** (Biblioteki).
   f) Kliknij opcję **Dodaj zewnętrzne pliki JAR**.
   g) Przejdź do katalogu *MQ_INSTALLATION_PATH*`\java\lib` , gdzie *MQ_INSTALLATION_PATH* jest folderem instalacyjnym produktu WebSphere MQ , a następnie wybierz opcję `com.ibm.mq.jar` i `com.ibm.mq.jmqi.jar` .
   h) Kliknij opcję **Otwórz** > **Zakończ.**
   i) Kliknij prawym przyciskiem myszy `src` w widoku **Eksplorator pakietów** .
   j) Wybierz opcję **Importuj ...** > **Ogólne** > **System plików** > **Dalej** > **Przeglądaj**... i przejdź do ścieżki *MQ_INSTALLATION_PATH*`\tools\wmqjava\samples` , gdzie *MQ_INSTALLATION_PATH* jest katalogiem instalacyjnym produktu WebSphere MQ .
   k) Na panelu **Import** (Import), Rysunek 20 na stronie 142, kliknij przycisk `samples` (nie zaznaczaj pola wyboru).
   l) Wybierz opcję `MQPubSubApiSample.java`. Pole **`Into folder`** powinno zawierać `MQ Java Samples/src`. Kliknij przycisk **Zakończ**.



Rysunek 20. Import systemu plików

2. Uruchom przykładowy program publikowania/subskrypcji.

   Istnieją dwa sposoby uruchamiania programu w zależności od tego, czy konieczna jest zmiana parametrów domyślnych.

   - Pierwszy wybór uruchamia program bez dokonywania żadnych zmian:
     - W menu głównym obszaru roboczego rozwiń folder `src` (Menu główne). Kliknij **MQPubSubApiSample.java** przyciskiem**Uruchom jako** > **1. Aplikacja Java**
   - Drugi wybór uruchamia program z parametrami lub z zmodyfikowanym kodem źródłowym dla Twojego środowiska:
     - Otwórz program `MQPubSubApiSample.java` i przestudiuj konstruktor `MQPubSubApiSample` .

– Zmodyfikuj atrybuty programu.

  Te atrybuty są modyfikowalne za pomocą przełącznika -D maszyny JVM lub przez podanie wartości domyślnej dla właściwości System, edytując kod źródłowy.

  - topicObject
  - QueueManagerName
  - subscriberCount

  Te atrybuty mogą być zmieniane tylko przez edycję kodu źródłowego w konstruktorze.

  - nazwa hosta
  - Port
  - kanał

  Aby ustawić właściwości systemowe, należy zakodować wartość domyślną w akcesorze, na przykład:

```
queueManagerName = System.getProperty("com.ibm.mq.pubSubSample.queueManagerName",
"QM3");
```

  Lub podaj parametr dla maszyny JVM za pomocą opcji `-D` , tak jak pokazano to w poniższych krokach:

a. Skopiuj pełną nazwę pliku System.Property , który ma zostać ustawiony, na przykład: `com.ibm.mq.pubSubSample.queueManagerName`.
b. W obszarze roboczym kliknij prawym przyciskiem myszy opcję **Uruchom** > **Otwórz okno dialogowe uruchamiania**. Kliknij dwukrotnie aplikację Java w obszarze **Utwórz, zarządzaj i uruchom aplikacje** , a następnie kliknij kartę **(x) = argumenty** .
c. W panelu **Argumenty maszyny VM:** wpisz `-D` i wklej nazwę System.property , `com.ibm.mq.pubSubSample.queueManagerName`, a następnie =QM3. Kliknij kolejno opcje **Zastosuj** > **Uruchom**.
d. Dodaj kolejne argumenty jako listę rozdzielaną przecinkami lub jako dodatkowe wiersze w panelu bez separatorów przecinków.

   Na przykład:`-Dcom.ibm.mq.pubSubSample.queueManagerName=QM3`, `-Dcom.ibm.mq.pubSubSample.subscriberCount=6`.

Przykładowy program wyjścia publikowania

AMQSPSE0 to przykładowy program w języku C wyjścia w celu przechwycenia publikacji, zanim zostanie dostarczony do subskrybenta. Następnie wyjście może na przykład zmienić nagłówki komunikatu, ładunek lub miejsce docelowe albo uniemożliwić opublikowanie komunikatu w subskrybencie.

Aby uruchomić przykład, wykonaj następujące czynności:

1. Skonfiguruj menedżer kolejek:
   - W systemach UNIX and Linux dodaj sekcję taką, jak ta, do pliku `qm.ini` :

```
PublishSubscribe:
           PublishExitPath=<Module>
           PublishExitFunction=EntryPoint
```

   gdzie moduł jest *MQ_INSTALLATION_PATH*`/samp/bin/amqspse` . *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ . W systemie Windows ustaw równoważne atrybuty w rejestrze.
2. Upewnij się, że moduł jest dostępny dla produktu WebSphere MQ.
3. Zrestartuj menedżer kolejek, aby odebrać konfigurację.
4. W procesie aplikacji, który ma być śledzony, należy opisać miejsce, w którym powinny być zapisywane pliki śledzenia. Na przykład:

- W systemach UNIX and Linux upewnij się, że katalog/`var/mqm/trace` istnieje i wyeksportuj następującą zmienną środowiskową:

```
export MQPSE_TRACE_LOGFILE=/var/mqm/trace/PubTrace
```

- W systemie Windows upewnij się, że katalog `C:\temp` istnieje, a następnie ustaw następującą zmienną środowiskową:

```
set MQPSE_TRACE_LOGFILE=C:\temp\PubTrace
```

Przykładowe programy umieszczania

Programy przykładowe umieszczone w kolejce umieszczają komunikaty w kolejce przy użyciu wywołania MQPUT.

Nazwy tych programów można znaleźć w sekcji "Funkcje demonstrowe w przykładowych programach" na stronie 100 .

Projekt przykładowego programu "Put"

Program korzysta z wywołania MQOPEN z opcją MQOO_OUTPUT, aby otworzyć kolejkę docelową w celu umieszczania komunikatów.

Jeśli nie jest możliwe otwarcie kolejki, program wyświetli komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN. Aby program był prosty, w tym i w kolejnych wywołaniach MQI program używa wartości domyślnych dla wielu opcji.

Dla każdego wiersza danych wejściowych program odczytuje tekst do buforu i korzysta z wywołania MQPUT w celu utworzenia komunikatu datagramu zawierającego tekst tego wiersza. Program będzie kontynuowany aż do momentu, gdy dojdzie do końca danych wejściowych lub wywołanie MQPUT nie powiedzie się. Jeśli program dociera do końca danych wejściowych, zamyka kolejkę za pomocą wywołania MQCLOSE.

Uruchamianie przykładowych programów umieszczania

Uruchamianie próbek amqsput i amqsputc

Każdy z tych programów ma 2 parametry:

1. Nazwa kolejki docelowej (wymagane)
2. Nazwa menedżera kolejek (opcjonalnie)

Jeśli menedżer kolejek nie jest określony, amqsput łączy się z domyślnym menedżerem kolejek, a amqsputc łączy się z menedżerem kolejek identyfikowany przez zmienną środowiskową lub plik definicji kanału klienta. Aby uruchomić te programy, wprowadź jedną z następujących opcji:

- `amqsput myqueue qmanagername`
- `amqsputc myqueue qmanagername`

gdzie `myqueue` jest nazwą kolejki, w której mają zostać umieszczone komunikaty, a `qmanagername` jest menedżerem kolejek, który jest właścicielem produktu `myqueue`.

Uruchamianie przykładu amq0put

Wersja COBOL nie ma żadnych parametrów. Jest on połączony z domyślnym menedżerem kolejek i po uruchomieniu zostanie wyświetlony monit:

```
Please enter the name of the target queue
```

Pobiera ona dane wejściowe z elementu StdIn i dodaje każdy wiersz danych wejściowych do kolejki docelowej. Pusty wiersz wskazuje, że nie ma więcej danych.

Przykładowe programy Message Message

Przykłady komunikatów referencyjnych umożliwiają przesyłanie dużego obiektu z jednego węzła do innego (zwykle w różnych systemach) bez konieczności przechowywania obiektu w kolejkach WebSphere MQ w węzłach źródłowych lub docelowych.

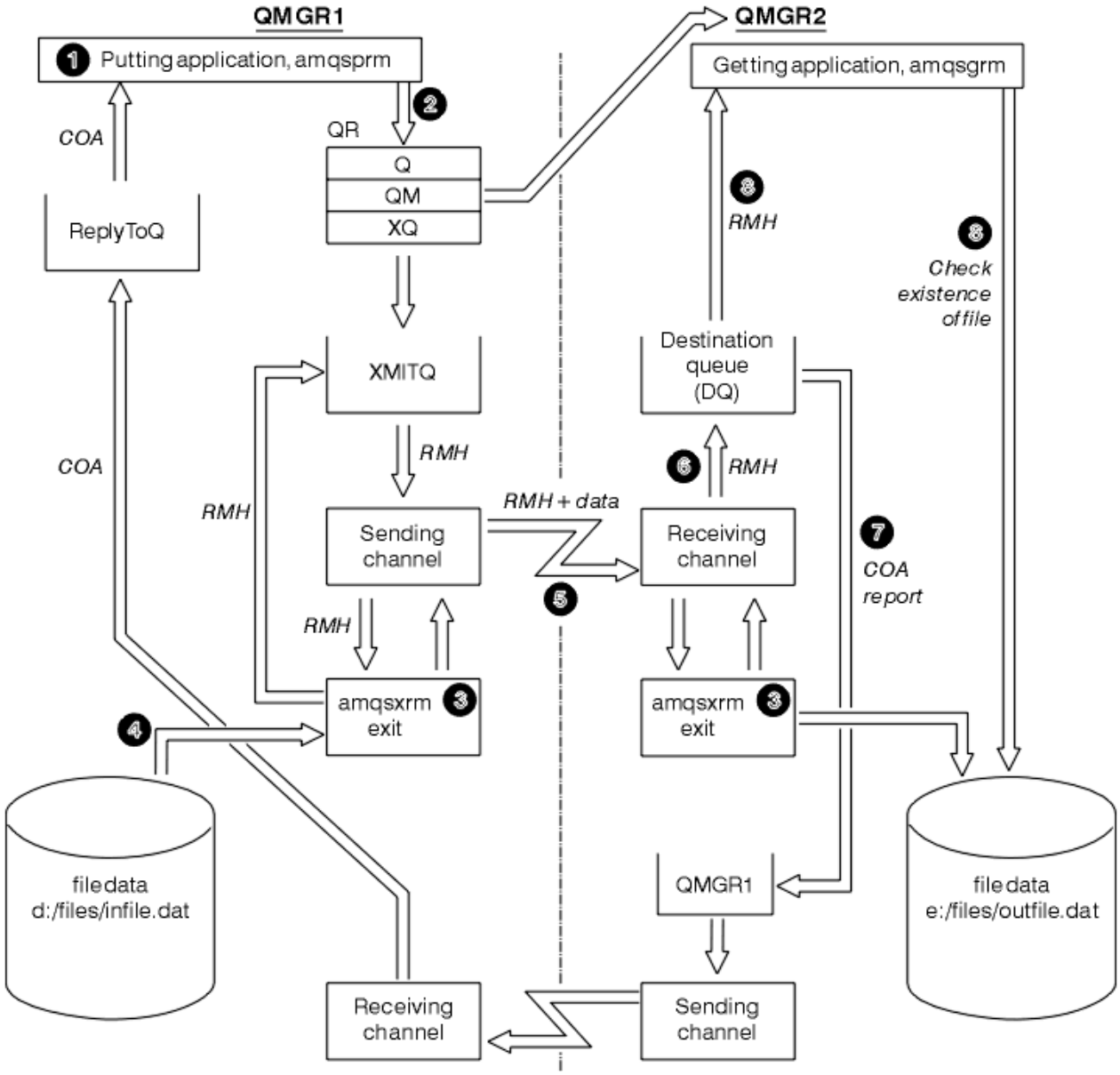
Udostępniono zestaw przykładowych programów w celu zademonstrować, w jaki sposób komunikaty odniesienia mogą być umieszczane w kolejce, odbierane przez wyjścia komunikatów i pobierane z kolejki. Programy przykładowe korzystają z komunikatów referencyjnych do przenoszenia plików. Jeśli chcesz przenieść inne obiekty, takie jak bazy danych, lub jeśli chcesz wykonać sprawdzenia bezpieczeństwa, zdefiniuj własne wyjście, na podstawie naszej próbki, amqsxrm. W poniższych sekcjach opisano programy przykładowe komunikatów odniesienia.

Wersja przykładowego programu obsługi wyjścia komunikatów odwołania, która ma być używana, zależy od platformy, na której działa kanał. Na wszystkich platformach użyj amqsxrma na końcu wysyłania. Jeśli odbiornik jest uruchomiony w dowolnym produkcie WebSphere MQ z wyjątkiem WebSphere MQ for IBM i, użyj komendy amqsxrm4 .

Uruchamianie przykładów komunikatów odniesienia

Informacje zawarte w tej sekcji umożliwiają zapoznanie się z informacjami na temat uruchamiania programów przykładowych komunikatów referencyjnych.

Przykłady komunikatów odniesienia są uruchamiane w następujący sposób:



Rysunek 21. Uruchamianie przykładów komunikatów odniesienia

1. Skonfiguruj środowisko, aby uruchomić programy nasłuchujące, kanały i monitory wyzwalacza, a także zdefiniuj kanały i kolejki.

   Aby opisać, w jaki sposób należy skonfigurować przykład komunikatu referencyjnego, odnosi się on do maszyny wysyłającej jako MACHINE1 z menedżerem kolejek o nazwie QMGR1 i komputerem odbierającym jako MACHINE2 z menedżerem kolejek o nazwie QMGR2.

   Uwaga: Następujące definicje umożliwiają budowanie komunikatu referencyjnego w celu wysłania pliku z typem obiektu FLATFILE z menedżera kolejek QMGR1 do QMGR2 i ponownego utworzenia pliku zgodnie z definicją w wywołaniu AMQSPRM (lub AMQSPRMA w systemie IBM i). Komunikat referencyjny (wraz z danymi pliku) jest wysyłany przy użyciu kanału CHL1 i kolejki transmisji XMITQ i umieszczany w kolejce DQ. Raporty o wyjątkach i COA są wysyłane z powrotem do QMGR1 za pomocą kanału REPORT i kolejki transmisji QMGR1.

   Aplikacja, która odbiera komunikat Reference (AMQSGRM) jest wyzwalana przy użyciu kolejki inicjuj INITQ i procesu PROC. Upewnij się, że pola CONNAME są ustawione poprawnie, a pole MSGEXIT odzwierciedla strukturę katalogów, w zależności od typu komputera i miejsca, w którym zainstalowano produkt WebSphere MQ .

W definicjach MQSC używane są style AIX służące do definiowania wyjść. Należy pamiętać, że w danych komunikatu FLATFILE jest rozróżniana wielkość liter, a próba nie będzie działać, jeśli nie jest zapisana wielkimi literami.

Na komputerze MACHINE1, menedżer kolejek QMGR1

Składnia MQSC

```
define chl(chl1) chltype(sdr) trptype(tcp) conname('machine2') xmitq(xmitq)
msgdata(FLATFILE) msgexit('/usr/lpp/mqm/samp/bin/amqsxrm(MsgExit)
')

define ql(xmitq) usage(xmitq)

define chl(report) chltype(rcvr) trptype(tcp) replace

define qr(qr) rname(dq) rqmname(qmgr2) xmitq(xmitq) replace
```

Uwaga: Jeśli nazwa menedżera kolejek nie zostanie określona, system użyje domyślnego menedżera kolejek.

```
CRTMQMCHL  CHLNAME(CHL1) CHLTYPE(*SDR) MQMNAME(QMGR1) +
             REPLACE(*YES) TRPTYPE(*TCP) +
             CONNAME('MACHINE2(60501)') TMQNAME(XMITQ) +
             MSGEXIT(QMQM/AMQSXRM4) MSGUSRDATA(FLATFILE)

CRTMQMQ    QNAME(XMITQ) QTYPE(*LCL) MQMNAME(QMGR1) +
             REPLACE(*YES) USAGE(*TMQ)

CRTMQMCHL  CHLNAME(REPORT) CHLTYPE(*RCVR) +
               MQMNAME(QMGR1) REPLACE(*YES) TRPTYPE(*TCP)

CRTMQMQ    QNAME(QR) QTYPE(*RMT) MQMNAME(QMGR1) +
             REPLACE(*YES) RMTQNAME(DQ) +
             RMTMQMNAME(QMGR2) TMQNAME(XMITQ)
```

Na komputerze MACHINE2, menedżer kolejek QMGR2

Składnia MQSC

```
define chl(chl1) chltype(rcvr) trptype(tcp)
msgexit('/usr/lpp/mqm/samp/bin/amqsxrm(MsgExit)')
        msgdata(flatfile)

define chl(report) chltype(sdr) trptype(tcp) conname('MACHINE1')
        xmitq(qmgr1)

define ql(initq)

define ql(qmgr1) usage(xmitq)

define pro(proc) applicid('/usr/lpp/mqm/samp/bin/amqsgrm')

define ql(dq) initq(initq) process(proc) trigger trigtype(first)
```

2. Po utworzeniu obiektów produktu WebSphere MQ :
   a. Jeśli ma to zastosowanie do platformy, uruchom nasłuchiwanie dla menedżerów kolejek wysyłających i odbierających
   b. Uruchom kanały CHL1 i REPORT
   c. W odbiorczym menedżerze kolejek uruchom monitor wyzwalacza dla kolejki inicjującej INITQ.
3. W wierszu komend wywołaj przykładowy program put Reference Message AMQSPRM , korzystając z następujących parametrów:

   -m Nazwa lokalnego menedżera kolejek. Domyślnie jest to domyślny menedżer kolejek.

   -i Nazwa i położenie pliku źródłowego

   -o Nazwa i położenie pliku docelowego

-q Nazwa kolejki

-g Nazwa menedżera kolejek, w którym kolejka, zdefiniowana w parametrze -q, istnieje jako wartość domyślna dla menedżera kolejek określonego w parametrze -m

-t Typ obiektu

-w Interwał oczekiwania, czyli czas oczekiwania na zgłoszenia wyjątków i COA z menedżera kolejek odbierających

Na przykład, aby użyć przykładu z obiektami zdefiniowanymi wcześniej, należy użyć następujących parametrów:

```
-mQMGR1 -iInput File -oOutput File -qQR -tFLATFILE -w120
```

Wydłużanie czasu oczekiwania pozwala na wysłanie dużego pliku w sieci przed wyświetleniem limitu czasu przez program.

```
amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
```

Uwaga: W przypadku platform UNIX and Linux należy użyć dwóch ukośników odwrotnych (\\) zamiast jednego, aby oznaczyć docelowy katalog plików. Oznacza to, że komenda **amqsprm** wygląda następująco:

```
amqsprm -i /files/infile.dat -o e:\\files\\outfile.dat -q QR
-m QMGR1 -w 30 -t FLATFILE
```

Uruchomienie programu put Message Message powoduje, że:

- Komunikat odniesienia jest umieszczany w kolejce QR w menedżerze kolejek QMGR1.
- Plik źródłowy i ścieżka są `d:\files\infile.dat` i istnieją w systemie, w którym została wydana przykładowa komenda.
- Jeśli kolejka QR jest kolejką zdalną, komunikat odniesienia jest wysyłany do innego menedżera kolejek w innym systemie, w którym tworzony jest plik o nazwie i ścieżce `e:\files\outfile.dat`. Zawartość tego pliku jest taka sama, jak w pliku źródłowym.
- amqsprm czeka przez 30 sekund na raport COA z docelowego menedżera kolejek.
- Typ obiektu to `flatfile`, więc kanał używany do przenoszenia komunikatów z kolejki QR musi określać to pole w polu *MsgData* .

4. Podczas definiowania kanałów należy wybrać wyjście komunikatów zarówno w wysyłającym, jak i odbierającym końcach, aby był on amqsxrm. Jest to zdefiniowane w produkcie WebSphere MQ dla produktu Windows w następujący sposób:

   ```
   msgexit('pathname\amqsxrm.dll(MsgExit)')
   ```

   Jest to zdefiniowane w produkcie WebSphere MQ for AIX, WebSphere MQ for HP-UXi WebSphere MQ dla systemu Solaris w następujący sposób:

   ```
   msgexit('pathname/amqsxrm(MsgExit)')
   ```

   Jeśli określiłeś nazwę ścieżki, podaj pełną nazwę. Jeśli nazwa ścieżki zostanie pominięta, zakłada się, że program znajduje się w ścieżce określonej w pliku `qm.ini` (lub w katalogu WebSphere MQ dla systemu Windows-ścieżka określona w rejestrze).
5. Wyjście kanału odczytuje nagłówek komunikatu odwołania i znajduje plik, do którego odwołuje się ten nagłówek.
6. Wyjście kanału może następnie segmentować plik przed wysłaniem go w dół kanału wraz z nagłówkiem. W produkcie WebSphere MQ for AIX, WebSphere MQ for HP-UXi WebSphere MQ

for Solaris zmień właściciela grupy docelowej na 'mqm', aby program obsługi wyjścia komunikatów mógł utworzyć plik w tym katalogu. Zmień również uprawnienia katalogu docelowego, aby zezwolić członkom grupy mqm na zapisywanie w tym katalogu. Dane pliku nie są zapisywane w kolejkach produktu WebSphere MQ .

7. Gdy ostatni segment zbioru jest przetwarzany przez wyjście komunikatu odbierającego, komunikat odniesienia jest umieszczany w kolejce docelowej określonej przez amqsprm. Jeśli ta kolejka jest wyzwalana (oznacza to, że definicja określa atrybuty kolejki *Trigger*, *InitQ*i *Process*), wyzwalany jest program określony przez parametr PROC kolejki docelowej. Program, który ma zostać wyzwolony, musi być zdefiniowany w polu *ApplId* atrybutu *Process* .
8. Gdy komunikat referencyjny osiągnie kolejkę docelową (DQ), raport COA jest przesyłany z powrotem do aplikacji umieszczanie (amqsprm).
9. Przykładowy komunikat Get Reference Message, amqsgrm, pobiera komunikaty z kolejki określonej w komunikacie wyzwalacza wejściowego i sprawdza, czy plik istnieje.

Projekt przykładowego komunikatu umieszczonego w odwołaniu (amqsprma.c, AMQSPRM4)

W tym temacie przedstawiono szczegółowy opis przykładowego komunikatu umieszczonego w odwołaniu.

W tym przykładzie tworzony jest komunikat referencyjny, który odwołuje się do pliku i umieszcza go w określonej kolejce:

1. Przykład łączy się z lokalnym menedżerem kolejek za pomocą MQCONN.
2. Następnie zostanie otwarta (MQOPEN) kolejka modelowa, która jest używana do odbierania komunikatów raportu.
3. W tym przykładzie tworzony jest komunikat referencyjny zawierający wartości wymagane do przeniesienia pliku, na przykład nazwy plików źródłowych i docelowych oraz typ obiektu. Przykład: próbka dostarczana razem z produktem WebSphere MQ buduje komunikat referencyjny, aby wysłać plik `d:\x\file.in` z `QMGR1` do `QMGR2` i ponownie utworzyć plik jako `d:\y\file.out` , używając następujących parametrów:

   ```
   amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
   ```

   Gdzie `QR` jest definicją kolejki zdalnej, która odwołuje się do kolejki docelowej w systemie `QMGR2`.

   Uwaga: W przypadku platform UNIX and Linux należy użyć dwóch ukośników odwrotnych (\\) zamiast jednego, aby oznaczyć docelowy katalog plików. Oznacza to, że komenda **amqsprm** wygląda następująco:

   ```
   amqsprm -q QR -m QMGR1 -i /x/file.in -o d:\\y\\file.out -t FLATFILE
   ```
4. Komunikat referencyjny jest umieszczany (bez żadnych danych) do kolejki określonej przez parametr /q. Jeśli jest to kolejka zdalna, komunikat jest umieszczany w odpowiedniej kolejce transmisji.
5. Próbka czeka, przez czas określony w parametrze wagowym (domyślnie 15 sekund), dla raportów COA, które wraz z raportami wyjątków są wysyłane z powrotem do kolejki dynamicznej utworzonej w lokalnym menedżerze kolejek (QMGR1).

Projekt przykładu wyjścia komunikatu odniesienia (amqsxrma.c, AMQSXRM4)

W tym przykładzie rozpoznawane są komunikaty odniesienia z typem obiektu zgodnym z typem obiektu w polu danych użytkownika wyjścia komunikatu definicji kanału.

W przypadku tych komunikatów wykonywane są następujące działania:

- W kanale nadawczym lub kanale serwera określona długość danych jest kopiowana z określonego przesunięcia określonego pliku do obszaru pozostałego w buforze agenta po komunikacie odniesienia. Jeśli koniec pliku nie zostanie osiągnięty, po zaktualizowaniu pola *DataLogicalOffset* zostanie ponownie umieszczony komunikat odniesienia w kolejce transmisji.

- W kanale requestera lub odbiornika, jeśli pole *DataLogicalOffset* ma wartość zero, a podany plik nie istnieje, zostanie utworzony. Dane następujące po komunikacie referencyjnym są dodawane na końcu określonego pliku. Jeśli komunikat odniesienia nie jest ostatnim komunikatem dla określonego pliku, zostanie on usunięty. W przeciwnym razie zostanie on zwrócony do wyjścia kanału bez dołączonych danych, które mają zostać umieszczone w kolejce docelowej.

W przypadku kanałów nadawcy i serwera, jeśli pole *DataLogicalLength* w wejściowym komunikacie odniesienia jest zerowe, pozostała część pliku, od *DataLogicalOffset* do końca pliku, ma być wysłana wzdłuż kanału. Jeśli wartość nie jest równa zero, wysyłana jest tylko określona długość.

Jeśli wystąpi błąd (na przykład, jeśli próbka nie może otworzyć pliku), MQCXP.Parametr *ExitResponse* jest ustawiony na wartość MQXCC_SUPPRESS_FUNCTION w taki sposób, że przetwarzany komunikat jest umieszczany w kolejce niedostarczonych komunikatów, a nie w kolejce docelowej. W MQCXP zwracany jest kod sprzężenia zwrotnego.*Feedback* i zwróć do aplikacji, która wstawiła komunikat w polu *Feedback* deskryptora komunikatu komunikatu raportu. Wynika to z faktu, że aplikacja żądała raportów o wyjątkach, ustawiając parametr MQRO_EXCEPTION w polu *Report* deskryptora MQMD.

Jeśli kodowanie lub *CodedCharacterSetId* (CCSID) komunikatu odwołania różni się od wartości w menedżerze kolejek, komunikat odniesienia jest przekształcany w kodowanie lokalne i identyfikator CCSID. W naszym przykładzie, amqsprm, formatem obiektu jest MQFMT_STRING, więc amqsxrm przekształca dane obiektu na lokalny identyfikator CCSID w odbierającym końcu, zanim dane zostaną zapisane do pliku.

Nie należy określać formatu przesyłanego pliku jako MQFMT_STRING, jeśli plik zawiera znaki wielobajtowe (na przykład DBCS lub Unicode). Wynika to z faktu, że znak wielobajtowy może zostać podzielony, gdy plik jest podzielony na segmenty podczas wysyłania. Aby przesłać i przekształcić taki plik, należy określić format jako inny niż MQFMT_STRING w taki sposób, aby program obsługi wyjścia komunikatu odwołania nie konwertuje go i konwertuje plik na odbiorczy koniec po zakończeniu przesyłania.

Kompilowanie przykładu Wyjście z komunikatu odniesienia

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

Aby skompilować amqsxrma, należy użyć następujących komend:

W systemie AIX

```
xlc_r -q64 -e MsgExit -bE:amqsxrm.exp -bM:SRE -o amqsxrm_64_r
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -lmqm_r amqsqrma.c
```

W systemie HP-UX

```
$ c89 +DD64 +z -c -D_HPUX_SOURCE -o amqsxrma.o amqsqrma.c -IMQ_INSTALLATION_PATH/inc
$ ld -b amqsxrma.o -o /var/mqm/exits64/amqsxrma -LMQ_INSTALLATION_PATH/lib64
-L/usr/lib/pa20_64 -lmqm_r -lpthread
```

wł.Linux

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsxrma amqsqrma.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm_r
```

W systemie Solaris

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/amqsxrma amqsqrma.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm

 -lsocket
-lnsl -ldl
```

W systemie Windows

Produkt WebSphere MQ udostępnia teraz bibliotekę mqm z pakietami klientów, a także pakiety serwera, dlatego w poniższym przykładzie używany jest produkt `mqm.lib` zamiast `mqmvx.lib`:

```
cl amqsqrma.c /link /out:amqsxrm.dll /dll mqm.lib mqm.lib /def:amqsxrm.def
```

Ogólne informacje na temat pisania i kompilowania wyjść kanału można znaleźć w sekcji “Pisanie programów obsługi wyjścia kanału” na stronie 409

Projekt przykładowego komunikatu Get Reference Message (amqsgrma.c, AMQSGRM4)

W tym temacie opisano sposób projektowania przykładowego komunikatu Get Reference Message (Pobierz komunikat odniesienia).

Logika programu jest następująca:

1. Próbka jest wyzwalana i wyodrębnia nazwy kolejek i menedżerów kolejek z wejściowego komunikatu wyzwalacza.
2. Następnie łączy się on z określonym menedżerem kolejek za pomocą MQCONN i otwiera określoną kolejkę za pomocą komendy MQOPEN.
3. Przykładowe problemy MQGET z odstępem czasu oczekiwania na 15 sekund w pętli, aby pobrać komunikaty z kolejki.
4. Jeśli komunikat jest komunikatem odniesienia, próbka sprawdza istnienie pliku, który został przesłany.
5. Następnie zamyka kolejkę i rozłącza się z menedżerem kolejek.

Przykładowe programy żądania

Przykładowe programy Request demonstrują przetwarzanie klient/serwer. Przykłady są to klienty, które umieszczają komunikaty żądań w docelowej kolejce serwera, która jest przetwarzana przez program serwera. Oczekują na to, że program serwera umieje umieścić komunikat odpowiedzi w kolejce odpowiedzi.

Przykłady żądań umieszczają serię komunikatów żądań w docelowej kolejce serwera przy użyciu wywołania MQPUT. Komunikaty te określają kolejkę lokalną SYSTEM.SAMPLE.REPLY jako kolejkę odpowiedzi, która może być kolejką lokalną lub zdalną. Programy oczekują na komunikaty odpowiedzi, a następnie wyświetlają je. Odpowiedzi są wysyłane tylko wtedy, gdy docelowa kolejka serwera jest przetwarzana przez aplikację serwera lub jeśli aplikacja jest wyzwalana w tym celu (programy przykładowe Inkwire, Set i Echo zostały zaprojektowane do wyzwolenia). Próbka C czeka 1 minuta (próbka języka COBOL czeka 5 minut), pierwsza odpowiedź na przyjazd (w celu umożliwienia wyzwolenia aplikacji serwera) oraz 15 sekund na kolejne odpowiedzi, ale obie próbki mogą zakończyć się bez uzyskania odpowiedzi. Nazwy programów przykładowych żądań można znaleźć w sekcji “Funkcje demonstrowe w przykładowych programach” na stronie 100 .

Uruchamianie przykładowych programów żądania

Uruchamianie przykładów amqsreq0.c, amqsreq i amqsreqc

W wersji C programu przyjmuje się trzy parametry:

1. Nazwa docelowej kolejki serwera (konieczne)
2. Nazwa menedżera kolejek (opcjonalnie)
3. Kolejka odpowiedzi (opcjonalnie)

Na przykład wprowadź jedną z następujących opcji:

- `amqsreq myqueue qmanagername replyqueue`
- `amqsreqc myqueue qmanagername`
- `amq0req0 myqueue`

gdzie myqueue to nazwa docelowej kolejki serwera, qmanagername to nazwa menedżera kolejek, który jest właścicielem myqueue, a replyqueue to nazwa kolejki odpowiedzi.

Jeśli nazwa menedżera kolejek zostanie pominięta, to przyjmuje się, że domyślny menedżer kolejek jest właścicielem kolejki. Jeśli nazwa kolejki odpowiedzi zostanie pominięta, zostanie podana domyślna kolejka odpowiedzi.

Uruchamianie przykładu amq0req0.cbl

Wersja COBOL nie ma żadnych parametrów. Jest on połączony z domyślnym menedżerem kolejek i po uruchomieniu zostanie wyświetlony monit:

```
Please enter the name of the target server queue
```

Program pobiera dane wejściowe z komendy StdIn i dodaje każdy wiersz do docelowej kolejki serwera, przyjmując każdy wiersz tekstu jako treść komunikatu żądania. Program kończy się, gdy odczytywa się wiersz o wartości NULL.

Uruchamianie przykładu AMQSREQ4

Program C tworzy komunikaty, przyjmując dane ze stdin (klawiatura) z pustym czasem kończącego się wejścia. Program przyjmuje maksymalnie trzy parametry: nazwę kolejki docelowej (wymagane), nazwę menedżera kolejek (opcjonalnie) oraz nazwę kolejki odpowiedzi (opcjonalnie). Jeśli nie zostanie podana nazwa menedżera kolejek, zostanie użyty domyślny menedżer kolejek. Jeśli nie podano żadnej kolejki odpowiedzi, należy użyć komendy SYSTEM.SAMPLE.REPLY jest używana.

Poniżej przedstawiono przykład wywołania przykładowego programu C, określając kolejkę odpowiedzi, ale zezwalając na domyślne ustawienie menedżera kolejek:

```
CALL PGM(QMQM/AMQSREQ4) PARM('SYSTEM.SAMPLE.LOCAL' '' 'SYSTEM.SAMPLE.REPLY')
```

Uwaga: Należy pamiętać, że w nazwach kolejek rozróżniana jest wielkość liter. Wszystkie kolejki utworzone przez przykładowy program do tworzenia pliku AMQSAMP4 mają nazwy utworzone w postaci wielkich liter.

Uruchamianie przykładu AMQ0REQ4

Program w języku COBOL tworzy komunikaty, akceptując dane z klawiatury. Aby uruchomić program, należy wywołać program i określić nazwę kolejki docelowej jako parametr. Program akceptuje wprowadzanie danych z klawiatury do buforu i tworzy komunikat żądania dla każdej linii tekstu. Program zostanie zatrzymany po wprowadzeniu pustego wiersza na klawiaturze.

Uruchamianie przykładu żądania przy użyciu wyzwalania

Jeśli próbka jest używana z wyzwalaniem i jednym z przykładowych programów Inquire, Set lub Echo, wiersz danych wejściowych musi być nazwą kolejki, do której ma być dostęp wyzwalany program.

Systemy UNIX, Linux i Windows

Aby uruchomić przykłady za pomocą wyzwalania:

1. Uruchom program monitor wyzwalacza RUNMQTRM w jednej sesji (kolejka inicjowania SYSTEM.SAMPLE.TRIGGER jest dostępna do użycia).
2. Uruchom program amqsreq w innej sesji.
3. Upewnij się, że zdefiniowano docelową kolejkę serwera.

   Przykładowe kolejki, których można użyć jako docelowej kolejki serwera dla próbki żądania w celu umieszczenia komunikatów to:

   - SYSTEM.SAMPLE.INQ -dla przykładowego programu Inquire

- SYSTEM.SAMPLE.SET -dla przykładowego programu Set
- SYSTEM.SAMPLE.ECHO -dla przykładowego programu Echo

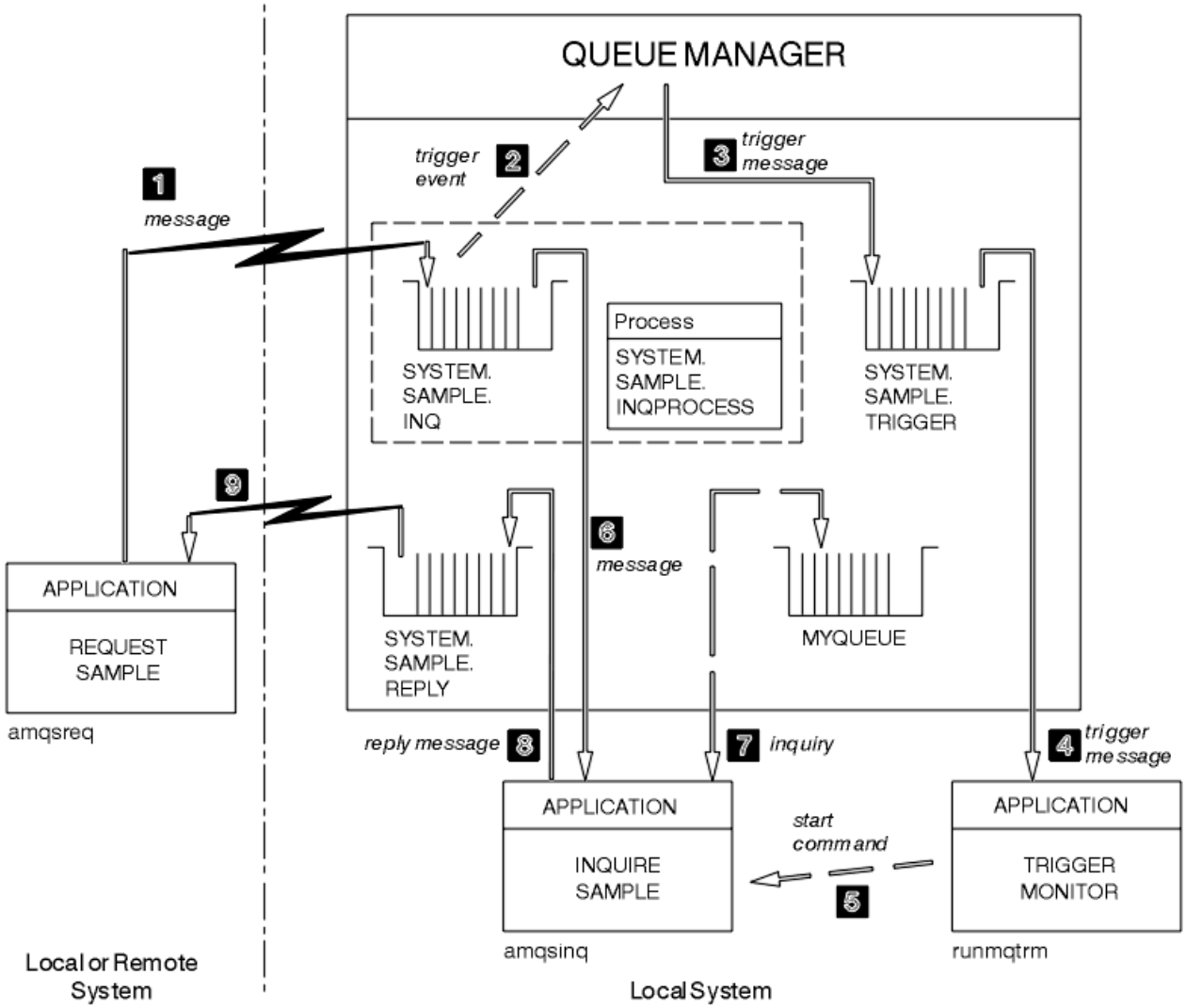
  Kolejki te mają typ wyzwalacza FIRST, więc jeśli przed uruchomieniem przykładu żądania są już komunikaty w kolejkach, aplikacje serwera nie są wyzwalane przez wysyłane komunikaty.

4. Upewnij się, że zdefiniowano kolejkę dla przykładowego programu Inquire, Set lub Echo do użycia.

Oznacza to, że monitor wyzwalacza jest gotowy, gdy próbka żądania wysyła komunikat.

Uwaga: Przykładowe definicje procesów utworzone za pomocą komend RUNMQSC i amqscos0.tst powodują wyzwolenie próbek C. Zmień definicje procesów w pliku amqscos0.tst i użyj komendy RUNMQSC z tym zaktualizowanym plikiem w celu użycia wersji COBOL.

Rysunek 22 na stronie 153 demonstruje, jak używać razem przykładów Request i Inquire.



Rysunek 22. Przykłady żądań i uzyskiwania informacji przy użyciu wyzwalania

W Rysunek 22 na stronie 153 próbka żądania umieszcza komunikaty w docelowej kolejce serwera SYSTEM.SAMPLE.INQ, a próbka zapytania wysyła zapytanie do kolejki, MYQUEUE. Alternatywnie można użyć jednej z kolejek przykładowych zdefiniowanych po uruchomieniu komendy amqscos0.tstlub dowolnej innej zdefiniowanej kolejki, dla przykładu Inquire.

Uwaga: Numery w programie Rysunek 22 na stronie 153 przedstawiają sekwencję zdarzeń.

Aby uruchomić przykłady żądań i uzyskiwania informacji przy użyciu wyzwalania:

1. Sprawdź, czy zdefiniowane są kolejki, które mają być używane. Uruchom komendę amqscos0.tst, aby zdefiniować przykładowe kolejki, a następnie zdefiniuj kolejkę MYQUEUE.
2. Uruchom komendę monitora wyzwalacza RUNMQTRM:

   ```
   RUNMQTRM -m qmanagername -q SYSTEM.SAMPLE.TRIGGER
   ```

3. Uruchom przykład żądania

   ```
   amqsreq SYSTEM.SAMPLE.INQ
   ```

   Uwaga: Obiekt procesu definiuje, co ma zostać wyzwolone. Jeśli klient i serwer nie są uruchomione na tej samej platformie, wszystkie procesy uruchomione przez monitor wyzwalacza muszą definiować *ApplType*, w przeciwnym razie serwer przyjmuje jego definicje domyślne (czyli typ aplikacji, która jest zwykle powiązana z maszyną serwera) i powoduje niepowodzenie.

   Listę typów aplikacji można znaleźć w sekcji ApplType.
4. Wprowadź nazwę kolejki, która ma być używana przez próbkę Inquiry:

   ```
   MYQUEUE
   ```

5. Wprowadź pusty wiersz (aby zakończyć działanie programu Request).
6. Następnie w przykładzie żądania zostanie wyświetlony komunikat zawierający dane programu Inquire uzyskanego z programu MYQUEUE.

W tym przypadku można użyć więcej niż jednej kolejki; w tym przypadku należy wprowadzić nazwy pozostałych kolejek w kroku “4” na stronie 154.

Więcej informacji na temat wyzwalania zawiera sekcja “Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy” na stronie 338.

Projekt przykładowego programu żądania

Program otwiera docelową kolejkę serwera, dzięki czemu może on umieszczać komunikaty. Korzysta on z wywołania MQOPEN z opcją MQOO_OUTPUT. Jeśli nie można otworzyć kolejki, w programie wyświetlany jest komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN.

Następnie program otwiera kolejkę zwrotną o nazwie SYSTEM.SAMPLE.REPLY , aby można było uzyskać komunikaty odpowiedzi. W tym celu program korzysta z wywołania MQOPEN z opcją MQOO_INPUT_EXCLUSIVE. Jeśli nie można otworzyć kolejki, w programie wyświetlany jest komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN.

Dla każdego wiersza danych wejściowych program odczytuje tekst do buforu i korzysta z wywołania MQPUT w celu utworzenia komunikatu żądania zawierającego tekst tego wiersza. W tym wywołaniu program korzysta z opcji raportu MQRO_EXCEPTION_WITH_DATA, aby zażądać, aby wszystkie komunikaty raportu wysłane na temat komunikatu żądania zawierały pierwsze 100 bajtów danych komunikatu. Program będzie kontynuowany aż do momentu, gdy dojdzie do końca danych wejściowych lub wywołanie MQPUT nie powiedzie się.

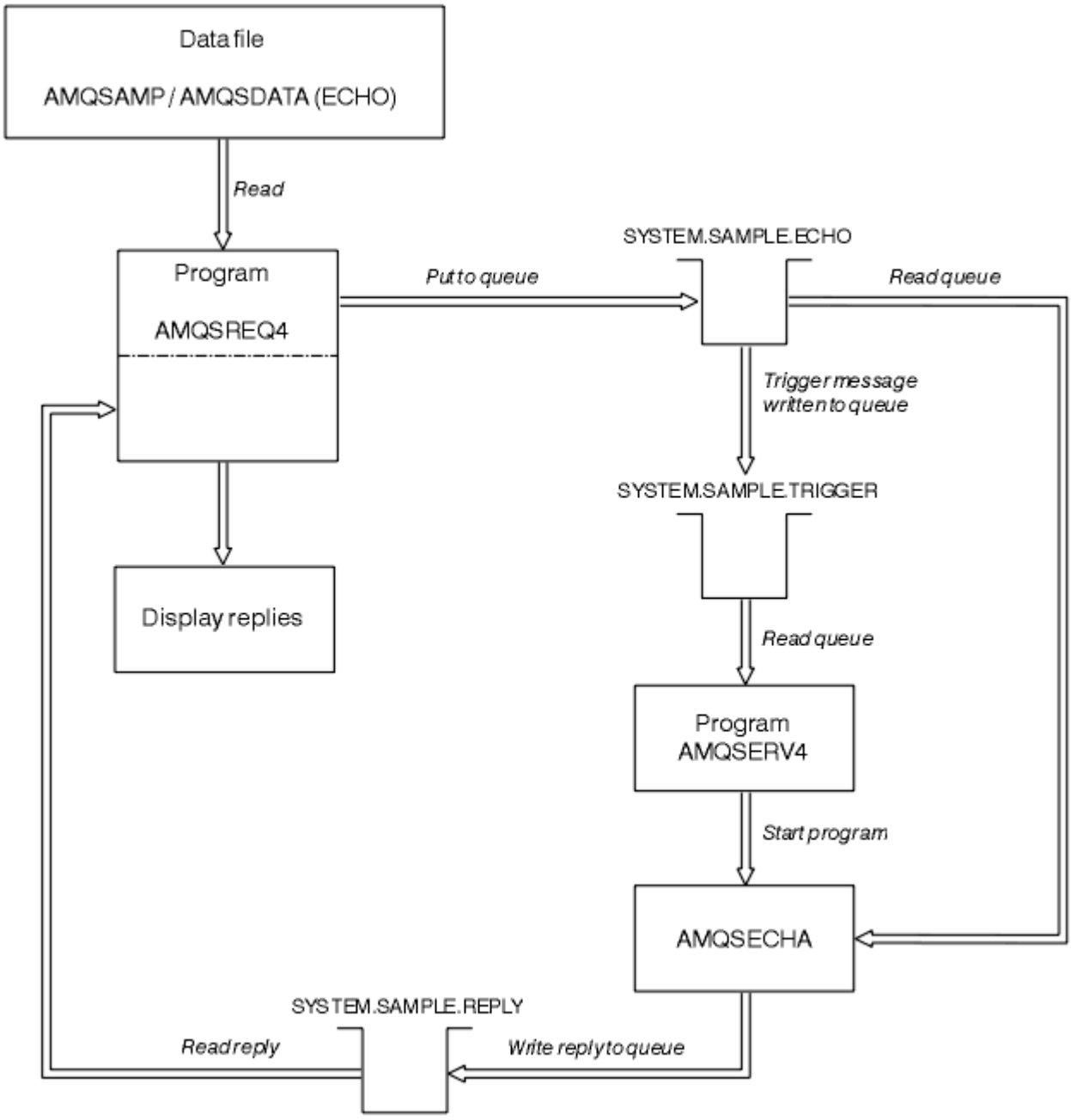
Następnie program korzysta z wywołania MQGET w celu usunięcia komunikatów odpowiedzi z kolejki i wyświetla dane zawarte w odpowiedziach. Wywołanie MQGET korzysta z opcji MQGMO_WAIT, MQGMO_CONVERT i MQGMO_ACCEPT_OBCIĘTE. *WaitInterval* jest 5 minut w wersji COBOL i 1 minuta w wersji C, dla pierwszej odpowiedzi (w celu umożliwienia wyzwolenia aplikacji serwera) i 15 sekund na kolejne odpowiedzi. Program czeka na te okresy, jeśli w kolejce nie ma żadnego komunikatu. Jeśli przed upływem tego okresu nie zostanie wyświetlony żaden komunikat, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_NO_MSG_AVAILABLE. Wywołanie korzysta również z opcji MQGMO_ACCEPT_TRUNCATED_MSG, dzięki czemu komunikaty dłuższe niż zadeklarowana wielkość buforu zostaną obcięte.

Program demonstruje, jak wyczyścić pola *MsgId* i *CorrelId* struktury MQMD po każdym wywołaniu MQGET, ponieważ wywołanie ustawia te pola na wartości zawarte w komunikacie, który pobiera.

Usunięcie zaznaczenia tych pól oznacza, że kolejne wywołania MQGET pobierają komunikaty w kolejności, w jakiej komunikaty są przechowywane w kolejce.

Program będzie kontynuowany do czasu, aż wywołanie MQGET zwróci kod przyczyny MQRC_NO_MSG_AVAILABLE lub wywołanie MQGET nie powiedzie się. Jeśli wywołanie nie powiedzie się, w programie zostanie wyświetlony komunikat o błędzie zawierający kod przyczyny.

Następnie program zamyka docelową kolejkę serwera oraz kolejkę odpowiedzi przy użyciu wywołania MQCLOSE.



Rysunek 23. Przykładowy schemat blokowy programu IBM i Client/Server (Echo)

Przykładowe programy

W przykładowych programach ustaw operacje umieszczania w kolejce są wykonywane za pomocą wywołania MQSET w celu zmiany atrybutu *InhibitPut* kolejki. Ponadto należy zapoznać się z projektowaniem przykładowych programów Set.

Nazwy tych programów można znaleźć w sekcji “Funkcje demonstrowe w przykładowych programach” na stronie 100 .

Programy są przeznaczone do uruchamiania jako programy wyzwalane, więc ich jedynym wejściem jest struktura MQTMC2 (komunikat wyzwalany), która zawiera nazwę kolejki docelowej z atrybutami, które mają zostać zapytane. W wersji C używana jest również nazwa menedżera kolejek. W wersji COBOL używany jest domyślny menedżer kolejek.

Aby proces wyzwalania działał poprawnie, należy upewnić się, że program przykładowy Ustaw, który ma być używany, jest wyzwalany przez komunikaty przychodzące do kolejki SYSTEM.SAMPLE.SET. W tym celu należy podać nazwę przykładowego programu Set, który ma być używany w polu *ApplicId* definicji procesu SYSTEM.SAMPLE.SETPROCESS. Kolejka przykładowa ma typ wyzwalacza FIRST; jeśli przed uruchomieniem przykładu żądania w kolejce istnieją już komunikaty, to próba ta nie jest wyzwalana przez wysyłane komunikaty.

Jeśli definicja została ustawiona poprawnie:

- W przypadku systemów UNIX, Linux i Windows uruchom program **runmqtrm** w jednej sesji, a następnie uruchom program amqsreq w innym.
- W przypadku produktu IBM iuruchom program AMQSERV4 w jednej sesji, a następnie uruchom program AMQSREQ4 w innej sesji. Można użyć komendy AMQSTRG4 zamiast AMQSERV4, ale ewentualne opóźnienia w składaniu zadań mogą spowodować, że śledzenie zdarzeń będzie mniej proste.

Za pomocą przykładowych programów żądania można wysyłać komunikaty żądań, z których każdy zawiera tylko nazwę kolejki, do kolejki SYSTEM.SAMPLE.SET. Dla każdego komunikatu żądania zestaw przykładowych programów wysyła komunikat odpowiedzi zawierający potwierdzenie, że operacje put zostały zahamowane w określonej kolejce. Odpowiedzi są wysyłane do kolejki odpowiedzi określonej w komunikacie żądania.

Projekt przykładowego programu Set

Program otwiera kolejkę nazwaną w strukturze komunikatu wyzwalacza, która została przekazana podczas jej uruchamiania. (Dla jasności zadzwonimy do tej *kolejki żądań*.) Program korzysta z wywołania MQOPEN, aby otworzyć tę kolejkę dla współużytkowanych danych wejściowych.

Program korzysta z wywołania MQGET w celu usunięcia komunikatów z tej kolejki. To wywołanie używa opcji MQGMO_ACCEPT_TRUNCATED_MSG i MQGMO_WAIT, z interwałem oczekiwania 5 sekund. Program testuje deskryptor każdego komunikatu, aby sprawdzić, czy jest to komunikat żądania. Jeśli tak nie jest, program usuwa komunikat i wyświetla komunikat ostrzegawczy.

Dla każdego komunikatu żądania usuniętego z kolejki żądań program odczytuje nazwę kolejki (którą nazywamy *kolejką docelową*) zawartą w danych i otwiera tę kolejkę przy użyciu wywołania MQOPEN z opcją MQOO_SET. Następnie program korzysta z wywołania MQSET w celu ustawienia wartości atrybutu *InhibitPut* kolejki docelowej na wartość MQQA_PUT_INHIBITED.

Jeśli wywołanie MQSET zakończy się pomyślnie, program korzysta z wywołania MQPUT1 w celu umieszczenia komunikatu odpowiedzi w kolejce odpowiedzi. Ten komunikat zawiera łańcuch PUT `inhibited`.

Jeśli wywołanie MQOPEN lub MQSET nie powiodło się, program korzysta z wywołania MQPUT1 w celu umieszczenia komunikatu `report` w kolejce odpowiedzi. W polu *Feedback* deskryptora komunikatu tego komunikatu raportu jest to kod przyczyny zwracany przez wywołanie MQOPEN lub MQSET, w zależności od tego, który błąd nie powiódł się.

Po wywołaniu komendy MQSET program zamknie kolejkę docelową przy użyciu wywołania MQCLOSE.

Jeśli w kolejce żądań nie pozostały żadne komunikaty, program zamknie tę kolejkę i rozłącza się z menedżerem kolejek.

Przykładowy program SSL/TLS

AMQSSLC to przykładowy program w języku C, który demonstruje sposób użycia struktur MQCNO i MQSCO w celu dostarczenia informacji o połączeniu klienta SSL/TLS w wywołaniu MQCONNX. Dzięki

temu aplikacja MQI klienta udostępnia definicję kanału połączenia klienckiego i ustawienia SSL/TLS w czasie wykonywania bez tabeli definicji kanału klienta (CCDT).

Jeśli zostanie podana nazwa połączenia, program konstruuje definicję kanału połączenia klienta w strukturze MQCD.

Jeśli zostanie podana nazwa macierzysta pliku repozytorium kluczy, program konstruuje strukturę MQSCO. Jeśli adres URL programu odpowiadającego OCSP jest również podany, program konstruuje strukturę MQAIR rekordu informacji uwierzytelniającej.

Następnie program łączy się z menedżerem kolejek za pomocą komendy MQCONNX. Określa on i drukuje nazwę menedżera kolejek, z którym jest połączony.

Ten program jest przeznaczony do połączenia jako aplikacja kliencka MQI. Może być jednak dowiązany jako zwykła aplikacja MQI. Następnie, po prostu łączy się z lokalnym menedżerem kolejek i ignoruje informacje o połączeniu klienta.

Program AMQSSLC akceptuje następujące parametry, z których wszystkie są opcjonalne:

-m QmgrName
Nazwa menedżera kolejek, z którym ma zostać nawiązane połączenie

-c ChannelName
Nazwa kanału, który ma być używany

-x ConnName
Nazwa połączenia z serwerem

Parametry SSL/TLS:

-k KeyReposStem
Nazwa macierzysta pliku repozytorium kluczy. Jest to pełna ścieżka do pliku bez przyrostka .kdb. Na przykład:

```
/home/user/client
C:\User\client
```

-s CipherSpec
Łańcuch SSL/TLS CipherSpec odpowiadający wartości SSLCIPH w definicji kanału SVRCONN w menedżerze kolejek.

-f
Określa, że tylko algorytmy certyfikowane FIPS 140-2 muszą być używane.

-b VALUE1[,VALUE2...]
Określa, że tylko algorytmy zgodne z pakietem B muszą być używane. Ten parametr jest listą rozdzielaną przecinkami jednej lub kilku z następujących wartości: NONE,128_BIT,192_BIT. Wartości te mają takie samo znaczenie jak w przypadku zmiennej środowiskowej MQSUITEB, a równoważne ustawienie EncryptionPolicySuiteB w sekcji SSL pliku konfiguracyjnego klienta.

-p Strategia
Określa strategię sprawdzania poprawności certyfikatu, która ma być używana. Może to być jedna z następujących wartości:

ANY
Zastosuj każdą ze strategii sprawdzania poprawności certyfikatów obsługiwanych przez bibliotekę bezpiecznych gniazd i zaakceptuj łańcuch certyfikatów, jeśli dowolna z strategii uzna łańcuch certyfikatów za poprawny. To ustawienie może być używane w celu zapewnienia maksymalnej wstecznej zgodności ze starszymi certyfikatami cyfrowymi, które nie są zgodne z nowoczesnymi standardami certyfikatów.

RFC5280
Zastosuj tylko strategię sprawdzania poprawności certyfikatu zgodną ze standardem RFC 5280. To ustawienie zapewnia bardziej restrykcyjne sprawdzanie poprawności niż ustawienie ANY, ale odrzuca niektóre starsze certyfikaty cyfrowe.

Wartość domyślna to ANY.

Parametr unieważnienia certyfikatu OCSP:

-o URL
Adres URL respondenta OCSP

Uruchamianie przykładowego programu SSL/TLS

Aby uruchomić przykładowy program SSL/TLS, należy najpierw skonfigurować środowisko SSL lub TLS. Następnie można uruchomić przykład z poziomu wiersza komend, podając liczbę parametrów.

O tym zadaniu

Poniższe instrukcje uruchamiają program przykładowy przy użyciu certyfikatów osobistych. W zależności od komendy można na przykład użyć certyfikatów ośrodka CA i sprawdzić ich status za pomocą modułu odpowiadającego OCSP. Należy zapoznać się z instrukcjami w ramach przykładu.

Procedura

1. Utwórz menedżer kolejek o nazwie QM1. Więcej informacji na ten temat zawiera sekcja crtmqm.
2. Utwórz repozytorium kluczy dla menedżera kolejek. Więcej informacji na ten temat zawiera sekcja Konfigurowanie repozytorium kluczy w systemach UNIX, Linux, and Windows.
3. Utwórz repozytorium kluczy dla klienta. Wywołaj ją *clientkey.kdb*.
4. Utwórz certyfikat osobisty dla menedżera kolejek. Więcej informacji na ten temat zawiera sekcja Tworzenie samopodpisanego certyfikatu osobistego w systemach UNIX, Linux, and Windows.
5. Utwórz certyfikat osobisty dla klienta.
6. Wyodrębnij certyfikat osobisty z repozytorium kluczy serwera i dodaj go do repozytorium klienta. Więcej informacji na ten temat zawiera sekcja Wyodrębnianie części publicznej certyfikatu samopodpisanego z repozytorium kluczy w systemach UNIX, Linux i Windows oraz Dodawanie certyfikatu ośrodka CA (lub publicznej części certyfikatu samopodpisanego) do repozytorium kluczy w systemach UNIX, Linux lub Windows.
7. Wyodrębnij certyfikat osobisty z repozytorium kluczy klienta i dodaj go do repozytorium kluczy serwera.
8. Utwórz kanał połączenia z serwerem za pomocą komendy MQSC:

   ```
   DEFINE CHANNEL(QM1SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCIPH(NULL_SHA)
   ```

   Więcej informacji na ten temat zawiera sekcja Serwer-kanał połączenia .
9. Zdefiniuj i uruchom program nasłuchujący kanału w menedżerze kolejek. Więcej informacji na ten temat zawiera sekcja DEFINE LISTENER i START LISTENER.
10. Uruchom przykładowy program za pomocą następującej komendy:

    ```
    AMQSSSLC -m QM1 -c QM1SVRCONN -x localhost
    -k "c:\Program Files\IBM\WebSphere MQ\clientkey" -s NULL_SHA
    -o http://dummy.OCSP.responder
    ```

Wyniki

Przykładowy program wykonuje następujące działania:

1. Łączy się z dowolnym określonym menedżerem kolejek lub z domyślnym menedżerem kolejek przy użyciu określonych opcji.
2. Służy do otwierania menedżera kolejek i uzyskiwania informacji o jego nazwie.
3. Zamyka menedżer kolejek.
4. Rozłącza się z menedżerem kolejek.

Jeśli przykładowy program zostanie uruchomiony pomyślnie, zostaną wyświetlone dane wyjściowe zbliżone do następującego przykładu:

```
Sample AMQSSSLC start
Connecting to queue manager QM1
Using the server connection channel QM1SVRCONN
on connection name localhost.
Using SSL CipherSpec NULL_SHA
Using SSL key repository stem c:\Program Files\IBM\WebSphere MQ\clientkey
Using OCSP responder URL http://dummy.OCSP.responder
Connection established to queue manager QM1

Sample AMQSSSLC end
```

Jeśli przykładowy program napotka problem, zostanie wyświetlony odpowiedni komunikat o błędzie, na przykład w przypadku podania niepoprawnego adresu URL odpowiadającego OCSP, zostanie wyświetlony następujący komunikat:

```
MQCONNX ended with reason code 2553
```

Lista kodów przyczyn znajduje się w sekcji Kody przyczyny funkcji API.

Przykładowe programy wyzwalające

Funkcja podana w przykładzie wyzwalającym jest podzbiorem tego udostępnionego w monitorze wyzwalacza w programie **runmqtrm** .

Nazwy tych programów można znaleźć w sekcji “Funkcje demonstrowe w przykładowych programach” na stronie 100 .

Wzór próby wyzwalającej

Program przykładowy wyzwalający otwiera kolejkę inicjującyprzy użyciu wywołania MQOPEN z opcją MQOO_INPUT_AS_Q_DEF. Pobiera on komunikaty z kolejki inicjuj przy użyciu wywołania MQGET z opcjami MQGMO_ACCEPT_TRUNCATED_MSG i MQGMO_WAIT, określając nieograniczony odstęp czasu oczekiwania. Program kasuje pola *MsgId* i *CorrelId* przed każdym wywołaniem MQGET w celu pobrania komunikatów w sekwencji.

Po pobraniu komunikatu z kolejki inicjuj program testuje komunikat, sprawdzając wielkość komunikatu, aby upewnić się, że jest to ta sama wielkość co struktura MQTM. Jeśli test zakończy się niepowodzeniem, program wyświetli ostrzeżenie.

W przypadku poprawnych komunikatów wyzwalacza próba wyzwalająca kopiuje dane z następujących pól: *ApplicId*, *EnvrData*, *Version*i *ApplType*. Ostatnie dwa z tych pól są numeryczne, dlatego program tworzy odpowiedniki znaków w celu użycia w strukturze MQTMC2 dla systemów UNIX, Linuxi Windows .

Próba wyzwalająca wysyła komendę uruchomienia do aplikacji określonej w polu *ApplicId* komunikatu wyzwalacza, a następnie przekazuje strukturę MQTMC2 lub MQTMC (wersja znakowa komunikatu wyzwalacza). W systemach UNIX, Linux i Windows pole *EnvData* jest używane jako rozszerzenie do wywołującego łańcucha komendy.

Na koniec program zamyka kolejkę inicjujący.

Uruchamianie przykładowych programów Triggering

Ten temat zawiera informacje na temat uruchamiania przykładowych programów Triggering.

Uruchamianie przykładów amqstrg0.c, amqstrg i amqstrgc

Program przyjmuje 2 parametry:

1. Nazwa kolejki inicjuj (niezbędnej)
2. Nazwa menedżera kolejek (opcjonalnie)

Jeśli menedżer kolejek nie jest określony, łączy się z domyślnym menedżerem kolejek. Przykładowa kolejka inicjujący zostanie zdefiniowana podczas uruchamiania komendy amqscos0.tst; , która jest nazwą kolejki SYSTEM.SAMPLE.TRIGGERi można go używać podczas uruchamiania tego programu.

Uwaga: Funkcja w tym przykładzie jest podzbiorem pełnej funkcji wyzwalającej, która jest dostarczana w programie **runmqtrm** .

Projekt serwera wyzwalacza

Projekt serwera wyzwalającego jest podobny do projektu monitora wyzwalacza, z wyjątkiem tego, że serwer wyzwalacza:

- Umożliwia aplikacjom MQAT_CICS oraz MQAT_OS400
- W przypadku aplikacji CICS substytuuje *EnvData*, na przykład w celu określenia regionu CICS , z komunikatu wyzwalacza w komendzie STRCICSUSR.
- Otwiera kolejkę inicjujową dla współużytkowanych danych wejściowych, tak aby wiele serwerów wyzwalających było uruchamianych jednocześnie.

Uwaga: Programy uruchomione przez program AMQSERV4 nie mogą korzystać z wywołania MQDISC, ponieważ powoduje to zatrzymanie serwera wyzwalacza. Jeśli programy uruchomione przez program AMQSERV4 korzystają z wywołania MQCONN, uzyskają kod przyczyny MQRC_ALREADY_CONNECTED.

Próbki TUXEDO

Dowiedz się więcej o programach umieszczania i pobierania próbek dla TUXEDO, a także w budowaniu środowiska serwera w TUXEDO.

Przed uruchomieniem tych przykładów należy zbudować środowisko serwera.

Uwaga: W tym temacie znak ukośnika odwrotnego (\) jest używany do dzielenia długich komend na więcej niż jedną linię. Nie wprowadzaj tego znaku. Każdą komendę należy wprowadzić jako pojedynczą linię.

Budowanie środowiska serwera

Informacje na temat budowania środowiska serwera dla produktu WebSphere MQ dla różnych platform.

Zakłada się, że użytkownik ma działające środowisko TUXEDO.

Budowanie środowiska serwera dla produktu WebSphere MQ for AIX (32-bitowe)

1. Utwórz katalog (na przykład < APPDIR>), w którym budowane jest środowisko serwera, a następnie wykonaj wszystkie komendy w tym katalogu.
2. Wyeksportuj następujące zmienne środowiskowe, gdzie TUXDIR to katalog główny dla TUXEDO, a *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym jest zainstalowany produkt WebSphere MQ :

```
$ export CFLAGS="-I MQ_INSTALLATION_PATH□inc -I /<APPDIR> -L MQ_INSTALLATION_PATH□lib"
$ export LDOPTS="-lmqm"
$ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=/<APPDIR>/amqstxvx.V
$ export LIBPATH=$TUXDIR□lib:MQ_INSTALLATION_PATH□lib:□lib
```

3. Dodaj następujące informacje do pliku udataobj/RM pliku TUXEDO

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa -lmqm
```

4. Uruchom następujące komendy:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
```

```
              -s MGET1:MGET \
              -v -bshm
       $ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
              -f MQ_INSTALLATION_PATH/lib/libmqm.a \
              -r MQSeries_XA_RMI -s MPUT2:MPUT
              -s MGET2:MGET \
              -v -bshm
       $ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
              -f MQ_INSTALLATION_PATH/lib/libmqm.a
       $ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
              -f MQ_INSTALLATION_PATH/lib/libmqm.a
```

5. Edytuj plik ubbstxcx.cfg i dodaj w razie potrzeby szczegóły dotyczące nazwy komputera, katalogów roboczych i menedżera kolejek:

```
       $ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Utwórz TLOGDEVICE:

```
      $tmadmin -c
```

   Zostanie wyświetlona zachęta. W tym pytaniu wpisz:

```
      > crdl -z /<APPDIR>/TLOG1
```

7. Uruchom menedżer kolejek:

```
       $ strmqm
```

8. Uruchom Tuxedo:

```
       $ tmboot -y
```

Teraz można użyć programów do umieszczania komunikatów w kolejce i dogets w celu umieszczenia komunikatów w kolejce i pobrania ich z kolejki.

Budowanie środowiska serwera dla produktu WebSphere MQ for AIX (wersja 64-bitowa)

1. Utwórz katalog (na przykład < APPDIR>), w którym budowane jest środowisko serwera, a następnie wykonaj wszystkie komendy w tym katalogu.
2. Wyeksportuj następujące zmienne środowiskowe, gdzie TUXDIR reprezentuje katalog główny dla TUXEDO, a *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym produkt WebSphere MQ jest installed.:

```
       $ export CFLAGS="-I MQ_INSTALLATION_PATH□inc -I /<APPDIR> -L
MQ_INSTALLATION_PATH□lib64"
       $ export LDOPTS="-lmqm"
       $ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
       $ export VIEWFILES=/<APPDIR>/amqstxvx.V
       $ export LIBPATH=$TUXDIR□lib64:MQ_INSTALLATION_PATH□lib64:□lib64
```

3. Dodaj następujące informacje do pliku udataobj/RM pliku TUXEDO

```
       MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa64 -lmqm
```

4. Uruchom następujące komendy:

```
       $ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
       $ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
       $ buildtms    -o MQXA -r MQSeries_XA_RMI
       $ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
              -f MQ_INSTALLATION_PATH/lib64/libmqm.a \
              -r MQSeries_XA_RMI -s MPUT1:MPUT \
```

```
              -s MGET1:MGET \
              -v -bshm
       $ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
              -f MQ_INSTALLATION_PATH/lib64/libmqm.a \
              -r MQSeries_XA_RMI -s MPUT2:MPUT
              -s MGET2:MGET \
              -v -bshm
       $ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
              -f MQ_INSTALLATION_PATH/lib64/libmqm.a
       $ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
              -f MQ_INSTALLATION_PATH/lib64/libmqm.a
```

5. Edytuj plik ubbstxcx.cfg i dodaj w razie potrzeby szczegóły dotyczące nazwy komputera, katalogów roboczych i menedżera kolejek:

```
       $ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Utwórz TLOGDEVICE:

```
      $tmadmin -c
```

   Zostanie wyświetlona zachęta. W tym pytaniu wpisz:

```
      > crdl -z /<APPDIR>/TLOG1
```

7. Uruchom menedżer kolejek:

```
       $ strmqm
```

8. Uruchom Tuxedo:

```
       $ tmboot -y
```

Teraz można użyć programów do umieszczania komunikatów w kolejce i dogets w celu umieszczenia komunikatów w kolejce i pobrania ich z kolejki.

Budowanie środowiska serwera dla produktu WebSphere MQ for Solaris (wersja 32-bitowa)

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

1. Utwórz katalog (na przykład *APPDIR*), w którym środowisko serwera jest budowane i wykonuje wszystkie komendy znajdujące się w tym katalogu.
2. Wyeksportuj następujące zmienne środowiskowe, gdzie TUXDIR to katalog główny dla TUXEDO:

```
       $ export CFLAGS="-I /APPDIR"
       $ export FIELDTBLS=amqstxvx.flds
       $ export VIEWFILES=amqstxvx.V
       $ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
       $ export LD_LIBRARY_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
```

3. Dodaj następujące informacje do pliku udataobj/RM pliku TUXEDO.

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib/libmqmxa.a MQ_INSTALLATION_PATH/lib/libmqm.so \
/opt/tuxedo/lib/libtux.a
```

4. Uruchom następujące komendy:

```
       $ mkfldhdr    amqstxvx.flds
       $ viewc       amqstxvx.v
       $ buildtms    -o MQXA -r MQSeries_XA_RMI
       $ buildserver -o MQSERV1 -f amqstxsx.c \
```

```
              -f MQ_INSTALLATION_PATH/lib/libmqm.so \
              -r MQSeries_XA_RMI -s MPUT1:MPUT \
              -s MGET1:MGET \
              -v -bshm
              -l -ldl
       $ buildserver -o MQSERV2 -f amqstxsx.c \
              -f MQ_INSTALLATION_PATH/lib/libmqm.so \
              -r MQSeries_XA_RMI -s MPUT2:MPUT \
              -s MGET2:MGET \
              -v -bshm
              -l -ldl
       $ buildclient -o doputs  -f amqstxpx.c \
              -f MQ_INSTALLATION_PATH/lib/libmqm.so  \

       $ buildclient -o dogets  -f amqstxgx.c \
              -f MQ_INSTALLATION_PATH/lib/libmqm.so
```

5. Edytuj plik ubbstxcx.cfg i dodaj w razie potrzeby szczegóły dotyczące nazwy komputera, katalogów roboczych i menedżera kolejek:

```
       $ tmloadcf    -y ubbstxcx.cfg
```

6. Utwórz TLOGDEVICE:

```
      $tmadmin -c
```

   Zostanie wyświetlona zachęta. W tym pytaniu wpisz:

```
      > crdl -z /APPDIR/TLOG1
```

7. Uruchom menedżer kolejek:

```
       $ strmqm
```

8. Uruchom Tuxedo:

```
       $ tmboot -y
```

Teraz można użyć programów do umieszczania komunikatów w kolejce i dogets w celu umieszczenia komunikatów w kolejce i pobrania ich z kolejki.

Budowanie środowiska serwera dla produktu WebSphere MQ for Solaris (wersja 64-bitowa)

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

1. Utwórz katalog (na przykład < APPDIR>), w którym budowane jest środowisko serwera, a następnie wykonaj wszystkie komendy w tym katalogu.
2. Wyeksportuj następujące zmienne środowiskowe, gdzie TUXDIR to katalog główny dla TUXEDO:

```
       $ export CFLAGS="-I /<APPDIR>"
       $ export FIELDTBLS=amqstxvx.flds
       $ export VIEWFILES=amqstxvx.V
       $ export SHLIB_PATH=$TUXDIR□lib:MQ_INSTALLATION_PATH□lib:□lib64
       $ export LD_LIBRARY_PATH=$TUXDIR□lib64:MQ_INSTALLATION_PATH□lib64:□lib64
```

3. Dodaj następujące informacje do pliku udataobj/RM pliku TUXEDO.

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib64/libmqmxa64.a MQ_INSTALLATION_PATH/lib64/libmqm.so \
/opt/tuxedo/lib64/libtux.a
```

4. Uruchom następujące komendy:

```
$ mkfldhdr    amqstxvx.flds
$ viewc       amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
       -l -ldl
$ buildserver -o MQSERV2 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
       -l -ldl
$ buildclient -o doputs  -f amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so  \

$ buildclient -o dogets  -f amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
```

5. Edytuj plik ubbstxcx.cfg i dodaj w razie potrzeby szczegóły dotyczące nazwy komputera, katalogów roboczych i menedżera kolejek:

```
$ tmloadcf    -y ubbstxcx.cfg
```

6. Utwórz TLOGDEVICE:

```
$tmadmin -c
```

   Zostanie wyświetlona zachęta. W tym pytaniu wpisz:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Uruchom menedżer kolejek:

```
$ strmqm
```

8. Uruchom Tuxedo:

```
$ tmboot -y
```

Teraz można użyć programów do umieszczania komunikatów w kolejce i dogets w celu umieszczenia komunikatów w kolejce i pobrania ich z kolejki.

Budowanie środowiska serwera dla produktu WebSphere MQ for HP-UX (wersja 32-bitowa)

Uwaga: 32-bitowe środowisko serwera TUXEDO może być zbudowane tylko na platformie Itanium.

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

1. Utwórz katalog (na przykład < APPDIR>), w którym budowane jest środowisko serwera, a następnie wykonaj wszystkie komendy w tym katalogu.
2. Wyeksportuj następujące zmienne środowiskowe, gdzie TUXDIR to katalog główny dla TUXEDO:

```
$ export CFLAGS="-Aa -D_HPUX_SOURCE"
$ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=$APPDIR/amqstxvx.V
$ export TUXCONFIG=$APPDIR/tuxconfig
$ export PATH=$TUXDIR/bin:/usr/bin:/sbin:MQ_INSTALLATION_PATH/bin:$PATH
$ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
$ export FLDTBLDIR=$APPDIR:$TUXDIR/udataobj
```

3. Dodaj następujące informacje do pliku udataobj/RM pliku TUXEDO

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib/libmqmxa.so MQ_INSTALLATION_PATH/lib/libmqm.so \
/opt/tuxedo/lib/libtux.sl
```

4. Uruchom następujące komendy:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
```

   Po uruchomieniu komend mkfldhdr i viewc, plik nagłówkowy amqstxvx.h jest tworzony w katalogu aplikacji TUXEDO. Skopiuj ten plik z katalogu aplikacji TUXEDO do katalogu włączanego TUXEDO, a następnie uruchom następujące komendy.

```
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
$ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
```

5. Edytuj plik ubbstxcx.cfg i dodaj w razie potrzeby szczegóły dotyczące nazwy komputera, katalogów roboczych i menedżera kolejek:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Utwórz TLOGDEVICE:

```
$tmadmin -c
```

   Zostanie wyświetlona zachęta. W tym pytaniu wpisz:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Uruchom menedżer kolejek:

```
$ strmqm
```

8. Uruchom TUXEDO:

```
$ tmboot -y
```

Teraz można użyć programów do umieszczania komunikatów w kolejce i dogets w celu umieszczenia komunikatów w kolejce i pobrania ich z kolejki.

Budowanie środowiska serwera dla produktu WebSphere MQ for HP-UX (wersja 64-bitowa)

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

1. Utwórz katalog (na przykład < APPDIR>), w którym budowane jest środowisko serwera, a następnie wykonaj wszystkie komendy w tym katalogu.

2. Wyeksportuj następujące zmienne środowiskowe, gdzie TUXDIR to katalog główny dla TUXEDO:

```
$ export CFLAGS="-Aa -D_HPUX_SOURCE"
$ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=$APPDIR/amqstxvx.V
$ export TUXCONFIG=$APPDIR/tuxconfig
$ export PATH=$TUXDIR/bin:/usr/bin:/sbin:MQ_INSTALLATION_PATH/bin:$PATH
$ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib64:/lib64
$ export FLDTBLDIR=$APPDIR:$TUXDIR/udataobj
```

3. Dodaj następujące informacje do pliku udataobj/RM pliku TUXEDO

   Na platformie HP-UX IA64 (IPF):

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib64/libmqmxa64.so MQ_INSTALLATION_PATH/lib64/libmqm.so \
/opt/tuxedo/lib/libtux.sl
```

   Uwaga: Biblioteki produktu WebSphere MQ dostarczone na platformie HP-UX IA64 (IPF) mają rozszerzenie nazwy pliku .so.

4. Uruchom następujące komendy:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
```

   Po uruchomieniu komend mkfldhdr i viewc, plik nagłówkowy amqstxvx.h jest tworzony w katalogu aplikacji TUXEDO. Skopiuj ten plik z katalogu aplikacji TUXEDO do katalogu włączanego TUXEDO, a następnie uruchom następujące komendy.

```
$ buildtms    -o MQXA -r MQSeries_XA_RMI
```

   Na platformie HP-UX IA64 (IPF):

```
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
$ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
```

5. Edytuj plik ubbstxcx.cfg i dodaj w razie potrzeby szczegóły dotyczące nazwy komputera, katalogów roboczych i menedżera kolejek:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Utwórz TLOGDEVICE:

```
$tmadmin -c
```

   Zostanie wyświetlona zachęta. W tym pytaniu wpisz:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Uruchom menedżer kolejek:

```
$ strmqm
```

8. Uruchom TUXEDO:

```
$ tmboot -y
```

Teraz można użyć programów do umieszczania komunikatów w kolejce i dogets w celu umieszczenia komunikatów w kolejce i pobrania ich z kolejki.

Budowanie środowiska serwera dla produktu WebSphere MQ for Windows (wersja 32-bitowa)

Uwaga: Zmień pola identyfikowane przez <> w następujący sposób, do ścieżek katalogów:

< MQMDIR>	Ścieżka katalogu określona podczas instalowania produktu WebSphere MQ , na przykład `g:\Program Files\IBM\WebSphere MQ` .
< KAT_TUX>	Ścieżka do katalogu określona podczas instalowania TUXEDO, na przykład `f:\tuxedo`
< KATALOG_PLIKU>	ścieżka do katalogu, która ma być używana dla przykładowej aplikacji, na przykład `f:\tuxedo\apps\mqapp`

Aby zbudować środowisko serwera i przykłady:

1. Utwórz katalog aplikacji, w którym ma zostać utworzona przykładowa aplikacja, na przykład:

```
f:\tuxedo\apps\mqapp
```

2. Skopiuj następujące przykładowe pliki z przykładowego katalogu produktu WebSphere MQ do katalogu aplikacji:

```
amqstxmn.mak
amqstxen.env
ubbstxcn.cfg
```

3. Zmodyfikuj każdy z tych plików, aby ustawić nazwy katalogów i ścieżki katalogów używane podczas instalacji.
4. Zmodyfikuj plik ubbstxcn.cfg (patrz Rysunek 24 na stronie 168), aby dodać szczegółowe informacje o nazwie komputera i menedżerze kolejek, z którym ma zostać nawiązane połączenie.
5. Dodaj następujący wiersz do pliku TUXEDO < TUXDIR > udataobj\rm

```
MQSeries_XA_RMI;MQRMIXASwitchDynamic;
   <MQMDIR>\tools\lib\mqmxa.lib <MQMDIR>\tools\lib\mqm.lib
```

gdzie < MQMDIR > jest zastępowany, tak jak przedstawiono to w poprzednim przykładzie. Pomimo tego, że w tym miejscu znajdują się dwie linie, nowa pozycja musi być jedną linią w pliku.

6. Ustaw następujące zmienne środowiskowe:

```
TUXDIR=<TUXDIR>
TUXCONFIG=<APPDIR>\tuxconfig
FIELDTBLS=<MQMDIR>\tools\c\samples\amqstxvx.fld
LANG=C
```

7. Utwórz urządzenie TLOG dla TUXEDO. Aby to zrobić, wywołaj komendę `tmadmin -c`i wprowadź komendę:

```
crdl -z <APPDIR>\TLOG
```

gdzie <*APPDIR*> jest zastępowany.

8. Ustaw bieżący katalog na < APPDIR>, a następnie wywołaj przykładowy plik makefile (amqstxmn.mak) jako zewnętrzny plik makefile projektu. Na przykład przy użyciu programu Microsoft Visual C++ wprowadź komendę:

   ```
   msvc amqstxmn.mak
   ```

   Wybierz opcję **build** (kompilacja), aby zbudować wszystkie przykładowe programy.

```
*RESOURCES
IPCKEY        99999
UID           0
GID           0
MAXACCESSERS  20
MAXSERVERS    20
MAXSERVICES   50
MASTER        SITE1
MODEL         SHM
LDBAL         N

*MACHINES
<MachineName> LMID=SITE1
              TUXDIR="f:\tuxedo"
              APPDIR="f:\tuxedo\apps\mqapp;g:\Program Files\IBM\WebSphere MQ\bin"
              ENVFILE="f:\tuxedo\apps\mqapp\amqstxen.env"
              TUXCONFIG="f:\tuxedo\apps\mqapp\tuxconfig"
              ULOGPFX="f:\tuxedo\apps\mqapp\ULOG"
              TLOGDEVICE="f:\tuxedo\apps\mqapp\TLOG"
              TLOGNAME=TLOG
              TYPE="i386NT"
              UID=0
              GID=0

*GROUPS
GROUP1
        LMID=SITE1  GRPNO=1
        TMSNAME=MQXA
        OPENINFO="MQSeries_XA_RMI:MYQUEUEMANAGER"

*SERVERS
DEFAULT: CLOPT="-A -- -m MYQUEUEMANAGER"

MQSERV1     SRVGRP=GROUP1 SRVID=1
MQSERV2     SRVGRP=GROUP1 SRVID=2

*SERVICES
MPUT1
MGET1
MPUT2
MGET2
```

Rysunek 24. Przykład pliku ubbstxcn.cfg dla produktu WebSphere MQ for Windows

Uwaga: Zmień nazwy katalogów i ścieżki katalogów tak, aby były zgodne z instalacją. Zmień także nazwę menedżera kolejek MYQUEUEMANAGER na nazwę menedżera kolejek, z którym ma zostać nawiązane połączenie. Inne informacje, które należy dodać, są identyfikowane przez znaki <> .

Przykładowy plik ubbconfig dla produktu WebSphere MQ for Windows jest wymieniony w sekcji Rysunek 24 na stronie 168. Jest on dostarczany w postaci pliku ubbstxcn.cfg w katalogu przykładów produktu WebSphere MQ .

Przykładowy plik makefile (patrz Rysunek 25 na stronie 169) dostarczany z produktem WebSphere MQ for Windows nosi nazwę ubbstxmn.maki znajduje się w katalogu przykładów produktu WebSphere MQ .

```
TUXDIR  = f:\tuxedo
MQMDIR  = g:\Program Files\IBM\WebSphere MQ
APPDIR  = f:\tuxedo\apps\mqapp
MQMLIB  = $(MQMDIR)\tools\lib
MQMINC  = $(MQMDIR)\tools\c\include
MQMSAMP = $(MQMDIR)\tools\c\samples
INC = -f "-I$(MQMINC) -I$(APPDIR)"
DBG = -f "/Zi"

amqstx.exe:
 $(TUXDIR)\bin\mkfldhdr    -d$(APPDIR) $(MQMSAMP)\amqstxvx.fld
 $(TUXDIR)\bin\viewc       -d$(APPDIR) $(MQMSAMP)\amqstxvx.v
 $(TUXDIR)\bin\buildtms    -o MQXA -r MQSeries_XA_RMI
 $(TUXDIR)\bin\buildserver -o MQSERV1 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT1:MPUT -s MGET1:MGET
 $(TUXDIR)\bin\buildserver -o MQSERV2 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT2:MPUT -s MGET2:MGET
 $(TUXDIR)\bin\buildclient -o doputs -f $(MQMSAMP)\amqstxpx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG)
 $(TUXDIR)\bin\buildclient -o dogets -f $(MQMSAMP)\amqstxgx.c \
                           -f $(MQMLIB)\mqm.lib $(INC) -v $(DBG)
 $(TUXDIR)\bin\tmloadcf    -y $(APPDIR)\ubbstxcn.cfg
```

Rysunek 25. Przykładowy plik makefile TUXEDO dla produktu WebSphere MQ for Windows

Budowanie środowiska serwera dla produktu WebSphere MQ for Windows (wersja 64-bitowa)

Uwaga: Zmień pola identyfikowane przez <> w następujący sposób, do ścieżek katalogów:

< MQMDIR>	Ścieżka katalogu określona podczas instalowania produktu WebSphere MQ , na przykład `g:\Program Files\IBM\WebSphere MQ` .
< KAT_TUX>	Ścieżka do katalogu określona podczas instalowania TUXEDO, na przykład `f:\tuxedo`
< KATALOG_PLIKU>	ścieżka do katalogu, która ma być używana dla przykładowej aplikacji, na przykład `f:\tuxedo\apps\mqapp`

Aby zbudować środowisko serwera i przykłady:

1. Utwórz katalog aplikacji, w którym ma zostać utworzona przykładowa aplikacja, na przykład:

   ```
   f:\tuxedo\apps\mqapp
   ```

2. Skopiuj następujące przykładowe pliki z przykładowego katalogu produktu WebSphere MQ do katalogu aplikacji:

   ```
   amqstxmn.mak
   amqstxen.env
   ubbstxcn.cfg
   ```

3. Zmodyfikuj każdy z tych plików, aby ustawić nazwy katalogów i ścieżki katalogów używane podczas instalacji.
4. Edytuj `ubbstxcn.cfg` (patrz sekcja Rysunek 26 na stronie 170), aby dodać szczegóły dotyczące nazwy komputera i menedżera kolejek, z którym ma zostać nawiązane połączenie.
5. Dodaj następujący wiersz do pliku TUXEDO <TUXDIR>`udataobj\rm`

   ```
   MQSeries_XA_RMI;MQRMIXASwitchDynamic;
      <MQMDIR>\tools\lib64\mqmxa64.lib <MQMDIR>\tools\lib64\mqm.lib
   ```

gdzie < MQMDIR > jest zastępowany. Pomimo tego, że w tym miejscu znajdują się dwie linie, nowa pozycja musi być jedną linią w pliku.

6. Ustaw następujące zmienne środowiskowe:

```
TUXDIR=<TUXDIR>
TUXCONFIG=<APPDIR>\tuxconfig
FIELDTBLS=<MQMDIR>\tools\c\samples\amqstxvx.fld
LANG=C
```

7. Utwórz urządzenie TLOG dla TUXEDO. Aby to zrobić, wywołaj komendę `tmadmin -c` i wprowadź komendę:

```
crdl -z <APPDIR>\TLOG
```

gdzie <*APPDIR*> jest zastępowany, tak jak pokazano to w poprzednim przykładzie.

8. Ustaw bieżący katalog na < APPDIR>, a następnie wywołaj przykładowy plik makefile (amqstxmn.mak) jako zewnętrzny plik makefile projektu. Na przykład przy użyciu programu Microsoft Visual C++ wprowadź komendę:

```
msvc amqstxmn.mak
```

Wybierz opcję **build** (kompilacja), aby zbudować wszystkie przykładowe programy.

```
*RESOURCES
IPCKEY        99999
UID           0
GID           0
MAXACCESSERS  20
MAXSERVERS    20
MAXSERVICES   50
MASTER        SITE1
MODEL         SHM
LDBAL         N

*MACHINES
<MachineName> LMID=SITE1
              TUXDIR="f:\tuxedo"
              APPDIR="f:\tuxedo\apps\mqapp;g:\Program Files\IBM\WebSphere MQ\bin"
              ENVFILE="f:\tuxedo\apps\mqapp\amqstxen.env"
              TUXCONFIG="f:\tuxedo\apps\mqapp\tuxconfig"
              ULOGPFX="f:\tuxedo\apps\mqapp\ULOG"
              TLOGDEVICE="f:\tuxedo\apps\mqapp\TLOG"
              TLOGNAME=TLOG
              TYPE="i386NT"
              UID=0
              GID=0

*GROUPS
GROUP1
        LMID=SITE1  GRPNO=1
        TMSNAME=MQXA
        OPENINFO="MQSeries_XA_RMI:MYQUEUEMANAGER"

*SERVERS
DEFAULT: CLOPT="-A -- -m MYQUEUEMANAGER"

MQSERV1     SRVGRP=GROUP1 SRVID=1
MQSERV2     SRVGRP=GROUP1 SRVID=2

*SERVICES
MPUT1
MGET1
MPUT2
MGET2
```

Rysunek 26. Przykład pliku ubbstxcn.cfg dla produktu WebSphere MQ for Windows

Uwaga: Zmień nazwy katalogów i ścieżki katalogów tak, aby były zgodne z instalacją. Zmień także nazwę menedżera kolejek MYQUEUEMANAGER na nazwę menedżera kolejek, z którym ma zostać nawiązane połączenie. Inne informacje, które należy dodać, są identyfikowane przez znaki <> .

Przykładowy plik ubbconfig dla produktu WebSphere MQ for Windows jest wymieniony w sekcji Rysunek 26 na stronie 170. Jest on dostarczany w postaci pliku ubbstxcn.cfg w katalogu przykładów produktu WebSphere MQ .

Przykładowy plik makefile (patrz Rysunek 27 na stronie 171) dostarczany z produktem WebSphere MQ for Windows nosi nazwę ubbstxmn.maki znajduje się w katalogu przykładów produktu WebSphere MQ .

```
TUXDIR  = f:\tuxedo
MQMDIR  = g:\Program Files\IBM\WebSphere MQ
APPDIR  = f:\tuxedo\apps\mqapp
MQMLIB  = $(MQMDIR)\tools\lib64
MQMINC  = $(MQMDIR)\tools\c\include
MQMSAMP = $(MQMDIR)\tools\c\samples
INC = -f "-I$(MQMINC) -I$(APPDIR)"
DBG = -f "/Zi"

amqstx.exe:
 $(TUXDIR)\bin\mkfldhdr    -d$(APPDIR) $(MQMSAMP)\amqstxvx.fld
 $(TUXDIR)\bin\viewc       -d$(APPDIR) $(MQMSAMP)\amqstxvx.v
 $(TUXDIR)\bin\buildtms    -o MQXA -r MQSeries_XA_RMI
 $(TUXDIR)\bin\buildserver -o MQSERV1 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT1:MPUT -s MGET1:MGET
 $(TUXDIR)\bin\buildserver -o MQSERV2 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT2:MPUT -s MGET2:MGET
 $(TUXDIR)\bin\buildclient -o doputs -f $(MQMSAMP)\amqstxpx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG)
 $(TUXDIR)\bin\buildclient -o dogets -f $(MQMSAMP)\amqstxgx.c \
                           -f $(MQMLIB)\mqm.lib $(INC) -v $(DBG)
 $(TUXDIR)\bin\tmloadcf    -y $(APPDIR)\ubbstxcn.cfg
```

Rysunek 27. Przykładowy plik makefile TUXEDO dla produktu WebSphere MQ for Windows

Przykładowy program serwera dla TUXEDO

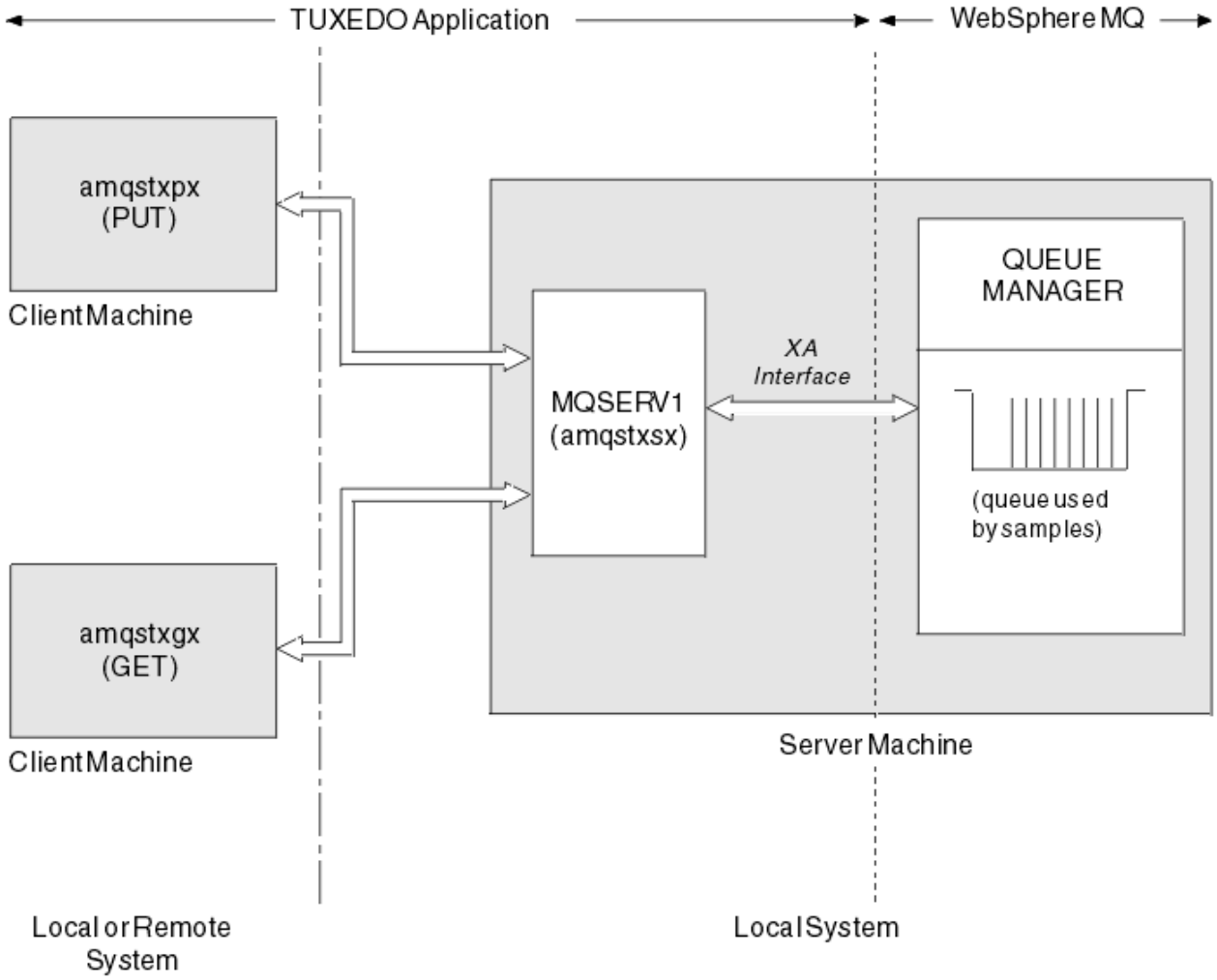
Przykładowy program serwera (amqstxsx) jest przeznaczony do uruchamiania z przykładowymi programami Put (amqstxpx.c) i Get (amqstxgx.c). Przykładowy program serwera jest uruchamiany automatycznie po uruchomieniu TUXEDO.

Uwaga: Należy uruchomić menedżer kolejek ***przed*** , należy uruchomić TUXEDO.

Przykładowy serwer udostępnia dwie usługi TUXEDO: MPUT1 i MGET1:

- Usługa MPUT1 jest sterowana przez przykład PUT i korzysta z komendy MQPUT1 w punkcie synchronizacji w celu umieszczenia komunikatu w jednostce pracy sterowanej przez TUXEDO. Pobiera ona parametry QName i tekst komunikatu, które są dostarczane przez przykład PUT.
- Usługa MGET1 zostanie otwarta i zamyka kolejkę za każdym razem, gdy zostanie wyświetlony komunikat. Pobiera ona parametry QName i tekst komunikatu, które są dostarczane przez przykład GET.

Wszystkie komunikaty o błędach, kody przyczyny i komunikaty o statusie są zapisywane w pliku dziennika TUXEDO.



Rysunek 28. Jak próbki TUXEDO współpracują ze sobą

Umieść przykładowy program dla TUXEDO

Ten przykład umożliwia wielokrotne umieszczanie komunikatu w kolejce w partiach, demonstrując za pomocą TUXEDO funkcję syncwskazujący jako menedżera zasobów.

Przykładowy program serwera `amqstxsx` musi być uruchomiony, aby próba umieszczenia przykładu powiodła się; przykładowy program serwera łączy się z menedżerem kolejek i używa interfejsu XA. Aby uruchomić przykład, wpisz:

- `doputs -n queuename -b batchsize -c trancount -t message`

Na przykład:

- `doputs -n myqueue -b 5 -c 6 -t "Hello World"`

Spowoduje to umieszczenie 30 komunikatów w kolejce o nazwie `myqueue`, w sześciu partiach, z których każda zawiera pięć komunikatów. Jeśli wystąpią jakiekolwiek problemy, wycofuje ona zadanie wsadowe komunikatów, w przeciwnym razie je zatwierdzije.

Wszystkie komunikaty o błędach są zapisywane w pliku dziennika TUXEDO i do standardowego wyjścia błędów. Wszelkie kody przyczyny są zapisywane w stderr.

Pobierz przykład dla TUXEDO

Ten przykład umożliwia pobieranie komunikatów z kolejki w zadaniach wsadowych.

Przykładowy program serwera `amqstxsx` musi być uruchomiony, aby próba umieszczenia przykładu powiodła się; przykładowy program serwera łączy się z menedżerem kolejek i używa interfejsu XA. Aby uruchomić przykład, wpisz:

- `dogets -n queuename -b batchsize -c trancount`

Na przykład:

- `dogets -n myqueue -b 6 -c 4`

Spowoduje to wyłączenie 24 komunikatów z kolejki o nazwie `myqueue`, w sześciu partiach, z których każda zawiera cztery komunikaty. W przypadku uruchomienia tego przykładu po umieszczonym przykładzie, który umieszcza 30 komunikatów w systemie `myqueue`, w systemie `myqueue`będzie mieć tylko sześć komunikatów. Liczba zadań wsadowych i wielkość zadania wsadowego mogą być różne między umieszczanie komunikatów i ich pobieranie.

Wszystkie komunikaty o błędach są zapisywane w pliku dziennika TUXEDO i do standardowego wyjścia błędów. Wszelkie kody przyczyny są zapisywane w stderr.

Korzystanie z wyjścia zabezpieczeń SSPI w systemach Windows

W tej sekcji opisano sposób korzystania z programów obsługi wyjścia kanału SSPI w systemach Windows . Podany kod wyjścia znajduje się w dwóch formatach: obiektowym i źródłowym.

Kod obiektu

Plik z kodem obiektu nosi nazwę amqrspin.dll. Zarówno dla klienta, jak i dla serwera, jest on instalowany jako standardowy element produktu WebSphere MQ for Windows w folderze produktu *MQ_INSTALLATION_PATH*`/exits/`*INSTALLATION_NAME* . Na przykład: `C:\Program Files\IBM\WebSphere MQ\exits\installation2`. Jest on ładowany jako standardowe wyjście użytkownika. Można uruchomić dostarczone wyjście kanału zabezpieczeń i użyć usług uwierzytelniania w definicji kanału.

W tym celu należy określić jedną z następujących wartości:

```
SCYEXIT('amqrspin(SCY_KERBEROS)')
SCYEXIT('amqrspin(SCY_NTLM)')
```

Aby udostępnić obsługę dla kanału zastrzeżonego, należy określić następujące informacje na kanale SVRCONN:

```
SCYDATA('remote_principal_name')
```

gdzie *nazwa_zdalnej_principal_name* jest w postaci DOMAIN\user. Bezpieczny kanał jest ustanawiany tylko wtedy, gdy nazwa zdalnego użytkownika jest zgodna z nazwą *nazwa_zdalnej_principalna_nazwa*.

Aby użyć dostarczonych programów obsługi wyjścia kanału między systemami, które działają w obrębie domeny zabezpieczeń Kerberos , należy utworzyć nazwę servicePrincipaldla menedżera kolejek.

Kod źródłowy

Plik kodu źródłowego wyjścia ma nazwę amqsspin.c. Jest on dostępny w produkcie `C:\Program Files\IBM\WebSphere MQ\Tools\c\Samples`.

W przypadku zmodyfikowania kodu źródłowego należy ponownie skompilować zmodyfikowane źródło.

Należy je skompilować i połączyć w taki sam sposób, jak inne wyjście kanału dla odpowiedniej platformy, z tą różnicą, że nagłówki SSPI muszą być dostępne w czasie kompilacji, a biblioteki bezpieczeństwa SSPI, wraz z dowolnymi rekomendowanymi powiązanymi bibliotekami, muszą być dostępne w czasie połączenia.

Przed wykonaniem poniższej komendy należy upewnić się, że plik cl.exeoraz biblioteka Visual C++ oraz folder `include` są dostępne w ścieżce użytkownika. Na przykład:

```
cl /VERBOSE /LD /MT /I<path_to_Microsoft_platform_SDK\include>
/I<path_to_WebSphere MQ\tools\c\include> amqsspin.c /DSECURITY_WIN32
-link  /DLL /EXPORT:SCY_KERBEROS /EXPORT:SCY_NTLM STACK:8192
```

Uwaga: Kod źródłowy nie zawiera żadnego przepisu dotyczącego śledzenia lub obsługi błędów. W przypadku zmodyfikowania i użycia kodu źródłowego należy dodać własne procedury śledzenia i obsługi błędów.

Uruchamianie przykładów przy użyciu kolejek zdalnych

Istnieje możliwość zademonstrowania zdalnego kolejkowania przez uruchomienie przykładów w połączonych menedżerach kolejek.

Program amqscos0.tst udostępnia lokalną definicję kolejki zdalnej (SYSTEM.SAMPLE.REMOTE), w którym używany jest zdalny menedżer kolejek o nazwie OTHER. Aby użyć tej definicji przykładu, należy zmienić wartość inną na nazwę drugiego menedżera kolejek, który ma być używany. Należy również skonfigurować kanał komunikatów między dwoma menedżerami kolejek. Aby uzyskać informacje na temat sposobu wykonania tej czynności, należy zapoznać się z informacjami w sekcji Definiowanie kanałów.

Przykładowe programy żądania umieją umieścić własną nazwę lokalnego menedżera kolejek w polu *ReplyToQMgr* wysyłanych przez nich komunikatów. Przykłady Inquire i Set wysyłają komunikaty odpowiedzi do kolejki i menedżera kolejek komunikatów o nazwie w polach *ReplyToQ* i *ReplyToQMgr* komunikatów żądania, które przetwarzali.

Przykładowy program monitorowania kolejki klastra (AMQSCLM)

W tym przykładzie wykorzystano wbudowane funkcje równoważenia obciążenia klastra IBM WebSphere MQ w celu kierowania komunikatów do instancji kolejek, które używają przyłączonych aplikacji. Ten automatyczny kierunek zapobiega gromadzeniu komunikatów w instancji kolejki klastra, do której nie jest przyłączona aplikacja konsumująca.

Przegląd

Istnieje możliwość skonfigurowania klastra, który ma więcej niż jedną definicję dla tej samej kolejki w różnych menedżerach kolejek. Ta konfiguracja zapewnia korzyści ze zwiększonej dostępności i równoważenia obciążenia. Nie ma jednak możliwości wbudowanej w program IBM WebSphere MQ możliwości dynamicznego modyfikowania dystrybucji komunikatów w klastrze w oparciu o stan dołączonych aplikacji. Z tego powodu aplikacja konsumująca zawsze musi być przyłączona do każdej instancji kolejki, aby zapewnić, że komunikaty będą przetwarzane.

Przykładowy program monitorujący kolejkę klastra monitoruje stan dołączonych aplikacji. Program dynamicznie dopasowuje wbudowaną konfigurację równoważenia obciążenia w celu kierowania komunikatów do instancji kolejki klastrowej z dołączonymi aplikacjami wykorzystudzonymi. W pewnych sytuacjach program ten może być używany do odprężania potrzeb aplikacji konsumowanych, aby zawsze były połączone z każdą instancją kolejki. Powoduje również ponowne wysyłanie komunikatów, które zostały umieszczone w kolejce w instancji kolejki, w której nie są przyłączone żadne aplikacje korzystające z aplikacji. Ponowne wysyłanie komunikatów umożliwia skierowanie komunikatów wokół aplikacji konsumowania, która jest tymczasowo zamknięta.

Program jest przeznaczony do użycia w przypadku, gdy aplikacje korzystające z pracy są aplikacjami długotrwałych, a nie często dołączaniem i odłączaniem aplikacji.

Przykładowy program monitorujący kolejkę klastra jest skompilowanym programem wykonywalnym przykładowego pliku C `amqsclma.c`.

Więcej informacji na temat klastrów i obciążenia można znaleźć w sekcji Używanie klastrów do zarządzania obciążeniem .

AMQSCLM: Projektowanie i planowanie korzystania z przykładu

Informacje na temat sposobu działania przykładowego programu monitorującego kolejkę klastra, punktów, które należy wziąć pod uwagę podczas konfigurowania systemu dla przykładowego programu, który ma być uruchomiony, oraz modyfikacji, jakie można wprowadzić w przykładowym kodzie źródłowym.

Projektowanie

Przykładowy program monitorujący kolejkę klastra monitoruje lokalne kolejki klastrowe, które używają przyłączonych aplikacji. Program monitoruje kolejki określone przez użytkownika. Nazwa kolejki może być konkretna, na przykład APP.TEST01, lub nazwa ogólna. Nazwy ogólne muszą być w formacie zgodnym z PCF (Programmable Command Format). Przykłady nazw ogólnych to APP.TEST*lub APP*.

Każdy menedżer kolejek w klastrze, który jest właścicielem instancji kolejki lokalnej, która ma być monitorowana, wymaga, aby program przykładowy monitorowania kolejki klastra był połączony z tym programem.

Dynamiczne kierowanie komunikatów

Przykładowy program monitorujący kolejkę klastra używa wartości **IPPROCS** (open for input process count) kolejki w celu określenia, czy ta kolejka ma konsumentów. Wartość większa od 0 wskazuje, że kolejka ma przyłączoną co najmniej jedną aplikację konsumująca. Takie kolejki są aktywne. Wartość 0 oznacza, że w kolejce nie są przyłączone programy korzystające z pamięci. Takie kolejki są nieaktywne.

W przypadku kolejki klastrowej z wieloma instancjami w klastrze produkt WebSphere MQ używa właściwości priorytetu obciążenia klastra **CLWLPRTY** każdej instancji kolejki w celu określenia instancji, do których mają być wysyłane komunikaty. Produkt WebSphere MQ wysyła komunikaty do dostępnych instancji kolejki o największej wartości **CLWLPRTY** .

Przykładowy program monitorujący kolejkę klastra aktywuje kolejkę klastra przez ustawienie lokalnej wartości **CLWLPRTY** na 1. Program dezaktywuje kolejkę klastra przez ustawienie jej wartości **CLWLPRTY** na 0.

Technologia klastrowa produktu WebSphere MQ propaguje zaktualizowaną właściwość **CLWLPRTY** kolejki klastrowej do wszystkich odpowiednich menedżerów kolejek w klastrze. Na przykład składnia

- Menedżer kolejek z podłączonym aplikacją, która umieszcza komunikaty w kolejce.
- Menedżer kolejek, który jest właścicielem kolejki lokalnej o tej samej nazwie w tym samym klastrze.

Propagacja jest wykonywana przy użyciu menedżerów kolejek pełnego repozytorium klastra. Nowe komunikaty dla kolejki klastra są kierowane do instancji o największej wartości **CLWLPRTY** w obrębie klastra.

Przesyłanie komunikatów w kolejce

Dynamiczna modyfikacja wartości **CLWLPRTY** wpływa na kierowanie nowych komunikatów. Ta dynamiczna modyfikacja nie ma wpływu na komunikaty znajdujące się już w kolejce w instancji kolejki bez przyłączonych konsumentów lub komunikaty, które przeszły mechanizm równoważenia obciążenia przed propagowaniem zmodyfikowanej wartości produktu **CLWLPRTY** przez klaster. W wyniku tego komunikaty pozostają w żadnej nieaktywnej kolejce i nie są przetwarzane przez aplikację konsumująca. Aby rozwiązać ten problem, przykładowy program monitorujący kolejkę klastra jest w stanie pobrać komunikaty z lokalnej kolejki bez konsumentów i wysłać te komunikaty do zdalnych instancji tej samej kolejki, w której są przyłączone konsumenty.

Przykładowy program monitorujący kolejkę klastra przesyła komunikaty z nieaktywnej kolejki lokalnej do jednej lub większej liczby aktywnych kolejek zdalnych, pobierając komunikaty (za pomocą **MQGET**) i umieszczając komunikaty (za pomocą programu **MQPUT**) do tej samej kolejki klastrowej. Ten transfer powoduje, że zarządzanie obciążeniem klastra WebSphere MQ wybiera inną instancję docelową w oparciu o wyższą wartość CLWLPRTY niż wartość instancji lokalnej kolejki. Trwałość komunikatu i kontekst są zachowywane podczas przesyłania komunikatów. Kolejność komunikatów i wszelkie opcje powiązania nie są zachowywane.

Planowanie

Przykładowy program monitorujący kolejkę klastra modyfikuje konfigurację klastra, gdy istnieje zmiana w połączeniach aplikacji konsumujących. Modyfikacje są przesyłane z menedżerów kolejek, w których przykładowy program monitorujący kolejkę klastra monitoruje kolejki, do pełnych menedżerów kolejek repozytorium w klastrze. Menedżery kolejek pełnego repozytorium przetwarają aktualizacje konfiguracji i ponownie je wznawiają we wszystkich odpowiednich menedżerach kolejek w klastrze. Odpowiednie menedżery kolejek obejmują te menedżery kolejek, które są właścicielkami kolejek klastrowych o tej samej nazwie (gdzie działa instancja programu przykładowego monitorowania kolejek klastra), oraz dowolny menedżer kolejek, w którym aplikacja otworzyła kolejkę klastra w celu umieszczenia komunikatów w kolejce w ciągu ostatnich 30 dni.

Zmiany są przetwarzane asynchronicznie w klastrze. Oznacza to, że po każdej zmianie różne menedżery kolejek w klastrze mogą mieć różne widoki konfiguracji na określony czas.

Przykładowy program monitorujący kolejkę klastra jest odpowiedni tylko dla systemów, w których aplikacje używają rzadko podłączanych lub odłączanych aplikacji, na przykład długotrwałe aplikacje korzystające z pamięci. Jeśli używane do monitorowania systemów, w których aplikacje konsumowane są przyłączane tylko przez krótkie okresy, opóźnienie związane z dystrybucją aktualizacji konfiguracji może spowodować, że menedżery kolejek w klastrze będą mieć niepoprawny widok kolejek, do których przyłączone są konsumenci. To opóźnienie może spowodować niepoprawne kierowanie komunikatów.

W przypadku monitorowania wielu kolejek stosunkowo niski wskaźnik zmian w przyłączonych konsumentach we wszystkich kolejkach może zwiększyć ruch konfiguracji klastra w klastrze. Zwiększone natężenie ruchu w konfiguracji klastra może spowodować nadmierne obciążenie jednego lub większej liczby następujących menedżerów kolejek.

- Menedżery kolejek, w których uruchomiony jest przykładowy program monitorowania kolejki klastra
- Menedżery kolejek pełnego repozytorium
- Menedżer kolejek z połączonym aplikacją, która umieszcza komunikaty w kolejce.
- Menedżer kolejek, który jest właścicielem kolejki lokalnej o tej samej nazwie w tym samym klastrze

Należy ocenić użycie procesora w pełnej kolejce menedżerów kolejek repozytorium. Dodatkowe wykorzystanie procesora jest widoczne jako ruch komunikatów w pełnej kolejce repozytorium SYSTEM.CLUSTER.COMMAND.QUEUE. Jeśli komunikaty są zbudowane w tej kolejce, oznacza to, że menedżery kolejek pełnego repozytorium nie mogą nadążać za szybkością zmiany konfiguracji klastra w systemie.

Jeśli wiele kolejek jest monitorowanych przez program przykładowy monitorujący kolejkę klastra, jest to ilość pracy wykonywana przez program przykładowy i menedżer kolejek. Praca ta jest wykonywana, nawet wtedy, gdy nie ma żadnych zmian w przyłączonych konsumentach. Argument **-i** można zmodyfikować, aby zmniejszyć użycie procesora w programie przykładowym w systemie lokalnym, zmniejszając częstotliwość cyklu monitorowania.

Aby pomóc w wykrywaniu nadmiernego działania, przykładowy program monitorujący kolejkę klastra zgłasza średni czas przetwarzania dla każdego okresu odpytywania, czas przetwarzania i liczbę zmian w konfiguracji. Raporty są dostarczane w komunikacie informacyjnym, **CLM0045I**, co 30 minut lub co 600 interwałów odpytywania, w zależności od tego, co nastąpi wcześniej.

Wymagania dotyczące użycia monitorowania kolejki klastra

Przykładowy program monitorujący kolejkę klastra ma wymagania i ograniczenia. Można zmodyfikować przykładowy kod źródłowy, aby zmienić niektóre z tych ograniczeń, w jaki sposób może być używany. Przykłady wymienione w tej sekcji szczegółów modyfikacji, które mogą zostać wprowadzone.

- Przykładowy program monitorujący kolejkę klastra jest przeznaczony do monitorowania kolejek, w których aplikacje konsumujące są przyłączone lub nie są przyłączone. Jeśli w systemie są używane aplikacje, które często są przyłączane i odłączane, przykładowy program może generować nadmierną aktywność konfiguracji klastra w całym klastrze. Może to mieć wpływ na wydajność menedżerów kolejek w klastrze.

- Przykładowy program monitorujący kolejkę klastra jest zależny od bazowego systemu WebSphere MQ i technologii klastrowych. Liczba monitorowanych kolejek, częstotliwość monitorowania oraz częstotliwość zmian stanu poszczególnych kolejek wpływa na obciążenie systemu. Czynniki te należy wziąć pod uwagę przy wybieraniu kolejek, które mają być monitorowane, oraz przedział czasu odpytywania w monitorowaniu.
- Instancja przykładowego programu monitorowania kolejki klastra musi być połączona z każdym menedżerem kolejek w klastrze, który jest właścicielem instancji kolejki, która ma być monitorowana. Nie jest konieczne łączenie przykładowego programu z menedżerami kolejek w klastrze, które nie są właścicielkami kolejek.
- Przykładowy program monitorujący kolejkę klastra musi być uruchomiony z odpowiednimi uprawnieniami dostępu do wszystkich wymaganych zasobów WebSphere MQ . Na przykład składnia
  - Menedżer kolejek, z którym ma zostać nawiązane połączenie
  - SYSTEM.ADMIN.COMMAND.QUEUE
  - Wszystkie kolejki, które mają być monitorowane, gdy wykonywane jest przesyłanie komunikatów
- Serwer komend musi być uruchomiony dla każdego menedżera kolejek z podłączonym programem przykładowym monitorowania kolejki klastra.
- Każda instancja przykładowego programu monitorowania kolejki klastra wymaga wyłącznego użycia lokalnej (nieklastrowej) kolejki w menedżerze kolejek, z którą jest on połączony. Ta kolejka lokalna jest używana do sterowania programem przykładowym i do odbierania komunikatów odpowiedzi z zapytań wykonanych na serwerze komend menedżera kolejek.
- Wszystkie kolejki, które mają być monitorowane przez pojedynczą instancję przykładowego programu monitorowania kolejki klastra, muszą znajdować się w tym samym klastrze. Jeśli menedżer kolejek zawiera kolejki w wielu klastrach, które wymagają monitorowania, wymagane jest wiele instancji programu przykładowego. Każda instancja wymaga lokalnej kolejki na potrzeby komunikatów sterujących i odpowiedzi.
- Wszystkie kolejki, które mają być monitorowane, muszą znajdować się w jednym klastrze. Kolejki skonfigurowane do korzystania z listy nazw klastrów nie są monitorowane.
- Włączenie przesyłania komunikatów z nieaktywnych kolejek jest opcjonalne. Odnosi się do wszystkich kolejek monitorowanych przez instancję przykładowego programu monitorowania kolejki klastra. Jeśli tylko podzbiór monitorowanych kolejek wymaga włączenia przesyłania komunikatów, konieczne jest zastosowanie dwóch instancji przykładowego programu monitorowania kolejki klastra. Jeden program przykładowy ma włączone przesyłanie komunikatów, a drugi ma wyłączony transfer komunikatów. Każda instancja programu przykładowego wymaga kolejki lokalnej na potrzeby komunikatów sterujących i odpowiedzi.
- Funkcja równoważenia obciążenia klastra WebSphere MQ domyślnie wysyła komunikaty do instancji kolejek klastrowych znajdujących się w tym samym menedżerze kolejek, z którym połączona jest aplikacja umieszczana w klastrze. Ta opcja musi być wyłączona, gdy kolejka lokalna jest nieaktywna w następujących okolicznościach:
  - Wprowadzanie aplikacji łączy się z menedżerami kolejek, które są właścicielkami nieaktywnej kolejki, która jest monitorowana
  - Komunikaty w kolejce są przesyłane z nieaktywnych kolejek do aktywnych kolejek.

  Lokalne preferencje równoważenia obciążenia w kolejce można wyłączyć statycznie, ustawiając wartość parametru CLWLUSEQ na ANY. W tej konfiguracji komunikaty umieszczane w kolejkach lokalnych są dystrybuowane do lokalnych i zdalnych instancji kolejek w celu zrównoważenia obciążenia, nawet jeśli istnieją lokalne aplikacje korzystające z pamięci masowej. Alternatywnie, przykładowy program monitorujący kolejkę klastra można skonfigurować w taki sposób, aby tymczasowo ustawiał wartość **CLWLUSEQ** na wartość ANY , podczas gdy kolejka nie ma przyłączonych konsumentów, co powoduje, że tylko komunikaty lokalne będą wysyłane do lokalnych instancji kolejki, gdy ta kolejka jest aktywna.
- System i aplikacje produktu WebSphere MQ nie mogą używać produktu **CLWLPRTY** dla kolejek, które mają być monitorowane, ani kanałów, które są używane. W przeciwnym razie działania przykładowego programu monitorowania kolejki klastra w atrybutach kolejki produktu **CLWLPRTY** mogą mieć niepożądane efekty.

- Przykładowy program monitorujący kolejkę klastra rejestruje informacje o środowisku wykonawczym w zestawie plików raportów. Katalog do przechowywania tych raportów jest wymagany, a przykładowy program monitorujący kolejkę klastra musi mieć uprawnienia do zapisu w tym programie.

AMQSCLM: Przygotowywanie i uruchamianie przykładu

Aby uruchomić przykład monitorowania kolejki klastra, należy skonfigurować menedżer kolejek w taki sposób, aby bezpiecznie akceptować przychodzące żądania połączeń z aplikacji działających w trybie klienta.

Zanim rozpoczniesz

Przed uruchomieniem przykładu monitorowania kolejki klastra należy wykonać następujące kroki.

1. Utwórz kolejkę roboczą dla każdego menedżera kolejek w celu wewnętrznego użycia przykładu.

   Każda instancja przykładu wymaga lokalnej kolejki nieklastrowego do wyłącznego użytku wewnętrznego. Można wybrać nazwę kolejki. W przykładzie użyto nazwy `AMQSCLM.CONTROL.QUEUE`. Na przykład w systemie Windowstę kolejkę można utworzyć za pomocą komendy **MQSC** .

   ```
   DEFINE QLOCAL(AMQSCLM.CONTROL.QUEUE)
   ```

   Wartości parametrów **MAXDEPTH** i **MAXMSGL** można pozostawić jako domyślne.

2. Utwórz katalog dla dzienników komunikatów o błędach i informacji.

   W tym przykładzie komunikaty diagnostyczne są zapisywane w plikach raportów. Należy wybrać katalog, w którym mają być przechowywane pliki. Na przykład w systemie Windowsmożna utworzyć katalog za pomocą następującej komendy:

   ```
   mkdir C:\AMQSCLM\rpts
   ```

   Pliki raportów utworzone przez przykład mają następującą konwencję nazewnictwa:

   ```
   QmgrName.ClusterName.RPT0n.LOG
   ```

3. (Opcjonalnie) Zdefiniuj przykład monitorowania kolejki klastra jako usługę IBM WebSphere MQ .

   Aby monitorować kolejki, próbka musi być zawsze uruchomiona. Aby upewnić się, że próbka monitorowania kolejki klastra jest zawsze uruchomiona, można zdefiniować przykład jako usługę menedżera kolejek. Zdefiniowanie przykładu jako usługi oznacza, że menedżer AMQSCLM jest uruchamiany podczas uruchamiania menedżera kolejek. W celu zdefiniowania przykładu monitorowania kolejki klastra jako usługi produktu IBM WebSphere MQ można użyć następującego przykładu **RUNMQSC** .

   ```
   define service(AMQSCLM) +
       descr('Active Cluster Queue Message Distribution Monitor - AMQSCLM') +
       control(qmgr) +
       servtype(server) +
       startcmd('<Install Root>\tools\c\samples\Bin\AMQSCLM.exe') +
    startarg('-m +QMNAME+ -c CLUSTER1 -q ABC* -r AMQSCLM.CONTROL.QUEUE -l c:\AMQSCLM\rpts') +
    stdout('C:\AMQSCLM\rpts\+QMNAME+.TSTCLUS.stdout.log') +
    stderr('C:\AMQSCLM\rpts\+QMNAME+.TSTCLUS.stderr.log')
   ```

   gdzie < Install Root > jest położeniem instalacji.

Definicja	Opis
service	Określa nazwę usługi. Można wybrać nazwę usługi.
descr	Określa tekstowy opis usługi.
control	Wskazuje, że usługa jest uruchamiana i zatrzymana w tym samym czasie co menedżer kolejek.

Definicja	Opis
servtype	Wskazuje, że obiekt usługi serwera, co oznacza tylko jedną instancję, może być wykonywany w danym momencie dla tego menedżera kolejek.
startcmd	Określa lokalizację i nazwę programu.
startarg	Określa argumenty próbki. Należy zwrócić uwagę na użycie parametru *+ QMNAME +*. Nazwa menedżera kolejek jest zastępowana automatycznie.
stdout	Pełna nazwa pliku, do którego przekierowuje się standardowe wyjście danych. Próbka zapisuje do tego pliku tylko komunikaty potwierdzające, że próba została zakończona. Przykład ten jest taki, ponieważ standardowy plik błędów został już zamknięty we wcześniejszej fazie przykładowego procesu zakończenia.
stderr	Pełna nazwa pliku, do którego przekierowane są standardowe wyjście błędów. Próbka zapisuje komunikaty o błędach w standardowym pliku błędów przed zakończeniem próby.

O tym zadaniu

To zadanie umożliwia uruchamianie i zatrzymywanie przykładu monitorowania kolejki klastra na różne sposoby. Umożliwia on również uruchamianie przykładu w trybie generującym pliki raportów zawierające informacje statystyczne na temat monitorowanych kolejek.

Program przykładowy można uruchomić za pomocą następującej komendy.

```
AMQSCLM -m QMgrName -c ClusterName (-q QNameMask | -f QListFile) -r MonitorQName
[-l ReportDir] [-t] [-u ActiveVal] [-i Interval] [-d] [-s] [-v]
```

Tabela zawiera listę argumentów, które mogą być używane wraz z próbką monitorowania kolejki klastra wraz z dodatkowymi informacjami na temat każdego z nich.

Argument	Zmienna	Dalsze informacje
-m	**QMgrName**	Menedżer kolejek do monitorowania.
-c	**ClusterName**	Klaster zawierający kolejki do monitorowania.
-q	**QNameMask**	Kolejka lub kolejki do monitorowania. Końcowy * monitoruje wszystkie kolejki o nazwach, które są zgodne z zero lub więcej znaków końcowych.
-f	**QListFile**	Pełna ścieżka i nazwa pliku zawierającego listę nazw kolejek masek nazwy kolejki, które mają być monitorowane. Plik musi zawierać jedną nazwę kolejki/maskę na wiersz. Można określić wartość **-q** lub **-f**, ale nie obie jednocześnie.
-r	**MonitorQName**	Kolejka lokalna używana wyłącznie przez przykład.
-l	**ReportDir**	Ścieżka do katalogu, w którym mają być zapisywane rejestrowane komunikaty informacyjne w zestawie opakowywania < fn> Dla każdego menedżera kolejek i kombinacji kolejek generowany jest plik raportu, który jest limitowany do określonej wielkości. Program rejestrujący zawsze zapisuje w tym samym pliku, ale zachowuje również dwie poprzednie wersje pliku. < /fn> report files.
-t		(Opcjonalne) Umożliwia przesyłanie komunikatów w kolejce z nieaktywnych kolejek lokalnych do aktywnych kolejek. Jeśli ta opcja nie jest włączona, tylko nowe komunikaty wprowadzane do klastra są dynamicznie kierowane do aktywnych instancji kolejki.
-u	**ActiveVal**	(Opcjonalnie) Automatycznie przełącza właściwość **CLWLUSEQ** monitorowanej instancji kolejki na ANY , gdy jest ona nieaktywna, oraz do właściwości **ActiveVal** , gdy jest aktywna. **ActiveVal** może mieć wartość LOCAL lub QMGR. Jeśli ten argument nie zostanie ustawiony w systemie, w którym aplikacje łączą się z tym samym menedżerem kolejek lub jeśli włączono przesyłanie komunikatów, to monitorowane kolejki muszą mieć wartość **CLWLUSEQ** ANYlub QMGR z menedżerem kolejek, który ma wartość ANY.
-i	**Interval**	(Opcjonalnie) Przedział czasu (w sekundach), w którym monitor sprawdza kolejki. Wartość domyślna to 300 sekund (5 minut).
-d		(Opcjonalnie) Włącza dodatkowe wyjście diagnostyczne. Dane wyjściowe debugowania mogą być przydatne podczas początkowego konfigurowania systemu lub podczas pracy z przykładowym kodem.
-s		(Opcjonalnie) Włącza minimalny wynik statystyczny na przedział czasu.
-v		(Opcjonalnie) W uzupełnieniu do plików raportu, informacje o raportach dziennika są dostępne w programie `standard out`.

Przykłady listy argumentów:

```
-m QMGR1 -c CLUS1 -f c:\QList.txt -r CLMQ -l c:\amqsclm\rpts -s
-m QMGR2 -c CLUS1 -q ABC* -r CLMQ -l c:\amqsclm\rpts -i 600
-m QMGR1 -c CLUSDEV -q QUEUE.* -r CLMQ -l c:\amqsclm\rpts -t -u QMGR -d
```

Przykładowy plik listy kolejek:

```
Q1
QUEUE.*
ABC
ABD
```

Procedura

1. Uruchom próbkę monitorowania kolejki klastra. Przykład można uruchomić w jeden z następujących sposobów:
   - Użyj wiersza komend z odpowiednimi autoryzacjami użytkownika.
   - Użyj komendy MQSC **START SERVICE** , jeśli przykład został skonfigurowany jako usługa IBM WebSphere MQ .

   Lista argumentów jest taka sama w obu przypadkach.

   Próbka nie uruchamia monitorowania kolejek przez 10 sekund po zainicjowaniu programu. To opóźnienie umożliwia aplikacjom konsumowanie najpierw połączenia się z monitorowanymi kolejkami, zapobiegając niepotrzebnym zmianom stanu aktywnego kolejki.
2. Zatrzymaj przykład monitorowania kolejki klastra. Przykład jest automatycznie zatrzymywany, gdy menedżer kolejek jest zatrzymany, zatrzymywany, wygaszany lub jeśli połączenie z menedżerem kolejek jest zerwane. Istnieją sposoby zatrzymania próby bez zakończenia menedżera kolejek:
   - Skonfiguruj kolejkę lokalną używaną wyłącznie przez przykład, aby wyłączyć funkcję Get.
   - Wyślij komunikat z **CorrelId** o wartości `"STOP CLUSTER MONITOR\0\0\0\0"`do kolejki lokalnej używanej wyłącznie przez przykład.
   - Zakończ proces przykładowy. Może to spowodować utratę nietrwałych komunikatów przesyłanych do aktywnych kolejek. Może to również spowodować, że kolejka lokalna używana przez próbkę jest wstrzymana przez liczbę sekund po zakończeniu. Ta sytuacja uniemożliwia natychmiastowe uruchomienie nowej instancji przykładu monitorowania kolejki klastra.

   Jeśli próbka została uruchomiona jako usługa IBM WebSphere MQ , produkt **STOP SERVICE** nie ma żadnego efektu. Możliwe jest użycie jednej z metod zakończenia opisanych jako skonfigurowany mechanizm **STOP SERVICE** w menedżerze kolejek.

Co dalej

Sprawdź status przykładu.

Jeśli raportowanie jest włączone, można przejrzeć pliki raportów w celu uzyskania statusu. Użyj następującej komendy, aby przejrzeć najbardziej aktualny plik raportu.

```
QMgrName.ClusterName.RPT01.LOG
```

Aby przejrzeć starsze pliki raportów, należy użyć następujących komend.

```
QMgrName.ClusterName.RPT02.LOG
QMgrName.ClusterName.RPT03.LOG
```

Wielkość plików raportu może wynosić maksymalnie ok. 1 MB. Gdy plik `RPT01` zostanie zapełniony, tworzony jest nowy plik `RPT01` . Nazwa starego pliku `RPT01` zostanie zmieniona na `RPT02`. Nazwa `RPT02` została zmieniona na `RPT03`. Stary `RPT03` jest odrzucany.

Przykład tworzy komunikaty informacyjne w następujących sytuacjach:

- Przy starcie
- Przy wypowiedzeniu
- Gdy oznacza kolejkę **ACTIVE** lub **INACTIVE**
- w przypadku ponownego przesłania komunikatów z nieaktywnej kolejki do aktywnej instancji lub instancji

W przykładzie zostanie utworzony komunikat o błędzie *CLMnnnnE* w celu zgłoszenia problemu, który wymaga uwagi.

Co 30 minut próbka raportuje średni czas przetwarzania dla przedziału czasu odpytywania i czas przetwarzania. Te informacje są przechowywane w komunikacie CLM0045I.

Gdy komunikaty statystyczne są włączone **-s**, przykładowy raport przedstawia następujące informacje statystyczne o każdej kontroli kolejki:

- Czas przetwarzania kolejek (w milisekundach)
- Liczba sprawdzanych kolejek
- Liczba wprowadzonych zmian aktywnych/nieaktywnych
- Liczba przesłanych wiadomości

Te informacje są zgłaszane w komunikacie CLM0048I.

Pliki raportów mogą gwałtownie rosnąć w trybie debugowania i szybko zawijać. W tej sytuacji limit wielkości 1 MB dla poszczególnych plików może zostać przekroczony.

AMQSCLM: Rozwiązywanie problemów

W poniższych sekcjach znajdują się informacje o scenariuszach, które mogą wystąpić podczas korzystania z przykładu. Dostępne są informacje na temat potencjalnych wyjaśnień dotyczących scenariusza oraz opcje dotyczące sposobu jego rozwiązania.

Scenariusz: AMQSCLM nie jest uruchamiany

Potencjalne wyjaśnienie: Niepoprawna składnia.

Działanie: Sprawdź standardowe wyjście błędów dla poprawnej składni.

Potencjalne wyjaśnienie: Menedżer kolejek nie jest dostępny.

Działanie: Sprawdź, czy plik raportu ma identyfikator CLM0010E.

Potencjalne wyjaśnienie: Nie można otworzyć ani utworzyć pliku lub plików raportu.

Działanie: Podczas inicjowania sprawdź standardowe wyjście błędów dla komunikatów o błędach.

Scenariusz: AMQSCLM nie zmienia kolejki na ACTIVE lub INACTIVE

Potencjalne wyjaśnienie: Kolejka nie znajduje się na liście kolejek do monitorowania

Działanie: Sprawdź wartości parametrów **-q** i **-f** .

Potencjalne wyjaśnienie: Kolejka nie jest kolejką lokalną w poprawnym klastrze.

Działanie: Sprawdź, czy kolejka jest lokalna i czy w poprawnym klastrze.

Potencjalne wyjaśnienie: Menedżer AMQSCLM nie działa dla tego menedżera kolejek i klastra.

Działanie: Uruchom komendę AMQSCLM dla odpowiedniego menedżera kolejek i klastra.

Potencjalne wyjaśnienie: Kolejka jest pozostawiona NIEAKTYWNE, **CLWLPRTY**= 0, ponieważ nie ma ona konsumentów. Alternatywnie, pozostanie AKTYWNE **CLWLPRTY**> =1, ponieważ ma co najmniej 1 konsumenta.

Działanie: Sprawdź, czy aplikacje korzystające z aplikacji są przyłączone do kolejki.

Potencjalne wyjaśnienie: Serwer komend menedżera kolejek nie jest uruchomiony.

Działanie: Sprawdź, czy w plikach raportów nie występują błędy.

Scenariusz: komunikaty nie są kierowane przez kolejki INACTIVE

Potencjalne wyjaśnienie: Komunikaty są umieszczane bezpośrednio w menedżerze kolejek, który jest właścicielem nieaktywnej kolejki, a wartością parametru **CLWLUSEQ** kolejki jest inny niż ANY, a argument **-u** nie jest używany dla komendy AMQSCLM.

Działanie: Sprawdź wartość **CLWLUSEQ** odpowiedniego menedżera kolejek lub upewnij się, że argument **-u** jest używany dla AMQSCLM.

Potencjalne wyjaśnienie: W żadnym menedżerze kolejek nie ma aktywnych kolejek. Komunikaty są równomiernie zbalansowane we wszystkich nieaktywnych kolejkach aż do momentu, gdy kolejka stanie się aktywna.

Działanie: Sprawdź status kolejek we wszystkich menedżerach kolejek.

Potencjalne wyjaśnienie: Komunikaty są umieszczane w innym menedżerze kolejek w klastrze do tego, który jest właścicielem nieaktywnej kolejki, a zaktualizowana wartość **CLWLPRTY** równa 0 nie jest propagowana do menedżera kolejek umieszczanego aplikacji.

Działanie: Sprawdź, czy kanały klastra między monitorowanym menedżerem kolejek i pełnym menedżerem kolejek repozytorium są uruchomione. Sprawdź, czy kanały między umieszczonym menedżerem kolejek a menedżerem kolejek pełnego repozytorium są uruchomione. Sprawdź dzienniki błędów monitorowanych, umieszczających i pełnych menedżerów kolejek repozytorium.

Potencjalne wyjaśnienie: Instancje kolejek zdalnych są aktywne (CLWLPRTY=1), ale komunikaty nie mogą być kierowane do tych instancji kolejek, ponieważ kanał nadawczy klastra z lokalnego menedżera kolejek nie jest uruchomiony.

Działanie: Sprawdź status kanałów nadawczych klastra z lokalnego menedżera kolejek do zdalnego menedżera kolejek lub menedżerów z aktywną instancją kolejki.

Scenariusz: AMQSCLM nie przesyła komunikatów z nieaktywnej kolejki

Potencjalne wyjaśnienie: przesyłanie komunikatów nie jest włączone (**-t**).

Działanie: Upewnij się, że przesyłanie komunikatów jest włączone (**-t**).

Potencjalne wyjaśnienie: Kolejka nie znajduje się na liście kolejek, które mają być monitorowane.

Działanie: Sprawdź wartości parametrów **-q** i **-f** .

Potencjalne wyjaśnienie: W przypadku tego lub innych menedżerów kolejek w klastrze, które są właścicielkami tej samej kolejki, nie jest uruchomiony menedżer AMQSCLM.

Działanie: Uruchom komendę AMQSCLM.

Potencjalne wyjaśnienie: W kolejce jest **CLWLUSEQ**=LOCAL lub **CLWLUSEQ**=QMGR, a argument **-u** nie jest ustawiony.

Działanie: Ustaw parametr **-u** lub zmień konfigurację kolejki lub menedżera kolejek na wartość ANY.

Potencjalne wyjaśnienie: W klastrze nie ma aktywnych instancji kolejki.

Działanie: Sprawdź, czy w przypadku instancji kolejki wartość **CLWLPRTY**jest równa 1 lub większa.

Potencjalne wyjaśnienie: Instancje kolejek zdalnych mają konsumenty (**IPPROCS**> = 1), ale są nieaktywne w tych menedżerach kolejek (**CLWLPRTY**= 0), ponieważ AMQSCLM nie monitoruje tych zdalnych instancji.

Działanie: Upewnij się, że menedżer AMQSCLM jest uruchomiony w tych menedżerach kolejek i/lub kolejka znajduje się na liście kolejek, które mają być monitorowane, sprawdzając wartości parametrów **-q** i **-f** .

Potencjalne wyjaśnienie: Instancje kolejek zdalnych są aktywne (**CLWLPRTY**= 1), ale są nieaktywne w lokalnym menedżerze kolejek (**CLWLPRTY**= 0). Ta sytuacja wynika ze zaktualizowanej wartości **CLWLPRTY** , która nie jest propagowana do tego menedżera kolejek.

Działanie: Upewnij się, że zdalne menedżery kolejek są połączone z co najmniej jednym z menedżerów kolejek pełnego repozytorium w klastrze. Upewnij się, że menedżery kolejek pełnego repozytorium działają poprawnie. Sprawdź, czy kanały między pełnymi menedżerami kolejek repozytorium i monitorowanymi menedżerami kolejek są uruchomione.

Potencjalne wyjaśnienie: Komunikaty nie są zatwierdzane, dlatego nie są dostępne do pobrania.

Działanie: Sprawdź, czy aplikacja wysyłający działa poprawnie.

Potencjalne wyjaśnienie: AMQSCLM nie ma dostępu do kolejki lokalnej, w której komunikaty są umieszczane w kolejce.

Działanie: Może to być spowodowane tym, że program AMQSCLM nie jest uruchomiony jako użytkownik z wystarczającą autoryzacją do uzyskania dostępu do kolejki.

Potencjalne wyjaśnienie: Serwer komend menedżera kolejek nie jest uruchomiony.

Działanie: Uruchom serwer komend menedżera kolejek.

Potencjalne wyjaśnienie: Wystąpił błąd AMQSCLM.

Działanie: Sprawdź, czy w plikach raportów nie występują błędy.

Potencjalne wyjaśnienie: Instancje kolejki zdalnej są aktywne (CLWLPRTY=1), ale komunikaty nie mogą być przesyłane do tych instancji kolejek, ponieważ kanał nadawczy klastra z lokalnego menedżera kolejek nie jest uruchomiony. Często towarzyszy temu ostrzeżenie CLM0030W w dzienniku raportów amqsclm.

Działanie: Sprawdź status kanałów nadawczych klastra z lokalnego menedżera kolejek do zdalnego menedżera kolejek lub menedżerów z aktywną instancją kolejki.

Przykładowy program do wyszukiwania punktów końcowych połączenia (CEPL)

IBM WebSphere MQ Przykład wyszukiwania punktów końcowych połączenia udostępnia prosty, a jednocześnie wydajny moduł obsługi wyjścia, który oferuje użytkownikom produktu WebSphere MQ sposób pobierania definicji połączeń z repozytorium LDAP, takiego jak Tivoli Directory Server.

Aby można było używać programu CEPL, należy zainstalować klienta Tivoli Directory Server v6.3 Client.

Do korzystania z tego przykładu wymagana jest praktowa znajomość funkcji administrowania produktem WebSphere MQ na obsługiwanych platformach.

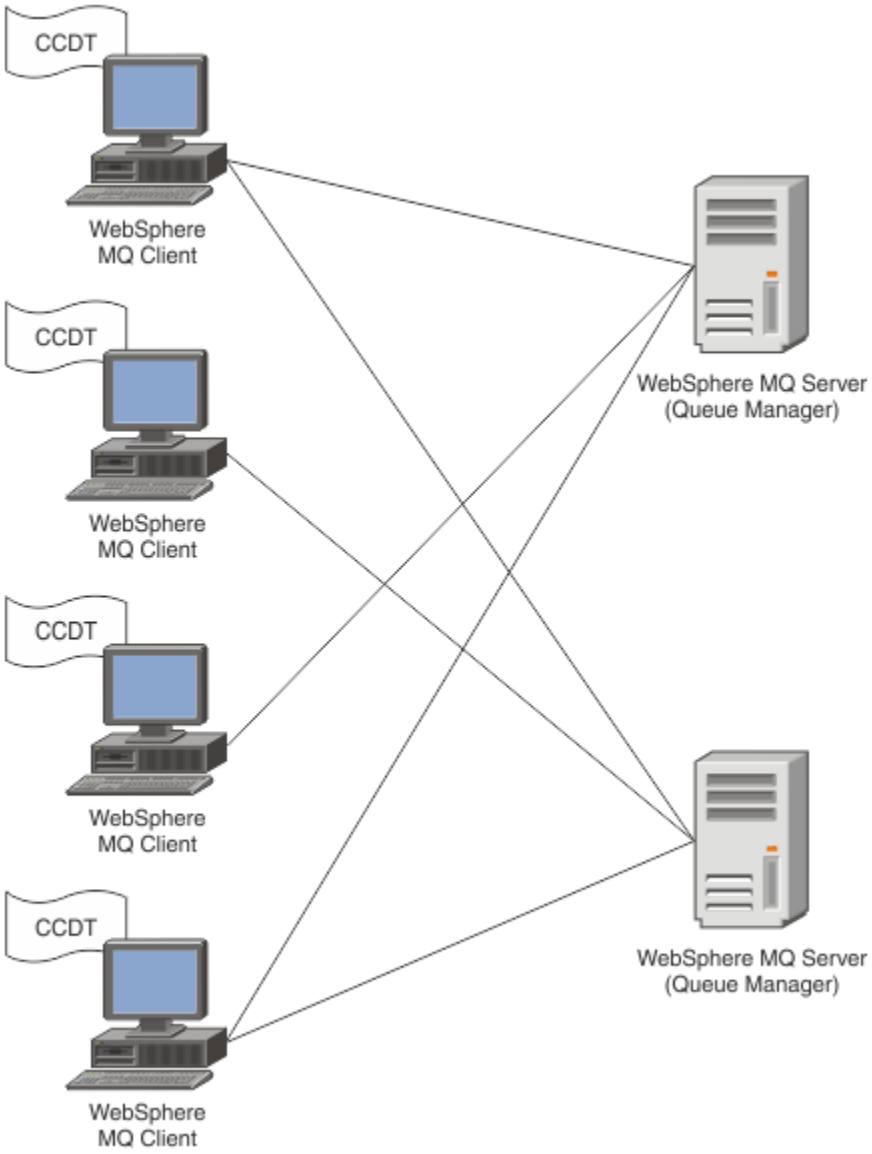
Wprowadzenie

Skonfiguruj globalne repozytorium, na przykład katalog LDAP (Lightweight Directory Access Protocol), w celu zapisania definicji połączenia klienta w celu konserwacji i administrowania pomocą.

Korzystanie z aplikacji klienta IBM WebSphere MQ w celu nawiązania połączenia z menedżerem kolejek za pośrednictwem tabeli definicji połączeń klienta (Client Connection Definition Table-CCDT).

Pakiet CCDT jest tworzony za pośrednictwem standardowego interfejsu administracyjnego MQSC produktu WebSphere MQ . Aby można było utworzyć definicje połączeń klientów, użytkownik musi być połączony z menedżerem kolejek, nawet jeśli dane zawarte w definicji nie są ograniczone do menedżera

kolejek. Wygenerowany plik CCDT musi być rozłożony ręcznie między maszynami klienckim i aplikacjami.

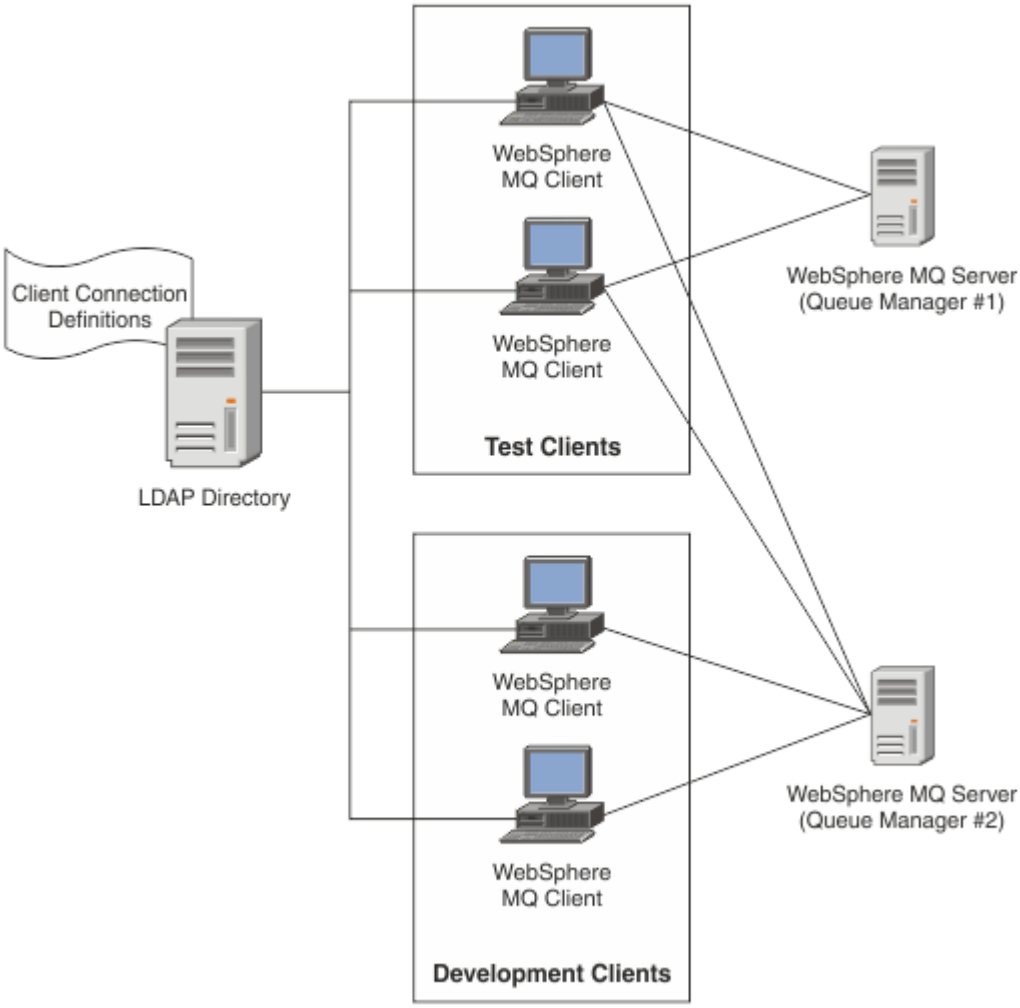


Plik CCDT musi być dystrybuowany do każdego klienta produktu WebSphere MQ . W przypadku, gdy tysiące klientów może istnieć lokalnie lub globalnie, wkrótce stanie się to trudne do utrzymania i administrowania. Potrzebne jest bardziej elastyczne podejście, aby zapewnić, że każdy klient będzie miał do dyspozycji poprawne definicje klientów.

Jednym z takich podejść jest przechowywanie definicji połączeń klienta w repozytorium globalnym, takim jak katalog LDAP (Lightweight Directory Access Protocol). Katalog LDAP może również udostępniać dodatkowe zabezpieczenia, indeksowanie i wyszukiwanie, umożliwiając każdemu klientowi dostęp tylko do tych definicji połączeń, które są z nimi związane.

Katalog LDAP można skonfigurować w taki sposób, aby dla określonych grup użytkowników były dostępne tylko konkretne definicje. Na przykład klienty testowe mogą uzyskiwać dostęp zarówno do menedżera

kolejek #1 , jak i do produktu #2, podczas gdy klienty programistyczne mają dostęp tylko do menedżera



kolejek #2 .

Moduł obsługi wyjścia może wyszukać repozytorium LDAP, na przykład IBM Tivoli Directory Server, w celu pobrania definicji kanałów. Korzystając z tych definicji połączeń, aplikacja kliencka WebSphere MQ może nawiązać połączenie z menedżerem kolejek.

Moduł wyjścia jest modułem wyjścia typu pre-connect, który umożliwia uzyskanie definicji kanału podczas wywołania MQCONN/MQCONNX z repozytorium LDAP.

Moduł wyjścia i schemat mogą być implementowane przez:

- Klienci, którzy już zbudowali bazę umiejętności, korzystając z istniejącej technologii opartej na pliku CCDT i chcą zmniejszyć koszty administrowania i dystrybucji.
- Dotychczasowi klienci, którzy już zatrudniają własną technologię dystrybuowania definicji połączeń klientów.
- Nowi lub obecni klienci, którzy obecnie nie korzystają z żadnego typu rozwiązania do połączenia z klientem i chcą korzystać z opcji oferowanych przez produkt IBM WebSphere MQ.
- Nowi lub obecni klienci, którzy chcą bezpośrednio wykorzystać lub dostroić swój model przesyłania komunikatów w sposób wstawiany do dowolnej bieżącej architektury biznesowej LDAP.

Obsługiwane środowiska

Przed uruchomieniem przykładu wyszukiwania punktu końcowego połączenia sprawdź, czy istnieje obsługiwany system operacyjny i odpowiednie oprogramowanie.

Przykładowy program do wyszukiwania punktów końcowych połączenia IBM WebSphere MQ wymaga następującego oprogramowania:

- IBM WebSphere MQ V7.0lub nowszy
- Klient Tivoli Directory Server V6.3 Client lub nowszy

Obsługiwane systemy operacyjne:

1. Windows (XP/2003/2008)
2. Solaris (SPARC i x86-64)
3. AIX
4. Linux
   - RHEL v4 i v5 w systemie System p
   - SUSE v9 i v10 w systemie System p
   - RHEL v4 i v5 System x32 bit i bit x64
   - SUSE v9 i v10 System x32 bit i bit x64
5. HP IA64.

Uwaga: Przykładowy program nie jest dostępny dla platform z/OS, i/5i HP PARISC.

Instalowanie i konfigurowanie

Instalowanie i konfigurowanie modułu obsługi wyjścia i schematu punktu końcowego połączenia.

Instalowanie modułu wyjścia

Podczas instalacji produktu WebSphere MQmoduł obsługi wyjścia jest instalowany w produkcie `tools/samples/c/preconnexit/bin`. W przypadku platform 32-bitowych moduł wyjścia musi być skopiowany do produktu `exit/<install name>/` , zanim będzie można go użyć. W przypadku platform 64-bitowych moduł obsługi wyjścia musi zostać skopiowany do katalogu exit64/<installation name>/ zanim będzie można go użyć.

Instalowanie schematu punktu końcowego połączenia

Wyjście korzysta ze schematu punktu końcowego połączenia *ibm-amq.schema* . Plik schematu musi zostać zaimportowany do dowolnego serwera LDAP, aby można było użyć wyjścia. Po zaimportowaniu schematu należy dodać wartości dla atrybutów.

Poniżej przedstawiono przykład importowania schematu punktu końcowego połączenia. W przykładzie założono, że używany jest serwer IBM Tivoli Directory Server (ITDS).

- Upewnij się, że serwer IBM Tivoli Directory Server jest uruchomiony, a następnie skopiuj plik *ibm-amq.schema* lub serwer FTP do serwera ITDS.
- Na serwerze ITDS wprowadź następującą komendę, aby zainstalować schemat w składnicy ITDS, gdzie ID LDAP i hasło LDAP są główną nazwą wyróżniającą i hasłem dla serwera LDAP:

  ldapadd -D "ID LDAP" -w "Hasło LDAP" -f ibm-amq.schema
- W oknie komend wprowadź następującą komendę lub użyj narzędzia innej firmy, aby przejrzeć schemat w celu weryfikacji:

  ldapsearch objectclass=ibm-amqClientConnection

Szczegółowe informacje na temat importowania pliku schematu można znaleźć w dokumentacji serwera LDAP.

Konfiguracja

Nowa sekcja o nazwie **PreConnect** musi zostać dodana do pliku konfiguracyjnego klienta, do pliku *mqclient.ini* . Sekcja PreConnect zawiera następujące słowa kluczowe:

Moduł : nazwa modułu zawierającego kod wyjścia API. Jeśli w tym polu znajduje się pełna ścieżka do modułu, jest on używany jako folder *exit* lub *exit64* w instalacji produktu WebSphere MQ .

Funkcja : Nazwa punktu wejścia funkcjonalnego w bibliotece, która zawiera kod wyjścia PreConnect . Definicja funkcji jest zgodna z prototypem MQ_PRECONNECT_EXIT.

Dane : identyfikator URI repozytorium LDAP zawierającego definicje kanałów.

Poniższy fragment kodu stanowi przykład zmian wymaganych w pliku *mqclient.ini* .

```
PreConnect:
  Module=amqlcelp
  Function=PreConnectExit
  Data=ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
  Sequence=1
```

Przegląd procedury obsługi wyjścia i schematu

Składnia i parametry używane do nawiązania połączenia z menedżerem kolejek.

Produkt WebSphere MQ v7.5 definiuje następującą składnię dla punktu wejścia w module wyjścia.

```
void MQENTRY MQ_PRECONNECT_EXIT ( PMQNXP  pExitParms
                                , PMQCHAR pQMgrName
                                , PPMQCNO ppConnectOpts
                                , PMQLONG pCompCode
                                , PMQLONG pReason)
```

Podczas wykonywania wywołania MQCONN/X program WebSphere MQ C Client ładuje moduł obsługi wyjścia zawierający implementację składni funkcji. Następnie wywołuje funkcję wyjścia w celu pobrania definicji kanałów. Pobrane definicje kanałów są następnie używane do nawiązania połączenia z menedżerem kolejek.

Parametry

pExitParms

Typ: PMQNXP input/output

Struktura parametru wyjścia PreConnection (PreConnection). Struktura jest przydzielana i obsługiwana przez program wywołujący wyjście.

```
struct tagMQNXP
{
  MQCHAR4    StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQLONG    ExitId;             /* Type of exit */
  MQLONG    ExitReason;         /* Reason for invoking exit */
  MQLONG    ExitResponse;       /* Response from exit */
  MQLONG    ExitResponse2;      /* Secondary response from exit */
  MQLONG    Feedback;           /* Feedback code (reserved) */
  MQLONG    ExitDataLength;        /* Exit data length */
  PMQCHAR      pExitDataPtr;           /* Exit data */
  MQPTR        pExitUserAreaPtr;       /* Exit user area */
  PMQCD *    ppMQCDArrayPtr;        /* Array of pointers to MQCDs */
  MQLONG    MQCDArrayCount;        /* Number of entries found */
  MQLONG    MaxMQCDVersion;        /* Maximum MQCD version */
};
```

NazwapQMgr

Typ: input/output PMQCHAR

Nazwa menedżera kolejek. Na wejściu parametr ten jest łańcuchem filtru dostarczonym do wywołania interfejsu API MQCONN za pomocą parametru **QMgrName** . To pole może być puste, jawne lub zawierać określone znaki wieloznaczne. Pole zostanie zmienione przez wyjście. Parametr ma wartość NULL, gdy wyjście jest wywoływane z MQXR_TERM.

ppConnectOpts (Opts)

Wpisz: ppConnectOpts input/output

Opcje, które sterują działaniem MQCONNX. Jest to wskaźnik do struktury opcji połączenia MQCNO, która steruje działaniem wywołania interfejsu API MQCONN. Parametr ma wartość NULL, gdy wyjście jest wywoływane z MQXR_TERM. Klient MQI zawsze udostępnia strukturę MQCNO do wyjścia, nawet jeśli nie została ona pierwotnie udostępniona przez aplikację. Jeśli aplikacja udostępnia strukturę

MQCNO, klient tworzy duplikat w celu przekazania go do wyjścia, w którym jest on modyfikowany. Klient zachowuje prawo własności do obiektu MQCNO. Tabela MQCD, do której odwołuje się parametr MQCNO, ma pierwszeństwo przed każdą definicją połączenia udostępnionej przez tablicę. Klient używa struktury MQCNO do łączenia się z menedżerem kolejek, a pozostałe są ignorowane.

KodpComp

Typ: PMQLONG input/output

Kod zakończenia. Wskaźnik do obiektu MQLONG, który odbiera kod zakończenia wyjścia. Musi to być jedna z następujących wartości:

MQCC_OK-pomyślne zakończenie

MQCC_WARNING-ostrzeżenie (częściowe zakończenie)

MQCC_FAILED-wywołanie nie powiodło się

pReason

Typ: PMQLONG input/output

Przyczyna kwalifikowania kodu pComp. Wskaźnik do obiektu MQLONG, który odbiera kod przyczyny wyjścia. Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE-(0, x '000') Nie ma powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Informacje o kontekście LDAP MQ

Wyjście korzysta z następującej struktury danych dla informacji kontekstowych.

MQNLDAPCTX

Struktura MQNLDAPCTX ma następujący prototyp C.

```
typedef struct tagMQNLDAPCTX MQNLDAPCTX;
typedef MQNLDAPCTX MQPOINTER PMQNLDAPCTX;

struct tagMQNLDAPCTX
{
  MQCHAR4    StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  LDAP *    objectDirectory;    /* LDAP Instance */
  MQLONG       ldapVersion;        /* Which LDAP version to use? */
  MQLONG       port;            /* Port number for LDAP server*/
  MQLONG       sizeLimit;        /* Size limit */
  MQBOOL    ssl;            /* SSL enabled? */
  MQCHAR *    host;            /* Hostname of LDAP server */
  MQCHAR *    password;        /* Password of LDAP server */
  MQCHAR *    searchFilter;        /* LDAP search filter */
  MQCHAR *    baseDN;            /* Base Distinguished Name */
  MQCHAR *    charSet;        /* Character set */
};
```

Przykładowy kod do budowania wyjścia wyszukiwania punktu końcowego połączenia

Fragmenty kodu służące do kompilowania źródła w systemie Windows i na platformach rozproszonych.

Kompilowanie źródła

Źródło można skompilować z dowolnymi bibliotekami klienta LDAP, na przykład bibliotekami klienta IBM Tivoli Directory Server 6.3 . W tej dokumentacji przyjęto założenie, że używane są biblioteki klienta Tivoli Directory Server 6.3 .

Uwaga: Biblioteka obsługi wyjścia wstępnego połączenia została przetestowana z następującymi serwerami LDAP:

- IBM Tivoli Directory Server V6.3
- Novell eDirectory V8.2

Poniższe fragmenty kodu opisują sposób kompilowania wyjść w systemie Windowsoraz inne platformy rozproszone:

Kompilowanie wyjścia na platformie Windows

Do kompilowania źródła wyjścia w systemie Windowsmożna użyć następującego fragmentu kodu:

```
CC=cl.exe
LL=link.exe
CCARGS=/c /I. /DWIN32 /W3 /DNDEBUG /EHsc /D_CRT_SECURE_NO_DEPRECATE /Zl

# The libraries to include
LDLIBS=Ws2_32.lib Advapi32.lib libibmldapstatic.lib libibmldapdbgstatic.lib  \
kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib \
shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib msvcrt.lib

OBJS=amqlcel0.obj

all: amqlcelp.dll

amqlcelp.dll: $(OBJS)
        $(LL) /OUT:amqlcelp.dll /INCREMENTAL /NOLOGO /DLL /SUBSYSTEM:WINDOWS /MACHINE: X86 /
DEF:amqlcelp.def $(OBJS) $(LDLIBS) /NODEFAULTLIB:msvcrt.lib

# The exit source
amqlcel0.obj: amqlcel0.c
              $(CC) $(CCARGS) $*.c
```

Uwaga: Podczas kompilowania bibliotek klienta IBM Tivoli Directory Server 6.3 z programem Microsoft Visual Studio 2005 lub powyżej kompilatora mogą być wyświetlane ostrzeżenia, jeśli używane są biblioteki klienta IBM Tivoli Directory Server 6.3 skompilowane z kompilatorem Microsoft Visual Studio 2003.

Kompilowanie wyjścia na innych platformach rozproszonych

Za pomocą poniższego fragmentu kodu można kompilować źródło wyjścia na innych platformach rozproszonych, na przykład Linux. Niektóre opcje kompilatora mogą się różnić od innych platform rozproszonych.

```
#Make file to build exit
CC=gcc

MQML=/opt/mqm/lib
MQMI=/opt/mqm/inc
TDSI=/opt/ibm/ldap/V6.3/include
XFLAG=-m32

TDSL=/opt/ibm/ldap/V6.3/lib
```

Produkt IBM Tivoli Directory Server jest dostarczany zarówno ze statycznymi, jak i dynamicznymi bibliotekami połączeń, ale można użyć tylko jednej z nich. W tym skrypcie przyjęto założenie, że używane są biblioteki statyczne.

```
#Use static libraries.
LDLIBS=-L$(TDSL) -libibmldapstatic

CFLAGS=-I. -I$(MQMI) -I$(TDSI)

all:amqlcepl

amqlcepl: amqlcel0.c
        $(CC) -o cepl amqlcel0.c  -shared -fPIC $(XFLAG) $(CFLAGS) $(LDLIBS)
```

Wyjdź z wywołania modułu

Moduł wyjścia PreConnect może być wywoływany z trzema różnymi kodami przyczyny. W tej sekcji opisano poszczególne przyczyny wyjścia w większej głębokości.

MQXR_INIT

Wyjście jest wywoływane z kodem przyczyny MQXR_INIT na potrzeby inicjowania i nawiązywania połączenia z serwerem LDAP.

Przed wywołaniem metody *MQXR_INIT* pole *pExitDataPtr* struktury MQNXP zostałoby zapełnione atrybutem Dane z sekcji PreConnect w pliku *mqclient.ini* (tzn. katalogu LDAP).

Adres URL LDAP składa się z co najmniej protokołu, nazwy hosta, numeru portu i podstawowej nazwy wyróżniającej dla wyszukiwania. Wyjście analizuje adres URL LDAP zawarty w polu *pExitDataPtr* ,

przydziela strukturę kontekstu wyszukiwania LDAP MQNLDAPCTX i odpowiednio zapełnia je. Adres tej struktury jest zapisany w polu *pExitUserAreaPtr* . Niepowodzenie poprawnego przeanalizowania wyników adresu URL LDAP w przypadku błędu MQCC_FAILED.

W tym momencie program obsługi wyjścia łączy się i łączy z serwerem LDAP za pomocą parametrów MQNLDAPCTX. Wynikowe uchwyty interfejsu API LDAP są również przechowywane w tej strukturze.

MQXR_PRECONNECT
Moduł obsługi wyjścia jest wywoływany z kodem przyczyny MQXR_PRECONNECT w celu pobrania definicji kanałów z serwera LDAP.

Program zewnętrzny przeszukuje serwer LDAP w celu uzyskania definicji kanałów zgodnych z podanym filtrem. Jeśli parametr *QMgrName* zawiera konkretną nazwę menedżera kolejek, to wyszukiwanie zwróci wszystkie definicje kanałów, których wartość atrybutu LDAP *ibm-amqQueueManagerName* jest zgodna z podaną nazwą menedżera kolejek.

Jeśli parametr *QMgrName* ma wartość '*' lub ' ' (puste), a następnie wyszukiwanie zwraca wszystkie definicje kanałów, których atrybut punktu końcowego *ibm-amqIsClientDefault* Connection jest ustawiony na wartość true.

Po pomyślnym wyszukaniu program obsługi wyjścia przygotowuje jedną lub tablicę definicji MQCD i powraca do programu wywołującego.

MQXR_TERM
Wyjście jest wywoływane z tym kodem przyczyny, gdy program obsługi wyjścia ma zostać wyczyszczony. W trakcie tego wyjścia rozłącza się z serwerem LDAP, zwalnia całą pamięć przydzieloną i utrzymywana przez wyjście. Będzie to obejmować strukturę *MQNLDAPCTX* , tablicę wskaźników i wszystkie odwołania MQCD, które są przywołane. Wszystkie pozostałe pola są ustawiane na wartości domyślne. Parametry wyjściowe *pQMgrName* i *ppConnectOpts* są nieużywane podczas operacji *MQXR_TERM* i mogą mieć wartość NULL.

Schematy LDAP
Dane połączenia klienta są przechowywane w repozytorium globalnym zwanym katalogiem LDAP (Lightweight Directory Access Protocol). Klient WebSphere MQ korzysta z katalogu LDAP w celu uzyskania definicji połączeń. Struktura definicji połączeń klienta WebSphere MQ w katalogu LDAP jest znana jako schemat LDAP. Schemat LDAP to kolekcja definicji typów atrybutów, definicji klas obiektów i innych informacji używanych przez serwer w celu określenia, czy asercja filtru lub wartości atrybutu jest zgodna z atrybutami pozycji, a także czy ma być dozwolone, dodawać i modyfikować operacje.

Zapisywanie danych w katalogu LDAP

Definicje połączeń klienta znajdują się w konkretnej gałęzi w drzewie katalogów, o którym wiadomo, że jest to punkt połączenia. Podobnie jak w przypadku wszystkich innych węzłów w katalogu LDAP, punkt połączenia ma powiązaną nazwę wyróżniającą (DN). Węzła tego można używać jako punktu początkowego dla zapytań, które są wykonywane w katalogu. Użyj filtrowania podczas wysyłania zapytania do katalogu LDAP w celu zwrócenia podzbioru definicji połączeń klienta. Dostęp do poddrzew można ograniczyć w oparciu o uprawnienia nadane w innych częściach drzewa katalogów-na przykład dla użytkowników, działów lub grup.

Definiowanie własnych atrybutów i klas
Zapisz definicję kanału klienta, modyfikując schemat LDAP. Wszystkie definicje danych LDAP wymagają obiektów i atrybutów. Obiekty i atrybuty są identyfikowane przez numer identyfikatora obiektu (OID), który jednoznacznie identyfikuje obiekt lub atrybut. Wszystkie klasy w schemacie LDAP dziedziczą bezpośrednio lub pośrednio z najwyższego obiektu. Obiekt definicji kanału klienta zawiera atrybuty obiektu najwyższego poziomu. Wszystkie definicje danych LDAP wymagają obiektów i atrybutów:

- Definicje obiektów to kolekcje atrybutów LDAP.
- Atrybuty to typy danych LDAP.

Opis każdego atrybutu i sposób ich odwzorowania na normalne właściwości produktu WebSphere MQ są opisane w sekcji Atrybuty LDAP .

LDAP - atrybuty

Zdefiniowane atrybuty LDAP są specyficzne dla produktu WebSphere MQ i są odwzorowywać bezpośrednio na właściwości połączenia klienta.

Atrybuty Łańcucha Katalogu Kanału Klienta WebSphere MQ

Atrybuty łańcucha znaków z ich odwzorowaniem na właściwości produktu WebSphere MQ są wymienione w poniższej tabeli. Atrybuty mogą zawierać wartości directoryString (UTF-8 zakodowany Unicode, czyli system kodowania bajtów, który zawiera kod IA5/ASCII jako podzbiór). Składnia jest określana na podstawie numeru identyfikacyjnego obiektu (OID).

Tabela 21. Atrybuty łańcuchowe katalogu kanału klienta WebSphere MQ

Atrybut LDAP	Opis	Właściwość produktu WebSphere MQ
CN	Nazwa zwykła składająca się z nazwy kanału i nazwy definiującego menedżera kolejek.	
Nazwa IBM-amqChannel	Nazwa definicji kanału.	CHANNEL
Nazwa IBM-amqConnection	Identyfikator połączenia komunikacyjnego.	CONNAME
ibm-amqDescription	Opis kanału.	DESCR
Adres IBM-amqLocal	Lokalny adres komunikacyjny kanału.	LOCLADDR
ibm-amqModeNazwa	s a bThe LU 6.2 mode name.	MODENAME
ibm-amqPassword	Hasło, które może być używane.	PASSWORD
ibm-amqQueueManagerName	Nazwa menedżera kolejek lub grupy menedżerów kolejek, do której może zażądać połączenia przez aplikację kliencką WebSphere MQ .	QMNAME
ibm-amqSecurityExitUserData	Dane użytkownika, które są przekazywane do wyjścia zabezpieczeń.	SCYDATA
ibm-amqSecurityExitName	Nazwa programu obsługi wyjścia, który ma być uruchomiony przez wyjście zabezpieczeń kanału.	SCYEXIT
ibm-amqSslCipherSpec	Pojedyncza specyfikacja CipherSpec dla połączenia SSL.	SSLCIPH
ibm-amqSslPeerName	Sprawdza nazwę wyróżniającą (DN) certyfikatu pochodzącego od menedżera kolejek węzła sieci lub klienta na drugim końcu kanału WebSphere MQ .	SSLPEER
ibm-amqTransactionProgramName	Nazwa programu transakcyjnego.	TPNAME
Identyfikator ibm-amqUser	Identyfikator użytkownika, który ma być używany przez agenta MCA podczas próby zainicjowania bezpiecznej sesji SNA z użyciem zdalnego agenta MCA.	USERID

Atrybuty liczby całkowitej połączenia klienta WebSphere MQ

Atrybuty z predefiniowanymi wartościami (na przykład typ wyliczeniowy) są zapisywane jako standardowe liczby całkowite. Wartości te są przechowywane w katalogu LDAP jako wartości całkowite, a nie przez użycie powiązanej nazwy stałej.

Tabela 22. Atrybuty liczby całkowitej katalogu kanału klienta WebSphere MQ		
Atrybut LDAP	**Opis**	**Właściwość produktu WebSphere MQ**
ibm-amqConnectionpowinowactwo	Określa, czy aplikacje klienckie, które łączą się wiele razy z tą samą nazwą menedżera kolejek, korzystają z tego samego kanału klienta.	AFFINITY
ibm-amqClientChannelWeight	Waga wpływająca na użycie definicji kanału połączenia klienta.	CLNTWGHT
ibm-amqHeartBeatInterval	Przybliżony czas między przepływami pulsu przekazywanymi od wysyłającego agenta MCA, kiedy nie ma żadnych komunikatów w kolejce wysyłania.	HBINT
ibm-amqKeepAliveInterval	Wartość limitu czasu dla kanału.	KAINT
ibm-amqMaximumMessageLength	Maksymalna długość komunikatu, który może zostać przesłany w kanale.	MAXMSGL
ibm-amqSharing-konwersacje	Maksymalna liczba konwersacji, które współużytkują każdą instancję kanału TCP/IP.	SHARECNV
IBM-amqTransportTyp	Typ transportu, który ma być używany.	TRPTYPE

Atrybut boolowski kanału klienta WebSphere MQ

Ten atrybut boolowski nie jest odwzorowany na żadną właściwość produktu WebSphere MQ . Składnia tego atrybutu wskazuje wartość boolowką.

Tabela 23. Atrybut boolowski kanału klienta WebSphere MQ	
Atrybut LDAP	**Opis**
ibm-amqIsClientDefault	Ten atrybut boolowski jest zdefiniowany w celu rozwiązania problemu wyszukiwania pozycji, których atrybut ibm-amqQueueManagerName nie został zdefiniowany.

Atrybuty listy kanałów klienta WebSphere MQ

Właściwości produktu WebSphere MQ są przechowywane jako jednowartościowe, rozdzielane przecinkami atrybuty listy w katalogu LDAP. Atrybuty są definiowane w taki sam sposób, jak inne atrybuty łańcuchowe katalogu. Atrybuty listy wraz z ich odwzorowaniem na właściwości produktu WebSphere MQ są opisane w poniższej tabeli.

Tabela 24. Atrybuty listy kanałów klienta WebSphere MQ		
Atrybut LDAP	**Opis**	**Właściwość produktu WebSphere MQ**
ibm-amqHeaderKompresja	Lista technik kompresji danych nagłówka obsługiwanych przez kanał.	COMPHDR
ibm-amqMessage-kompresja	Lista technik kompresji danych komunikatu obsługiwanych przez kanał.	COMPMSG
ibm-amqSendExitUserData	Dane użytkownika, które są przekazywane do wyjścia wysyłania.	SENDDATA
Nazwa programu ibm-amqSendExitUser	Nazwa programu obsługi wyjścia, który ma być uruchomiony przez wyjście wysyłania kanału.	SENDEXIT

Tabela 24. Atrybuty listy kanałów klienta WebSphere MQ (kontynuacja)		
Atrybut LDAP	**Opis**	**Właściwość produktu WebSphere MQ**
ibm-amqReceiveExitUserData	Dane użytkownika, które są przekazywane do wyjścia odbierania.	RCVDATA
ibm-amqReceiveExitName	Nazwa programu użytkownika obsługi wyjścia, który ma być uruchomiony przez kanał wyjściowy odbierania użytkownika.	RCVEXIT

Nazwa zwykła
Nazwa zwykła (CN) składa się z nazwy kanału i nazwy definiującego menedżera kolejek.

Jest to istniejący atrybut.

Format CN jest następujący:

```
CN=CHANNEL_NAME(DEFINING_Q_MGR_NAME)
```

Na przykład:

```
CN=TC1(QM_T1)
```

Dla tego atrybutu można podać tylko jedną wartość.

Ten atrybut jest atrybutem łańcuchowym, a w wartościach nie jest rozróżniana wielkość liter. Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania, przy użyciu podłańcucha (na przykład CN=jim *, gdzie CN jest atrybutem) i zawiera jedną lub więcej znaków wieloznacznych.

Nazwa IBM-amqChannel
Ten atrybut określa nazwę definicji kanału.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 20 znaków, w przypadku której nie jest rozróżniana wielkość liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania, przy użyciu podłańcucha i zawiera jedną lub więcej znaków wieloznacznych.

ibm-amqDescription
Ten atrybut LDAP zawiera opis kanału.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 64 bajtów, bez rozróżniania wielkości liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

Nazwa IBM-amqConnection
Ten atrybut LDAP jest identyfikatorem połączenia komunikacyjnego. Określa on konkretne łącza komunikacyjne, które mają być używane przez ten kanał.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 264 znaków, bez rozróżniania wielkości liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

Adres ibm-amqLocal
Ten atrybut określa adres komunikacji lokalnej dla kanału.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 48 znaków, w przypadku której nie jest rozróżniana wielkość liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

ibm-amqModeName
Ten atrybut jest używany dla połączeń LU 6.2. Dodatkowa definicja parametrów sesji dla połączenia, gdy wykonywana jest alokacja sesji komunikacyjnej.

Ten atrybut ma jedną wartość łańcuchową o długości dokładnie 8 znaków, bez rozróżniania wielkości liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

ibm-amqPassword
Ten atrybut LDAP określa hasło, które może być używane przez agenta MCA podczas próby zainicjowania bezpiecznej sesji LU 6.2 ze zdalnym agentem MCA.

Ten atrybut ma pojedynczą wartość całkowitą z maksymalną liczbą 12 cyfr. Nie jest to istniejący atrybut.

ibm-amqQueueManagerName
Ten atrybut określa nazwę menedżera kolejek lub grupy menedżerów kolejek, do których aplikacja kliencka WebSphere MQ może zażądać połączenia.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 48 znaków, w przypadku której nie jest rozróżniana wielkość liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

ibm-amqSecurityExitUserData
Ten atrybut LDAP określa dane użytkownika, które są przekazywane do wyjścia zabezpieczeń.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 999 znaków, w przypadku której nie jest rozróżniana wielkość liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

ibm-amqSecurityExitName
Ten atrybut LDAP określa nazwę programu obsługi wyjścia, który ma być uruchamiany przez wyjście zabezpieczeń kanału.

Jeśli wyjście zabezpieczeń kanału nie jest aktywne, pozostaw to pole puste.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 999 znaków, w przypadku której nie jest rozróżniana wielkość liter. Ten atrybut nie jest wstępem do wyjścia.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

ibm-amqSslCipherSpec
Ten atrybut LDAP określa pojedynczą wartość atrybutu CipherSpec dla połączenia SSL.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 32 znaków, która nie rozróżnia wielkości liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

ibm-amqSslPeerName
Ten atrybut LDAP jest używany do sprawdzania nazwy wyróżniającej (DN) certyfikatu z menedżera kolejek węzła sieci lub klienta na drugim końcu kanału WebSphere MQ .

Ten atrybut LDAP ma pojedynczą wartość łańcuchową o maksymalnej wielkości 1024 bajtów, w przypadku której nie jest rozróżniana wielkość liter. Nie jest to już wcześniej.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

ibm-amqTransactionProgramName
Ten atrybut LDAP określa nazwę programu transakcyjnego. Jest on przeznaczony do użytku z połączeniami LU 6.2 .

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 64 znaków, która nie rozróżnia wielkości liter. Nie jest to już wcześniej.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

Identyfikator ibm-amqUser
Ten atrybut LDAP określa ID użytkownika, który ma być używany przez agenta MCA podczas próby zainicjowania bezpiecznej sesji SNA ze zdalnym agentem MCA.

Ten atrybut ma pojedynczą wartość łańcuchową o długości dokładnie 12 znaków, bez rozróżniania wielkości liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

ibm-amqConnectionAffinity
Ten atrybut LDAP określa, czy aplikacje klienckie, które łączą wiele razy przy użyciu tej samej nazwy menedżera kolejek, korzystają z tego samego kanału klienta.

Ten atrybut ma pojedynczą wartość całkowitą. Nie jest to istniejący atrybut.

ibm-amqClientChannelWeight
Ten atrybut LDAP określa wagę, która ma wpływ na używaną definicję kanału połączenia klienta.

Atrybut ważenia kanału klienta jest używany do bias wyboru definicji kanału klienta, jeśli dostępna jest więcej niż jedna odpowiednia definicja.

Ten atrybut ma pojedynczą wartość całkowitą. Nie jest to istniejący atrybut.

ibm-amqHeartBeatInterval
Ten atrybut LDAP określa przybliżony czas między przepływami pulsu, które mają być przekazywane z wysyłającego agenta MCA, gdy w kolejce transmisji nie ma żadnych komunikatów.

Ten atrybut ma pojedynczą wartość całkowitą. Nie jest to istniejący atrybut. Wartością domyślną jest 1. Wartość domyślna jest ustawiana w bieżącej operacji zmiennej środowiskowej MQSERVER.

ibm-amqKeepAliveInterval
Ten atrybut LDAP jest używany do określania wartości limitu czasu dla kanału.

Wartość tego atrybutu jest przekazywana do stosu komunikacyjnego określającego czas sprawdzania połączenia dla kanału. Można użyć tej opcji, aby określić inną wartość sprawdzania połączenia dla każdego kanału.

Ten atrybut ma pojedynczą wartość całkowitą. Nie jest to istniejący atrybut.

ibm-amqMaximumMessageLength
Ten atrybut LDAP określa maksymalną długość komunikatu, który może być przesłany w kanale.

Wartością domyślną tego atrybutu jest 104857600 zgodnie z bieżącą operacją zmiennej środowiskowej MQSERVER. Ten atrybut ma pojedynczą wartość całkowitą, a nie jest to atrybut wcześniej istniejący.

Konwersacje z programem ibm-amqSharing
Ten atrybut LDAP określa maksymalną liczbę konwersacji, które współużytkują każdą instancję kanału TCP/IP.

Ten atrybut ma pojedynczą wartość całkowitą. Ten atrybut nie jest wstępnie istniejącym atrybutem.

Typ ibm-amqTransport
Ten atrybut LDAP określa typ transportu, który ma być używany.

Ten atrybut ma pojedynczą wartość całkowitą. Nie jest to istniejący atrybut.

ibm-amqIsClientDefault
Ten atrybut boolowski rozwiązuje problem wyszukiwania pozycji, w których atrybut ibm-amqQueueManagerName nie został zdefiniowany.

Moduły wyjścia preconnect zwykle przeszukiwane są serwery LDAP z wartością atrybutu ibm-amqQueueManagerName jako kryterium wyszukiwania. Takie zapytanie zwróci wszystkie pozycje, w których wartość atrybutu ibm-amqQueueManagerName jest zgodna z nazwą menedżera kolejek określonego w wywołaniu MQCONN/X. Jednak w przypadku korzystania z tabel definicji kanału klienta (CCDT) można ustawić nazwę menedżera kolejek w wywołaniu MQCONN/X jako puste lub poprzedzić nazwę znakiem gwiazdki (*). Jeśli nazwa menedżera kolejek jest pusta, klient łączy się z domyślnym menedżerem kolejek. Jeśli nazwa jest poprzedzona znakiem gwiazdki (*) w menedżerze kolejek, klient łączy się z dowolnym menedżerem kolejek.

Analogicznie, atrybut ibm-amqQueueManagerName w pozycji może pozostać niezdefiniowany. W tym przypadku oczekuje się, że klient korzystający z tego punktu końcowego może połączyć się z dowolnym menedżerem kolejek. Na przykład pozycja zawiera następujące wiersze:

```
ibm-amqChannelName = "CHANNEL1"
ibm-amqConnectionName = myhost(1414)
```

W tym przykładzie klient próbuje połączyć się z określonym menedżerem kolejek działającym w systemie `myhost`.

Jednak w serwerach LDAP wyszukiwanie nie jest wykonywane dla wartości atrybutu, która nie została zdefiniowana. Jeśli na przykład pozycja zawiera informacje o połączeniu z wyjątkiem ibm-amqQueueManagerName, to wyniki wyszukiwania nie będą zawierały tej pozycji. Aby rozwiązać ten problem, można ustawić opcję ibm-amqIsClientDefault. Jest to atrybut boolowski i przyjmuje się, że jeśli nie jest zdefiniowany, ma wartość FALSE.

For entries where the ibm-amqQueueManagerName has not been defined and are expected to be part of the search, set ibm-amqIsClientDefault to TRUE. Jeśli jako nazwę menedżera kolejek w wywołaniu MQCONN/X określono nazwę menedżera kolejek lub gwiazdkę (*), program zewnętrzny preconnect przeszukuje serwer LDAP dla wszystkich pozycji, w których wartość atrybutu ibm-amqIsClientDefault jest ustawiona na TRUE.

Uwaga: Nie ustawiaj lub nie definiuj atrybutu ibm-amqQueueManagerName , jeśli wartość parametru ibm-amqIsClientDefault jest ustawiona na TRUE.

Kompresja ibm-amqHeader
Ten atrybut LDAP jest listą technik kompresji danych nagłówka obsługiwanych przez kanał.

Maksymalna wielkość tego atrybutu wynosi 48 znaków. Nie jest to istniejący atrybut.

Dla tego atrybutu można podać tylko jedną wartość.

Ten atrybut listy jest określany jako łańcuchy katalogów przy użyciu formatu rozdzielonego przecinkami. Na przykład wartości określone dla parametru **ibm-amqHeaderCompression** to `0` , które są odwzorowane na wartość `NONE`. Wszystkie wartości, które przekraczają maksymalny dozwolony limit, zostaną zignorowane przez klienta. Na przykład: ibm-amqHeader-kompresja zawiera maksymalnie 2 liczby całkowite na liście.

Kompresja ibm-amqMessage
Ten atrybut LDAP jest listą technik kompresji danych komunikatu obsługiwanych przez kanał.

Maksymalna wielkość tego atrybutu wynosi 48 znaków. Nie jest to istniejący atrybut.

Ten atrybut nie obsługuje wielu wartości.

Ten atrybut listy jest określany jako łańcuchy katalogów przy użyciu formatu rozdzielonego przecinkami. Na przykład wartością określoną dla tego atrybutu jest 1,2,4, która jest odwzorowana na bazową sekwencję kompresji RLE, ZLIBFAST i ZLIBHIGH.

Wszystkie wartości przekraczające maksymalny dozwolony limit są ignorowane przez klienta. Na przykład: ibm-amqMessage-kompresja zawiera maksymalnie 16 liczb całkowitych na liście.

Dane produktu IBM-amqSendExitUser
Ten atrybut LDAP określa dane użytkownika, które są przekazywane do wyjścia wysyłania.

Ten atrybut LDAP ma pojedynczą wartość łańcuchową o maksymalnej długości 999 znaków, w przypadku której nie jest rozróżniana wielkość liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

Uwaga: Produkty **ibm-amqSendExitName** i **ibm-amqSendExitUserData** muszą być synchronizowane w parach. Dane użytkownika powinny być synchronizowane z nazwą wyjścia. Tak więc, jeśli ktoś jest określony, drugi również musi być określony symetrycznie, nawet jeśli nie zawiera żadnych danych.

ibm-amqSendExitName
Ten atrybut LDAP określa nazwę programu obsługi wyjścia, który ma być uruchamiany przez wyjście wysyłania kanału.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 999 znaków, w przypadku której nie jest rozróżniana wielkość liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

Uwaga: Produkty **ibm-amqSendExitName** i **ibm-amqSendExitUserData** muszą być zsynchronizowane w parach. Dane użytkownika muszą być zsynchronizowane z nazwą wyjścia. Tak więc, jeśli jest określony, drugi również musi być określony symetrycznie, nawet jeśli nie zawiera żadnych danych.

Dane programu ibm-amqReceiveExitUser
Ten atrybut LDAP określa dane użytkownika, które są przekazywane do wyjścia odbierania.

Istnieje możliwość uruchomienia sekwencji wyjść odbierania. Łańcuch danych użytkownika dla serii wyjść jest oddzielony przecinkiem, spacjami lub dwoma znakami.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 999 znaków, w przypadku której nie jest rozróżniana wielkość liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

Uwaga: Produkty **ibm-amqReceiveExitName** i **ibm-amqReceiveExitUserData** muszą być zsynchronizowane w parach. Dane użytkownika muszą być zsynchronizowane z nazwą wyjścia. Tak więc, jeśli jest określony, drugi również musi być określony symetrycznie, nawet jeśli nie zawiera żadnych danych.

ibm-amqReceiveExitName
Ten atrybut LDAP określa nazwę programu użytkownika obsługi wyjścia, który ma być uruchamiany przez wyjście użytkownika odbierania przez kanał.

Ten atrybut jest listą nazw programów, które mają być uruchamiane w ramach sukcesji. Pozostaw puste pole, jeśli żaden kanał odbierający nie jest w stanie zakończyć działania.

Ten atrybut ma pojedynczą wartość łańcuchową o maksymalnej długości 999 znaków, w przypadku której nie jest rozróżniana wielkość liter. Nie jest to istniejący atrybut.

Dopasowywanie podłańcucha jest ignorowane. Dopasowywanie podłańcuchów jest regułą dopasowywania używaną w podschemacie, która określa zachowanie atrybutu w filtrze wyszukiwania.

Uwaga: Produkty **ibm-amqReceiveExitName** i **ibm-amqReceiveExitUserData** muszą być zsynchronizowane w parach. Dane użytkownika muszą być zsynchronizowane z nazwą wyjścia. Jeśli więc zostanie podany, drugi musi być również określony symetrycznie, nawet jeśli nie zawiera żadnych danych.

Pisanie aplikacji kolejkowania

Ta sekcja zawiera informacje na temat pisania aplikacji kolejkowania, łączenia się i rozłączania z menedżerem kolejek, publikowania/subskrypcji oraz obiektów otwierających i zamykających.

Aby dowiedzieć się więcej na temat pisania aplikacji, skorzystaj z poniższych odsyłaczy:

Pojęcia pokrewne

“Pojęcia związane z projektowaniem aplikacji” na stronie 8
Do pisania aplikacji IBM WebSphere MQ można użyć wyboru języków proceduralnych lub obiektowych. Odsyłacze w tym temacie można znaleźć w informacjach dotyczących pojęć związanych z produktem IBM WebSphere MQ , które są przydatne dla programistów aplikacji.

“Wybór języka programowania do użycia” na stronie 80
W tej sekcji znajdują się informacje na temat języków programowania i środowisk obsługiwanych przez produkt IBM WebSphere MQ, a także niektóre zagadnienia dotyczące ich używania.

“Projektowanie aplikacji produktu IBM WebSphere MQ” na stronie 91
Po zdecydowaniu się, w jaki sposób aplikacje mogą korzystać z platform i środowisk, które są dostępne dla użytkownika, należy zdecydować, w jaki sposób korzystać z funkcji oferowanych przez produkt WebSphere MQ.

“Przykładowe programy produktu WebSphere MQ” na stronie 98
Ta kolekcja tematów zawiera informacje na temat przykładowych programów WebSphere MQ na różnych platformach.

“Pisanie aplikacji klienckich” na stronie 360
Co należy wiedzieć, aby pisać aplikacje klienckie w produkcie WebSphere MQ.

“Korzystanie z usług Web Service w produkcie WebSphere MQ” na stronie 972
Aplikacje produktu IBM WebSphere MQ dla usług Web Services można tworzyć przy użyciu transportu IBM WebSphere MQ dla protokołu SOAP lub mostu IBM WebSphere MQ dla protokołu HTTP.

“Budowanie aplikacji IBM WebSphere MQ” na stronie 439
Ta sekcja zawiera informacje na temat budowania aplikacji produktu IBM WebSphere MQ na różnych platformach.

“Obsługa błędów programu” na stronie 561
Te informacje wyjaśniają błędy związane z wywołaniami MQI aplikacji, gdy wywołuje połączenie, lub gdy jego komunikat jest dostarczany do miejsca docelowego.

Interfejs kolejki komunikatów-przegląd

Informacje na temat komponentów interfejsu kolejek komunikatów (Message Queue Interface-MQI).

Interfejs kolejki komunikatów składa się z następujących elementów:

- *Wywołania* , za pomocą których programy mogą uzyskiwać dostęp do menedżera kolejek i jego obiektów
- *Struktury* , których programy używają do przekazywania danych do menedżera kolejek i pobierania danych z niego
- *Elementarne typy danych* służące do przekazywania danych do menedżera kolejek i pobierania danych z tego menedżera kolejek.

Produkty WebSphere MQ for Windows i WebSphere MQ w systemach UNIX and Linux również dostarczają:

- Wywołania, za pośrednictwem których programy WebSphere MQ for Windows i WebSphere MQ on UNIX and Linux systems mogą zatwierdzić i wycofać zmiany.
- *Uwzględnij pliki* , które definiują wartości stałych dostarczonych na tych platformach.
- *Pliki biblioteki* , aby połączyć aplikacje.
- Pakiet przykładowych programów, który demonstruje sposób użycia interfejsu MQI na tych platformach. Więcej informacji na temat tych przykładów można znaleźć w sekcji "Przykładowe programy dla platform rozproszonych" na stronie 99.
- Przykładowy kod źródłowy i kod wykonywalny dla powiązań z zewnętrznymi menedżerami transakcji.

Aby dowiedzieć się więcej na temat interfejsu MQI, należy użyć następujących odsyłaczy:

- "Wywołania MQI" na stronie 200
- "Wywołania punktu synchronizacji" na stronie 201
- "Konwersja danych, typy danych, definicje danych i struktury" na stronie 201
- "Programy kodu pośredniczącego i pliki bibliotek produktu IBM WebSphere MQ" na stronie 202
- "Parametry wspólne dla wszystkich wywołań" na stronie 207
- "Określanie buforów" na stronie 208
- "Obsługa sygnału UNIX and Linux" na stronie 208

Pojęcia pokrewne

"Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego" na stronie 211
Aby można było korzystać z usług programistycznych produktu WebSphere MQ , program musi mieć połączenie z menedżerem kolejek. Informacje zawarte w tej sekcji umożliwiają zapoznanie się z informacjami na temat nawiązywania połączenia z menedżerem kolejek i z jego rozłączeniem.

"Otwieranie i zamykanie obiektów" na stronie 220
Te informacje udostępniają wgląd w otwieranie i zamykanie obiektów WebSphere MQ .

"Umieszczanie komunikatów w kolejce" na stronie 231
Te informacje umożliwiają zapoznanie się z informacjami na temat umieszczania komunikatów w kolejce.

"Pobieranie komunikatów z kolejki" na stronie 246
Ta sekcja zawiera informacje na temat pobierania komunikatów z kolejki.

"Sprawdzanie i ustawianie atrybutów obiektu" na stronie 328
Atrybuty to właściwości, które definiują parametry obiektu WebSphere MQ .

"Zatwierdzanie i wycofywanie jednostek pracy" na stronie 331
W tej sekcji opisano sposób zatwierdzania i tworzenia kopii zapasowych wszystkich możliwych do odtworzenia operacji get i put, które wystąpiły w jednostce pracy.

"Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy" na stronie 338
Informacje na temat wyzwalaczy i sposobów uruchamiania aplikacji IBM WebSphere MQ przy użyciu wyzwalaczy.

"Praca z interfejsem MQI i klastrami" na stronie 355
Istnieją specjalne opcje dotyczące wywołań i kodów powrotu, które odnoszą się do technologii klastrowej.

Wywołania MQI

Ta sekcja zawiera informacje na temat wywołań interfejsu MQI.

Wywołania w interfejsie MQI mogą być pogrupowane w następujący sposób:

MQCONN, MQCONNX i MQDISC
: Za pomocą tych wywołań można połączyć program z (z opcjami lub bez) i odłączyć program od menedżera kolejek. Jeśli programy CICS są zapisywane w systemie z/OS, nie ma potrzeby korzystania z tych wywołań. Zaleca się jednak korzystanie z nich, jeśli aplikacja ma zostać portów aplikacji na innych platformach.

MQOPEN i MQCLOSE
: Te wywołania umożliwiają otwarcie i zamknięcie obiektu, takiego jak kolejka.

MQPUT i MQPUT1
Użyj tych wywołań, aby umieścić komunikat w kolejce.

MQGET
Za pomocą tego wywołania można przeglądać komunikaty w kolejce lub usuwać komunikaty z kolejki.

MQSUB, MQSUBRQ
Te wywołania umożliwiają zarejestrowanie subskrypcji tematu i żądanie publikacji zgodnych z subskrypcją.

MQINQ
Użyj tego wywołania, aby dowiedzieć się o atrybutach obiektu.

MQSET
To wywołanie umożliwia ustawienie niektórych atrybutów kolejki. Nie można ustawić atrybutów innych typów obiektów.

MQBEGIN, MQCMIT i MQBACK
Należy użyć tych wywołań, gdy WebSphere MQ jest koordynatorem jednostki pracy. Komenda MQBEGIN uruchamia jednostkę pracy. MQCMIT i MQBACK kończą jednostkę pracy, zatwierdzając lub wycofując aktualizacje wprowadzone podczas jednostki pracy. Używane są komendy rodzimego uruchamiania kontroli transakcji, zatwierdzania i wycofywania zmian.

MQCRTMH, MQBUFMH, MQMHBUF, MQDLTMH
Za pomocą tych wywołań można utworzyć uchwyt komunikatu, przekształcić uchwyt komunikatu w bufor lub bufor na uchwyt komunikatu oraz usunąć uchwyt komunikatu.

MQSETMP, MQINQMP, MQDLTMP
Te wywołania umożliwiają ustawienie właściwości komunikatu na uchwycie komunikatu, zapytanie o właściwość komunikatu i usunięcie właściwości z uchwytu komunikatu.

MQCB, MQCB_FUNCTION, MQCTL
Te wywołania umożliwiają zarejestrowanie i sterowanie funkcją zwrotną.

MQSTAT
To wywołanie służy do pobierania informacji o statusie poprzednich asynchronicznych operacji put.

Opis wywołań MQI zawiera sekcja Opisy wywołań .

Wywołania punktu synchronizacji

Te informacje umożliwiają znalezienie informacji na temat wywołań punktu synchronizacji na różnych platformach.

Wywołania punktu synchronizacji są dostępne w następujący sposób:

Wywołania produktu IBM WebSphere MQ na platformach Windows, UNIX i Linux

Następujące produkty udostępniają wywołania MQCMIT i MQBACK:

- IBM WebSphere MQ dla Windows
- IBM WebSphere MQ w systemach UNIX and Linux

Użyj wywołań punktu synchronizacji w programach, aby poinformować menedżera kolejek o tym, że wszystkie operacje MQGET i MQPUT od ostatniego punktu synchronizacji mają zostać wykonane na stałe (zatwierdzone) lub mają być wycofane. Aby zatwierdzić zmiany i wycofać zmiany w środowisku CICS , należy użyć komend takich jak EXEC CICS SYNCPOINT i EXEC CICS SYNCPOINT ROLLBACK.

Konwersja danych, typy danych, definicje danych i struktury

Ta sekcja zawiera informacje na temat konwersji danych, podstawowych typów danych, definicji danych produktu WebSphere MQ oraz struktur w przypadku korzystania z interfejsu kolejki komunikatów.

Konwersja danych

Wywołanie MQXCNVC (przekształcanie znaków) przekształca dane znakowe komunikatu z jednego zestawu znaków na inny. Z wyjątkiem produktu WebSphere MQ for z/OS, wywołanie to jest używane tylko z wyjścia konwersji danych.

Informacje na temat składni używanej w wywołaniu MQXCNVC zawiera sekcja MQXCNVC-Convert characters (MQXCNVC-Convert characters), a w celu uzyskania wskazówek dotyczących zapisywania i wywoływania wyjść konwersji danych, należy zapoznać się z “Pisanie wyjść konwersji danych” na stronie 426 .

Elementarne typy danych

W przypadku obsługiwanych języków programowania interfejs MQI udostępnia elementarne typy danych lub pola nieustrukturyzowane.

Te typy danych są w pełni opisane w sekcji Typy danych elementarnych.

Definicje danych produktu WebSphere MQ

Pliki definicji danych dostarczane z produktem WebSphere MQ zawierają następujące elementy:

- Definicje wszystkich stałych i kodów powrotu produktu WebSphere MQ
- Definicje struktur i typów danych produktu WebSphere MQ
- Stałe definicje inicjowania struktur
- Prototypy funkcji dla każdego z wywołań (tylko dla języka PL/I i języka C)

Pełny opis plików definicji danych produktu WebSphere MQ znajduje się w sekcji “Pliki definicji danych produktu IBM WebSphere MQ” na stronie 82.

Struktury

Struktury, używane z wywołaniami MQI wymienionymi w sekcji “Wywołania MQI” na stronie 200, są dostarczane w plikach definicji danych dla każdego obsługiwanego języka programowania.

Podsumowanie struktur zawiera sekcja Podsumowanie typów danych struktury .

Programy kodu pośredniczącego i pliki bibliotek produktu IBM WebSphere MQ

W tym miejscu wyświetlane są programy pośredniczenia i pliki bibliotek dla każdej platformy.

Więcej informacji na temat korzystania z programów pośredniczących i plików bibliotek podczas budowania aplikacji wykonywalnej zawiera sekcja “Budowanie aplikacji IBM WebSphere MQ” na stronie 439. Więcej informacji na temat łączenia z plikami biblioteki C++ zawiera sekcja Używanie języka C++ *WebSphere MQ Korzystanie z języka C++*.

IBM WebSphere MQ dla Windows

W systemie IBM WebSphere MQ dla systemu Windowsnależy połączyć program z plikami biblioteki MQI dostarczonym dla środowiska, w którym jest uruchamiana aplikacja, oprócz tych udostępnionych przez system operacyjny:

Tabela 25. Pliki bibliotek dla aplikacji Windows

Plik biblioteki	**Środowisko**
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	Serwer dla C (32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	Klient dla C (32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib\mqmxa.lib	Interfejs XA serwera dla języka C (32-bitowego)
MQ_INSTALLATION_PATH\Tools\Lib\mqcxa.lib	Interfejs klienta XA dla języka C (32-bitowego)

Tabela 25. Pliki bibliotek dla aplikacji Windows (kontynuacja)	
Plik biblioteki	**Środowisko**
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	Klient MTS dla C (32-bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcics4.lib	Obsługa serwera TXSeries CICS dla języka C (wersja 32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib\mqccics4.lib	Obsługa klienta TXSeries CICS dla języka C (32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib\mqmzf.lib	Wyjścia usług instalowalnych dla języka C (32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb.lib	Serwer dla IBM COBOL (wersja 32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb.lib	Serwer dla Micro Focus COBOL (32-bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb.lib	Klient dla IBM COBOL (wersja 32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb.lib	Klient dla Micro Focus COBOL (32-bit)
MQ_INSTALLATION_PATH\Tools\Lib\imqs23vn.lib	Serwer dla języka C++ (32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib\imqc23vn.lib	Klient dla języka C++ (32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib\imqb23vn.lib	Podstawowa dla języka C++ (32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib\imqx23vn.lib	Klient MTS dla C++ (32-bitowa)
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	Serwer dla języka C (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	Klient dla języka C (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmxa.lib	Interfejs XA serwera dla języka C (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\mqcxa.lib	Interfejs klienta XA dla języka C (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	Klient MTS for C (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb.lib	Serwer dla IBM COBOL (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb.lib	Serwer dla Micro Focus COBOL (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb.lib	Klient dla IBM COBOL (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb.lib	Klient dla Micro Focus COBOL (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\imqs23vn.lib	Serwer dla C++ (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\imqc23vn.lib	Klient dla C++ (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\imqb23vn.lib	Podstawowy dla C++ (64-bitowy)
MQ_INSTALLATION_PATH\Tools\Lib64\imqx23vn.lib	Klient MTS dla C++ (64-bitowy)

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

Program `amqmdnet.dll` służy do kompilowania programów .NET. Więcej informacji na ten temat zawiera sekcja “Kompilowanie programów WebSphere MQ .NET” na stronie 608 w sekcji “Używanie środowiska .NET” na stronie 573 .

Pliki te są dostarczane pod kątem zgodności z poprzednimi wersjami:

mqic32.lib
mqic32xa.lib

IBM WebSphere MQ dla AIX

W systemie IBM WebSphere MQ dla systemu AIXnależy połączyć program z plikami biblioteki MQI dostarczonym dla środowiska, w którym aplikacja jest uruchamiana, oprócz tych udostępnionych przez system operacyjny.

W aplikacji, która nie jest wielowątkowa:

Tabela 26. Pliki bibliotek dla niewielowątkowych aplikacji systemu AIX

Zbiór biblioteki	**Środowisko**
libmqm.a	Serwer dla C
libmqic.a & **libmqm.a**	Klient dla C
libmqmzf.a	Możliwe do zainstalowania wyjścia usługi dla języka C
libmqmxa.a	Interfejs XA serwera
libmqmxa64.a	Alternatywny interfejs XA serwera
libmqcxa.a	Interfejs klienta XA
libmqcxa64.a	Alternatywny interfejs XA klienta
libmqmcbrt.o	Biblioteka środowiska wykonawczego produktu WebSphere MQ dla obsługi języków Micro Focus COBOL
libmqmcb.a	Serwer dla języka COBOL
libmqicb.a	Klient dla języka COBOL
libimqc23ia.a	Klient dla C++
libimqs23ia.a	Serwer dla języka C++

W aplikacji wielowątkowej:

Tabela 27. Pliki bibliotek dla wielowątkowych aplikacji AIX

Zbiór biblioteki	**Środowisko**
libmqm_r.a	Serwer dla C
libmqic_r.a & **libmqm_r.a**	Klient dla C
libmqmzf_r.a	Możliwe do zainstalowania wyjścia usługi dla języka C
libmqmxa_r.a	Interfejs XA serwera
libmqmxa64_r.a	Alternatywny interfejs XA serwera
libmqcxa_r.a	Interfejs klienta XA
libmqcxa64_r.a	Alternatywny interfejs XA klienta

Tabela 27. Pliki bibliotek dla wielowątkowych aplikacji AIX (kontynuacja)	
Zbiór biblioteki	**Środowisko**
libimqc23ia_r.a	Klient dla C++
libimqs23ia_r.a	Serwer dla języka C++

IBM WebSphere MQ dla systemu HP-UX

W systemie IBM WebSphere MQ dla systemu HP-UXnależy połączyć program z plikami biblioteki MQI dostarczonym dla środowiska, w którym aplikacja jest uruchamiana, oprócz tych udostępnionych przez system operacyjny.

Platforma IA64 (IPF)

W aplikacji, która nie jest wielowątkowa:

Tabela 28. Pliki bibliotek dla niewątkowych aplikacji HP-UX	
Zbiór biblioteki	**Środowisko**
libmqm.so	Serwer dla C
libmqic.so & **libmqm.so**	Klient dla C
libmqmzf.so	Możliwe do zainstalowania wyjścia usługi dla języka C
libmqmxa.so	Interfejs XA serwera
libmqmxa64.so	Alternatywny interfejs XA serwera
libmqcxa.so	Interfejs klienta XA
libmqcxa64.so	Alternatywny interfejs XA klienta
libimqi23ah.so	C++
libmqmcbrt.o	Biblioteka środowiska wykonawczego produktu WebSphere MQ dla obsługi języków Micro Focus COBOL
libmqmcb.so	Serwer dla języka COBOL
libmqicb.so	Klient dla języka COBOL

W aplikacji wielowątkowej:

Tabela 29. Pliki bibliotek dla wielowątkowych aplikacji HP-UX	
Zbiór biblioteki	**Środowisko**
libmqm_r.so	Serwer dla C
libmqmzf_r.so & **libmqm_r.so**	Możliwe do zainstalowania wyjścia usługi dla języka C
libmqmxa_r.so	Interfejs XA serwera
libmqmxa64_r.so	Alternatywny interfejs XA serwera
libmqcxa_r.so	Interfejs klienta XA
libmqcxa64_r.so	Alternatywny interfejs XA klienta
libimqi23ah_r.so	C++

IBM WebSphere MQ dla Linux

W systemie IBM WebSphere MQ for Linuxnależy połączyć program z plikami biblioteki MQI dostarczonym dla środowiska, w którym aplikacja jest uruchamiana, oprócz tych udostępnionych przez system operacyjny.

W aplikacji, która nie jest wielowątkowa:

Tabela 30. Pliki bibliotek dla niewielowątkowych aplikacji produktu Linux

Zbiór biblioteki	**Środowisko**
libmqm.so	Serwer dla C
libmqic.so & **libmqm.so**	Klient dla C
libmqmzf.so	Możliwe do zainstalowania wyjścia usługi dla języka C
libmqmxa.so	Interfejs XA serwera
libmqmxa64.so	Alternatywny interfejs XA serwera
libmqcxa.so	Interfejs klienta XA
libmqcxa64.so	Alternatywny interfejs XA klienta
libimqc23gl.so	Klient dla C++
libimqs23gl.so	Serwer dla języka C++

W aplikacji wielowątkowej:

Tabela 31. Pliki bibliotek dla wielowątkowych aplikacji produktu Linux

Zbiór biblioteki	**Środowisko**
libmqm_r.so	Serwer dla C
libmqic_r.so & **libmqm_r.so**	Klient dla C
libmqmzf_r.so	Możliwe do zainstalowania wyjścia usługi dla języka C
libmqmxa_r.so	Interfejs XA serwera
libmqmxa64_r.so	Alternatywny interfejs XA serwera
libmqcxa_r.so	Interfejs klienta XA
libmqcxa64_r.so	Alternatywny interfejs XA klienta
libimqc23gl_r.so	Klient dla C++
libimqs23gl_r.so	Serwer dla języka C++

IBM WebSphere MQ dla systemu Solaris

W systemie IBM WebSphere MQ dla systemu Solaris należy połączyć program z plikami biblioteki MQI dostarczonym dla środowiska, w którym działa aplikacja, oprócz tych, które są udostępniane przez system operacyjny.

Tabela 32. Pliki bibliotek dla aplikacji Solaris

Plik biblioteki	**Środowisko**
libmqm.so	Serwer i klient dla języka C
libmqmzse.so	Dla C

Tabela 32. Pliki bibliotek dla aplikacji Solaris (kontynuacja)	
Plik biblioteki	**Środowisko**
libmqic.so	Klient dla C
libmqmcs.so	Wspólne usługi dla C
libmqmzf.so	Możliwe do zainstalowania wyjścia usługi dla języka C
libmqmxa.so	Interfejs XA serwera
libmqmxa64.so	Alternatywny interfejs XA serwera
libmqcxa.so	Interfejs klienta XA
libmqcxa64.so	Alternatywny interfejs XA klienta
libimqc23as.a	Klient dla C++
libimqs23as.a	Serwer dla języka C++

Parametry wspólne dla wszystkich wywołań

Istnieją dwa typy parametrów wspólne dla wszystkich wywołań: uchwyty i kody powrotu.

Używanie uchwytów

Wszystkie wywołania MQI używają jednego lub więcej *uchwytów*. Identyfikują one menedżer kolejek, kolejkę lub inny obiekt, komunikat lub subskrypcję, odpowiednio do wywołania.

Aby program mógł komunikować się z menedżerem kolejek, program musi mieć unikalny identyfikator, za pomocą którego wie on, że menedżer kolejek jest poprawny. Ten identyfikator jest nazywany *uchwytem połączenia*, czasem nazywanych *Hconn*. W przypadku programów CICS uchwyt połączenia jest zawsze równy zero. W przypadku wszystkich innych platform lub stylów programów uchwyt połączenia jest zwracany przez wywołanie MQCONN lub MQCONNX, gdy program łączy się z menedżerem kolejek. Programy przekazują uchwyt połączenia jako parametr wejściowy, gdy korzystają z innych wywołań.

Aby program działał z obiektem WebSphere MQ , program musi mieć unikalny identyfikator, za pomocą którego obiekt ten wie, że obiekt jest obiektem. Ten identyfikator jest nazywany *uchwytem obiektu*, czasem nazywanych *Hobj*. Uchwyt jest zwracany przez wywołanie MQOPEN, gdy program otworzy obiekt do pracy z nim. Programy przekazują uchwyt obiektu jako parametr wejściowy, gdy korzystają z kolejnych wywołań MQPUT, MQGET, MQINQ, MQSET lub MQCLOSE.

Podobnie wywołanie MQSUB zwraca *uchwyt subskrypcji* lub *Hsub*, który jest używany do identyfikowania subskrypcji w kolejnych wywołaniach MQGET, MQCB lub MQSUBRQ, a niektóre wywołania przetwarzania komunikatów wymagają użycia *uchwytu komunikatu* lub *Hmsg*.

Kody powrotu zrozumienia

Kod zakończenia i kod przyczyny są zwracane przez każde wywołanie jako parametry wyjściowe. Są to znane zbiorczo *kody powrotu*.

Aby określić, czy wywołanie powiodło się, każde wywołanie zwraca *kod zakończenia* po zakończeniu wywołania. Kod zakończenia zwykle ma wartość MQCC_OK wskazującą powodzenie lub MQCC_FAILED wskazuje niepowodzenie. Niektóre wywołania mogą zwrócić stan pośredni, MQCC_WARNING, wskazując częściowy sukces.

Każde wywołanie zwraca także *kod przyczyny* , który przedstawia przyczynę niepowodzenia wywołania lub jego częściowy sukces. Istnieje wiele kodów przyczyny, obejmujących takie okoliczności, jak kolejka jest pełna, operacje pobierania nie są dozwolone dla kolejki, a konkretna kolejka nie jest zdefiniowana dla menedżera kolejek. Programy mogą używać kodu przyczyny w celu podjęcia decyzji o sposobie postępowania. Na przykład użytkownicy mogą podpowiadać użytkownikom zmianę danych wejściowych, a następnie ponownie wywołać wywołanie lub zwrócić użytkownikowi komunikat o błędzie.

Jeśli kod zakończenia ma wartość MQCC_OK, kodem przyczyny jest zawsze MQRC_NONE.

Kody zakończenia i przyczyny dla każdego wywołania są wymienione wraz z opisem tego wywołania. Zapoznaj się z opisami wywołań i wybierz odpowiednie wywołanie z listy.

Więcej szczegółowych informacji, w tym pomysłów na działania naprawcze, można znaleźć w:

- Kody przyczyn dla wszystkich pozostałych platform WebSphere MQ

Określanie buforów

Menedżer kolejek odwołuje się do buforów tylko wtedy, gdy są one wymagane. Jeśli w wywołaniu nie jest wymagany bufor lub bufor ma wartość zerową, można użyć pustego wskaźnika do buforu.

Zawsze używaj długości fali podczas określania wielkości buforu, który jest wymagany.

Jeśli używany jest bufor do przechowywania danych wyjściowych z wywołania (na przykład w celu przechowywania danych komunikatu dla wywołania MQGET lub wartości atrybutów, których dotyczy wywołanie MQINQ), menedżer kolejek próbuje zwrócić kod przyczyny, jeśli podany bufor nie jest poprawny lub jest w pamięci masowej tylko do odczytu. Może jednak nie zawsze być w stanie zwrócić kod przyczyny.

Uwagi dotyczące produktu UNIX and Linux

Uwagi, o których należy pamiętać.

Podczas tworzenia aplikacji produktu UNIX and Linux należy pamiętać o następujących kwestiach.

Wywołanie systemowe fork w systemach UNIX and Linux

Należy zwrócić uwagę na te uwagi w przypadku korzystania z wywołania systemowego fork w aplikacjach IBM WebSphere MQ .

Jeśli aplikacja chce używać produktu `fork`, proces nadrzędny tej aplikacji powinien wywoływać produkt `fork` przed nawiązaniem wszystkich wywołań produktu IBM WebSphere MQ , na przykład `MQCONN`, lub w celu utworzenia obiektu IBM WebSphere MQ za pomocą programu **`ImqQueueManager`**.

Jeśli aplikacja chce utworzyć proces potomny po wprowadzeniu wszystkich wywołań produktu IBM WebSphere MQ , kod aplikacji musi używać `fork()` z `exec()` , aby upewnić się, że jest to nowa instancja, a nie dokładna kopia elementu nadrzędnego.

Jeśli aplikacja nie korzysta z produktu `exec()`, wywołanie funkcji API IBM WebSphere MQ wykonane w procesie potomnym zwraca wartość MQRC_ENVIRONMENT_ERROR.

Obsługa sygnału UNIX and Linux

Nie dotyczy to produktu WebSphere MQ for z/OS ani produktu WebSphere MQ for Windows.

W systemach ogólnych, UNIX, Linux i IBM i przeniesiono z środowiska wielowątkowego do środowiska wielowątkowego. W środowisku bez gwintów niektóre funkcje mogą być zaimplementowane tylko za pomocą sygnałów, choć większość aplikacji nie musi być świadoma sygnałów i obsługi sygnałów. W środowisku wielowątkowym operacje podstawowe oparte na wątkach obsługują niektóre funkcje, które są używane do implementowania w środowiskach, które nie są gwintowane, przy użyciu sygnałów.

W wielu przypadkach sygnały i obsługa sygnałów, mimo że są obsługiwane, nie mieszczą się dobrze w środowisku wielowątkowym i istnieją różne ograniczenia. Może to być problematyczne podczas integrowania kodu aplikacji z różnymi bibliotekami oprogramowania pośredniego (działaniowymi jako część aplikacji) w środowisku wielowątkowym, w którym każdy próbuje obsłużyć sygnały. Tradycyjne podejście do zapisywania i odtwarzania procedur obsługi sygnału (zdefiniowanych dla każdego procesu), które pracowało w sytuacji, gdy w procesie jest tylko jeden wątek wykonywania, nie działa w środowisku wielowątkowym. Jest to spowodowane tym, że wiele wątków wykonywania może próbować zapisać i odtworzyć zasób o zasięgu całego procesu, którego wyniki są nieprzewidywalne.

Aplikacje wielowątkowe
Nie dotyczy systemu Solaris, ponieważ wszystkie aplikacje są traktowane jako wątki wielowątkowe, nawet jeśli używają tylko jednego wątku.

Każda funkcja MQI służy do konfigurowania własnej procedury obsługi sygnału dla sygnałów:

SIGALRM
SIGBUS
SIGFPE
SIGSEGV
SIGILL

Procedury obsługi użytkowników dla tych elementów są zastępowane przez czas trwania wywołania funkcji MQI. Inne sygnały mogą być wychwytywać w normalny sposób przez osoby zajmujące się obsługą pisemną. Jeśli program obsługi nie zostanie zainstalowany, zostaną wykonane działania domyślne (na przykład ignorowanie, zrzut jądra lub wyjście).

Po WebSphere MQ obsługuje sygnał synchroniczny (SIGSEGV, SIGBUS, SIGFPE, SIGILL), który próbuje przekazać sygnał do dowolnej zarejestrowanej procedury obsługi sygnału przed wywołaniem funkcji MQI.

Aplikacje gwintowane
Wątek jest uznawany za połączony z produktem WebSphere MQ za pomocą MQCONN (lub MQCONNX) do momentu wywołania MQDISC.

Sygnały synchroniczne

Sygnały synchroniczne pojawiają się w określonym wątku.

Systemy UNIX and Linux bezpiecznie zezwalają na konfigurowanie procedury obsługi sygnału dla takich sygnałów dla całego procesu. Produkt WebSphere MQ konfiguruje jednak własną procedurę obsługi dla następujących sygnałów w procesie aplikacji, podczas gdy dowolny wątek jest połączony z produktem WebSphere MQ:

SIGBUS
SIGFPE
SIGSEGV
SIGILL

Jeśli piszesz aplikacje wielowątkowe, dla każdego sygnału istnieje tylko jedna procedura obsługi sygnału dla całego procesu. Jeśli produkt WebSphere MQ konfiguruje własne procedury obsługi sygnałów synchronicznych, to zapisuje wszystkie zarejestrowane procedury obsługi dla każdego sygnału. Po tym, jak produkt WebSphere MQ obsługuje jeden z wymienionych sygnałów, produkt WebSphere MQ próbuje wywołać procedurę obsługi sygnału, która była skuteczna w momencie pierwszego połączenia WebSphere MQ w ramach procesu. Poprzednio zarejestrowane procedury obsługi są odtwarzane, gdy wszystkie wątki aplikacji są odłączane od produktu WebSphere MQ.

Ponieważ procedury obsługi sygnałów są składowane i odtwarzane przez program WebSphere MQ, wątki aplikacji nie mogą ustanawiać procedur obsługi sygnałów dla tych sygnałów, podczas gdy istnieje jakakolwiek możliwość, że inny wątek tego samego procesu jest również połączony z produktem WebSphere MQ.

Uwaga: Gdy aplikacja lub biblioteka oprogramowania pośredniego (działająca jako część aplikacji) tworzy procedurę obsługi sygnału, gdy wątek jest połączony z produktem WebSphere MQ, procedura obsługi sygnału aplikacji musi wywoływać odpowiednią procedurę obsługi WebSphere MQ podczas przetwarzania tego sygnału.

Przy ustanawianiu i odtwarzaniu procedur obsługi sygnału, generalna zasada jest taka, że ostatnia procedura obsługi sygnału, która ma być składowana, musi być pierwszą, która ma zostać odtworzona:

- Gdy aplikacja tworzy procedurę obsługi sygnału po nawiązaniu połączenia z produktem WebSphere MQ, poprzednia procedura obsługi sygnału musi zostać odtworzona przed rozłączeniu aplikacji z produktu WebSphere MQ.

- Gdy aplikacja tworzy procedurę obsługi sygnału przed nawiązaniem połączenia z produktem WebSphere MQ, przed odtworzeniem procedury obsługi sygnału aplikacja musi odłączyć się od produktu WebSphere MQ .

Uwaga: Nieprzestrzeganie ogólnej zasady, zgodnie z którą ostatnia procedura obsługi sygnału, która ma być składowana, musi być pierwszą, która ma być odtworzona, może skutkować nieoczekiwaną obsługą sygnału w aplikacji i, potencjalnie, utratą sygnałów przez aplikację.

Sygnały asynchroniczne

Produkt WebSphere MQ nie używa żadnych sygnałów asynchronicznych w aplikacjach wielowątkowych, chyba że są to aplikacje klienckie.

Dodatkowe uwagi dotyczące wielowątkowych aplikacji klienckich

Produkt WebSphere MQ obsługuje następujące sygnały podczas operacji we/wy na serwerze. Sygnały te są definiowane przez stos komunikacji. Aplikacja nie może ustanawiać procedury obsługi sygnału dla tych sygnałów, gdy wątek jest połączony z menedżerem kolejek:

SIGPIPE (dla TCP/IP)

Dodatkowe uwarunkowania
Należy zwrócić uwagę na te uwagi podczas korzystania z obsługi sygnału systemu UNIX .

Aplikacje typu fastpath (zaufane)

Aplikacje krótkiej ścieżki działają w tym samym procesie, co produkt WebSphere MQ , a więc działają w środowisku wielowątkowym.

W tym środowisku produkt WebSphere MQ obsługuje sygnały synchroniczne SIGSEGV, SIGBUS, SIGFPE i SIGILL. Wszystkie inne sygnały nie mogą być dostarczane do aplikacji Fastpath, podczas gdy jest ona połączona z produktem WebSphere MQ. Zamiast tego muszą być zablokowane lub obsługiwane przez aplikację. Jeśli aplikacja Fastpath przechwyci takie zdarzenie, menedżer kolejek musi zostać zatrzymany i zrestartowany lub może zostać w niezdefiniowanym stanie. Pełną listę ograniczeń dotyczących aplikacji Fastpath w produkcie MQCONNX można znaleźć w sekcji “Nawiąże połączenie z menedżerem kolejek przy użyciu wywołania MQCONNX” na stronie 214.

Wywołania funkcji MQI w procedurach obsługi sygnału

Gdy użytkownik jest w procedurze obsługi sygnału, nie wywoła funkcji MQI.

W przypadku próby wywołania funkcji MQI z procedury obsługi sygnału w czasie, gdy aktywna jest inna funkcja MQI, zwracana jest wartość MQRC_CALL_IN_PROGRESS. Próba wywołania funkcji MQI z procedury obsługi sygnału, gdy żadna inna funkcja MQI nie jest aktywna, prawdopodobnie nie powiedzie się w czasie operacji ze względu na ograniczenia systemu operacyjnego, w którym mogą być wysyłane tylko wywołania selektywne z programu obsługi lub w jego obrębie.

W przypadku metod destruktorów C + +, które mogą być wywoływane automatycznie podczas programu obsługi wyjścia programu, użytkownik może nie być w stanie zatrzymać funkcji MQI z wywołania. Zignoruj wszystkie błędy związane z plikiem MQRC_CALL_IN_PROGRESS. Jeśli wywołanie procedury obsługi sygnału wywoła wyjście (), program WebSphere MQ wycofuje niezatwierdzone komunikaty w punkcie synchronizacji, jak zwykle i zamyka wszystkie otwarte kolejki.

Sygnały podczas wywołań MQI

Funkcje MQI nie zwracają kodu EINTR ani żadnego odpowiednika dla programów aplikacji.

Jeśli podczas wywołania MQI wystąpi sygnał, a procedura obsługi wywołuje *return*, wywołanie kontynuuje działanie tak, jakby sygnał nie miał miejsca. W szczególności MQGET nie może zostać przerwany przez sygnał, aby natychmiast zwrócić kontrolę do aplikacji. Aby przerwać operację MQGET, należy ustawić kolejkę na wartość GET_DISABLED; alternatywnie użyć pętli wokół wywołania MQGET ze skończonym

czasem utraty ważności (MQGMO_WAIT z opcją gmo.WaitInterval) i użyć procedury obsługi sygnału (w środowisku bez połączenia z wątkiem) lub funkcji równoważnej w środowisku wielowątkowym, aby ustawić flagę, która pęka pętlę.

W środowisku AIX produkt WebSphere MQ wymaga, aby wywołania systemowe przerywane przez sygnały były restartowane. Podczas ustanawiania własnej procedury obsługi sygnału za pomocą sigaction (2), należy ustawić flagę SA_RESTART w polu sa_flags nowej struktury działania, w przeciwnym razie program WebSphere MQ może nie być w stanie zakończyć żadnego wywołania przerwanego przez sygnał.

Programy zewnętrzne i usługi instalacyjne

Procedury zewnętrzne i instalacyjne, które są uruchamiane jako część procesu WebSphere MQ w środowisku wielowątkowym, mają takie same ograniczenia, jak w przypadku aplikacji fastpath. Należy wziąć pod uwagę te, które mają być trwale połączone z produktem WebSphere MQ , a więc nie są używane sygnały lub wywołania systemu operacyjnego inne niż wątkowo bezpieczne.

Procedury obsługi wyjścia VMS

Użytkownicy mogą instalować procedury obsługi wyjścia dla aplikacji WebSphere MQ za pomocą usługi systemowej **SYS$DCLEXH** .

Procedura obsługi wyjścia odbiera sterowanie przy wyjściu z obrazu. Wyjście obrazu zwykle występuje, gdy użytkownik wywoła usługę Exit ($EXIT) lub Force Exit ($FORCEX). $FORCEX przerywa proces docelowy w trybie użytkownika. Następnie wszystkie procedury obsługi wyjścia trybu użytkownika (ustalone przez $DCLEXH) zaczynają być wykonywane w odwrotnej kolejności tworzenia. Więcej informacji na temat procedur obsługi wyjścia i $FORCEX można znaleźć w podręczniku *VMS Programming Concepts Manual* (Podręcznik programowania VMS) oraz w podręczniku *VMS System Services Manual*(Podręcznik usług systemowych VMS).

Jeśli funkcja MQI jest wywoływana z poziomu procedury obsługi wyjścia, zachowanie funkcji zależy od sposobu, w jaki obraz został zakończony. Jeśli obraz został przerwany, a inna funkcja MQI jest aktywna, zwracana jest wartość `MQRC_CALL_IN_PROGRESS` .

Istnieje możliwość wywołania funkcji MQI z poziomu procedury obsługi wyjścia, jeśli żadna inna funkcja MQI nie jest aktywna, a wywołania upcall są wyłączone dla aplikacji WebSphere MQ . Jeśli dla aplikacji WebSphere MQ włączone są wywołania upcalls, nie powiedzie się ona z kodem przyczyny `MQRC_HCONN_ERROR`.

Zasięg wywołania MQCONN lub MQCONNX jest zwykle wątkiem, który go wydał. Jeśli włączone są wywołania upcalls, procedura obsługi wyjścia jest uruchamiana jako osobny wątek, a uchwyty połączeń nie mogą być współużytkowane.

Procedury obsługi wyjścia są uruchamiane w ramach przerwanego kontekstu procesu docelowego. Do aplikacji należy dopilnowanie, aby działania podejmowane przez procedurę obsługi były bezpieczne i niezawodne w przypadku asynchronicznie przerwanego kontekstu, z którego są wywoływane.

Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego

Aby można było korzystać z usług programistycznych produktu WebSphere MQ , program musi mieć połączenie z menedżerem kolejek. Informacje zawarte w tej sekcji umożliwiają zapoznanie się z informacjami na temat nawiązywania połączenia z menedżerem kolejek i z jego rozłączeniem.

Sposób, w jaki to połączenie jest nawiązany, zależy od platformy i środowiska, w którym działa program:

z/OS batch, WebSphere MQ for IBM i, WebSphere MQ on UNIX systems, WebSphere MQ on Linux systems, and WebSphere MQ for Windows

Programy, które działają w tych środowiskach, mogą używać wywołania MQCONN MQI, z którym można nawiązać połączenie, oraz wywołania MQDISC, z którym ma zostać odłączone połączenie, a także menedżer kolejek. Alternatywnie programy mogą używać wywołania MQCONNX.

Program wsadowy z/OS może łączyć się, kolejno lub współbieżnie, z wieloma menedżerami kolejek w tym samym TCB.

IMS

Region sterujący IMS jest połączony z jednym lub większą liczbę menedżerów kolejek po jego uruchomieniu. To połączenie jest kontrolowane przez komendy IMS . Jednak programy piszące komunikaty kolejkowania komunikatów IMS muszą używać wywołania MQCONN MQI w celu określenia menedżera kolejek, z którym mają zostać nawiązane połączenie. Mogą one używać wywołania MQDISC do rozłączenia się z tym menedżerem kolejek.

Po wywołaniu programu IMS , który ustanawia punkt synchronizacji, oraz przed przetwornikiem komunikatu dla innego użytkownika, adapter IMS zapewnia, że aplikacja zamknie uchwyty i rozłącza się z menedżerem kolejek.

Programy IMS mogą łączyć się, kolejno lub współbieżnie, z wieloma menedżerami kolejek w tym samym TCB.

CICS Transaction Server for z/OS and CICS for MVS/ESA

Programy CICS nie muszą wykonywać żadnej pracy w celu nawiązania połączenia z menedżerem kolejek, ponieważ jest on połączony z systemem CICS . To połączenie jest zwykle wykonywane automatycznie podczas inicjowania, ale można również użyć transakcji CKQC, która jestDostarczany razem z produktem WebSphere MQ for z/OS.

Zadania CICS mogą łączyć się tylko z menedżerem kolejek, z którym połączony jest region CICS .

Uwaga: Programy CICS mogą również korzystać z wywołań interfejsu MQI i rozłączania (MQCONN i MQDISC). Może to być konieczne, aby można było portów tych aplikacji w środowiskach innych niż CICS, które mają minimum rekodowania. Jednak te wywołania *zawsze* są zakończone pomyślnie w środowisku CICS . Oznacza to, że kod powrotu może nie odzwierciedlać rzeczywistego stanu połączenia z menedżerem kolejek.

TXSeries dla systemów Windows i Open Systems

Programy te nie muszą wykonywać żadnej pracy w celu nawiązania połączenia z menedżerem kolejek, ponieważ jest on połączony z systemem CICS . W związku z tym obsługiwane jest tylko jedno połączenie w danym momencie. Aplikacje CICS muszą wywoływać wywołanie MQCONN w celu uzyskania uchwytu połączenia, a wywołanie MQDISC przed ich wyjściem.

Użyj poniższych odsyłaczy, aby dowiedzieć się więcej na temat połączenia i rozłączenia z menedżerem kolejek:

- “Nawiąże połączenie z menedżerem kolejek przy użyciu wywołania MQCONN” na stronie 213
- “Nawiąże połączenie z menedżerem kolejek przy użyciu wywołania MQCONNX” na stronie 214
- “Rozłączanie programów z menedżera kolejek za pomocą MQDISC” na stronie 219

Pojęcia pokrewne

“Interfejs kolejki komunikatów-przegląd” na stronie 199
Informacje na temat komponentów interfejsu kolejek komunikatów (Message Queue Interface-MQI).

“Otwieranie i zamykanie obiektów” na stronie 220
Te informacje udostępniają wgląd w otwieranie i zamykanie obiektów WebSphere MQ .

“Umieszczanie komunikatów w kolejce” na stronie 231
Te informacje umożliwiają zapoznanie się z informacjami na temat umieszczania komunikatów w kolejce.

“Pobieranie komunikatów z kolejki” na stronie 246
Ta sekcja zawiera informacje na temat pobierania komunikatów z kolejki.

“Sprawdzanie i ustawianie atrybutów obiektu” na stronie 328
Atrybuty to właściwości, które definiują parametry obiektu WebSphere MQ .

“Zatwierdzanie i wycofywanie jednostek pracy” na stronie 331
W tej sekcji opisano sposób zatwierdzania i tworzenia kopii zapasowych wszystkich możliwych do odtworzenia operacji get i put, które wystąpiły w jednostce pracy.

“Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy” na stronie 338
Informacje na temat wyzwalaczy i sposobów uruchamiania aplikacji IBM WebSphere MQ przy użyciu wyzwalaczy.

“Praca z interfejsem MQI i klastrami” na stronie 355
Istnieją specjalne opcje dotyczące wywołań i kodów powrotu, które odnoszą się do technologii klastrowej.

Nawiąże połączenie z menedżerem kolejek przy użyciu wywołania MQCONN

Informacje zawarte w tej sekcji umożliwiają poznanie sposobu łączenia się z menedżerem kolejek przy użyciu wywołania MQCONN.

W ogólnym przypadku można nawiązać połączenie z określonym menedżerem kolejek lub z domyślnym menedżerem kolejek:

- W przypadku produktu IBM WebSphere MQ for z/OSw środowisku wsadowym domyślny menedżer kolejek jest określony w module CSQBDEFV.
- W przypadku systemów IBM WebSphere MQ dla systemów Windows, IBM i, UNIXi Linux domyślny menedżer kolejek jest określony w pliku mqs.ini .

Alternatywnie w środowiskach z/OS MVS , TSO i RRS można nawiązać połączenie z dowolnym menedżerem kolejek w ramach grupy współużytkowania kolejki. Żądanie MQCONN lub MQCONNX wybiera jeden z aktywnych elementów grupy.

Po nawiązaniu połączenia z menedżerem kolejek musi on być lokalny względem zadania. Musi on należeć do tego samego systemu, co aplikacja IBM WebSphere MQ .

W środowisku IMS menedżer kolejek musi być połączony z regionem sterującym IMS i z regionem zależnym, z którego korzysta program. Domyślny menedżer kolejek jest określony w module CSQQDEFV, gdy zainstalowany jest produkt IBM WebSphere MQ for z/OS .

W przypadku środowiska TXSeries CICS i parametru TXSeries dla produktów Windows i AIXmenedżer kolejek musi być zdefiniowany jako zasób XA w produkcie CICS.

Aby nawiązać połączenie z domyślnym menedżerem kolejek, należy wywołać MQCONN, określając nazwę składającą się całkowicie z odstępów lub rozpoczynając od znaku o wartości NULL (X'00 ').

Aplikacja musi być autoryzowana, aby można było pomyślnie nawiązać połączenie z menedżerem kolejek. Więcej informacji na ten temat zawiera sekcja Zabezpieczenia.

Dane wyjściowe MQCONN są następujące:

- Uchwyt połączenia (**Hconn**)
- Kod zakończenia
- Kod przyczyny

Użyj uchwytu połączenia przy kolejnych wywołaniach MQI.

Jeśli kod przyczyny wskazuje, że aplikacja jest już połączona z tym menedżerem kolejek, zwracany uchwyt połączenia jest taki sam, jak ten, który został zwrócony podczas pierwszego połączenia z aplikacją. Aplikacja nie może wywołać wywołania MQDISC w tej sytuacji, ponieważ aplikacja wywołująca oczekuje, że pozostanie połączona.

Zakres uchwytu połączenia jest taki sam, jak zasięg uchwytu obiektu (patrz “Otwieranie obiektów za pomocą wywołania MQOPEN” na stronie 221).

Opisy parametrów są podane w opisie wywołania MQCONN w MQCONN.

Wywołanie MQCONN kończy się niepowodzeniem, jeśli menedżer kolejek jest w stanie wygaszania podczas wydawania wywołania lub gdy menedżer kolejek jest zamykany.

Zasięg MQCONN lub MQCONNX

Zasięg wywołania MQCONN lub MQCONNX jest zwykle wątkiem, który go wydał. Oznacza to, że uchwyt połączenia zwrócony z wywołania jest poprawny tylko w obrębie wątku, który wywołał wywołanie. Tylko jedno połączenie może być wykonane w dowolnym momencie za pomocą uchwytu. Jeśli jest używany z innego wątku, jest on odrzucany jako niepoprawny. Jeśli w aplikacji znajduje się wiele wątków, a każda

z nich chce korzystać z wywołań programu IBM WebSphere MQ , każda z nich musi wydać komendę MQCONN lub MQCONNX.

Nie jest konieczne, aby każde wywołanie było wykonywane w tym samym menedżerze kolejek, gdy proces wykonuje wiele wywołań MQCONN. Jednak tylko jedno połączenie WebSphere MQ może być wykonane z wątku w danym momencie. Alternatywnie można rozważyć użycie produktu "Połączenia współużytkowane (niezależne od wątku) z produktem MQCONNX" na stronie 217 w celu zezwolenia na wiele połączeń produktu WebSphere MQ z jednego wątku i połączenia produktu WebSphere MQ , które mają być używane z dowolnego wątku.[1]

Jeśli aplikacja działa jako klient, może on łączyć się z więcej niż jednym menedżerem kolejek w obrębie wątku.

Nawiąże połączenie z menedżerem kolejek przy użyciu wywołania MQCONNX

Wywołanie MQCONNX jest podobne do wywołania MQCONN, ale zawiera opcje służące do sterowania sposobem działania wywołania.

Jako dane wejściowe do tabeli MQCONNX można podać nazwę menedżera kolejek lub nazwę grupy współużytkowania kolejki w systemach współużytkowanych kolejek systemu z/OS . Dane wyjściowe komendy MQCONNX są następujące:

- Uchwyt połączenia (Hconn)
- Kod zakończenia
- Kod przyczyny

Uchwyt połączenia jest używany podczas kolejnych wywołań MQI.

Opis wszystkich parametrów komendy MQCONNX jest podany w tabeli MQCONNX. Pole *Options* umożliwia ustawienie wartości STANDARD_BINDING, FASTPATH_BINDING, SHARED_BINDING lub ISOLATED_BINDING dla dowolnej wersji MQCNO. Za pomocą wywołania MQCONNX można również udostępnić współużytkowane (niezależne od wątku) połączenia. Więcej informacji na ten temat zawiera sekcja "Połączenia współużytkowane (niezależne od wątku) z produktem MQCONNX" na stronie 217 .

MQCNO_STANDARD_BINDING

Domyślnie MQCONNX (np. MQCONN) oznacza dwa wątki logiczne, w których aplikacja WebSphere MQ i agent lokalnego menedżera kolejek są uruchamiane w oddzielnych procesach. Aplikacja WebSphere MQ żąda operacji WebSphere MQ , a lokalny agent menedżera kolejek jest żądał żądania. Wartość ta jest definiowana przez opcję MQCNO_STANDARD_BINDING w wywołaniu MQCONNX.

Jeśli zostanie określona wartość MQCNO_STANDARD_BINDING, wywołanie MQCONNX użyje wartości MQCNO_SHARED_BINDING lub MQCNO_ISOLATED_BINDING, w zależności od wartości atrybutu DefaultBindatrybutu menedżera kolejek zdefiniowanego w pliku qm.ini lub w rejestrze systemu Windows .

Jest to wartość domyślna.

W przypadku łączenia z biblioteką `mqm` najpierw podejmowana jest próba połączenia standardowego z serwerem przy użyciu domyślnego typu powiązania. Jeśli nie powiodła się próba załadowania bazowej biblioteki serwera, zostanie podjęta próba nawiązania połączenia z klientem.

- Jeśli określono zmienną środowiskową MQ_CONNECT_TYPE, można podać jedną z następujących opcji, aby zmienić zachowanie wartości MQCONN lub MQCONNX, jeśli określono wartość MQCNO_STANDARD_BINDING. (Wyjątkiem jest to, że parametr MQCNO_FASTPATH_BINDING został określony z parametrem MQ_CONNECT_TYPE ustawionym na wartość LOCAL lub STANDARD , aby zezwolić na obniżenie wartości połączeń fastpath przez administratora bez powiązanej zmiany z aplikacją:

[1] Jeśli używane są aplikacje wielowątkowe z produktem IBM WebSphere MQ w systemach UNIX and Linux , należy upewnić się, że aplikacje mają wystarczającą wielkość stosu dla wątków. Należy rozważyć użycie stosu o wielkości 256 kB lub większej, gdy aplikacje wielowątkowe wykonują wywołania MQI samodzielnie lub z innymi procedurami obsługi sygnału (na przykład CICS).

Wartość	Znaczenie
KLIENT	Próba nawiązania połączenia z klientem jest wykonywana tylko przez klienta.
Krótka ścieżka	Ta wartość była obsługiwana w poprzednich wersjach, ale została zignorowana, jeśli została określona.
LOKALNA	Próba nawiązania połączenia z serwerem jest wykonywana tylko przez serwer. Połączenia krótkiej ścieżki są obniżane do standardowego połączenia z serwerem.
STANDARDOWA	Obsługiwane w celu zapewnienia zgodności z poprzednimi wersjami. Ta wartość jest teraz traktowana jako LOCAL.

- Jeśli zmienna środowiskowa MQ_CONNECT_TYPE nie jest ustawiona, gdy wywoływana jest wartość MQCONN, podejmowana jest próba połączenia standardowego z serwerem przy użyciu domyślnego typu powiązania. Jeśli ładowanie biblioteki serwera nie powiedzie się, zostanie podjęta próba nawiązania połączenia z klientem.

MQCNO_FASTPATH_BINDING

Zaufane aplikacje oznacza, że aplikacja WebSphere MQ i agent lokalnego menedżera kolejek stają się tym samym procesem. Ponieważ proces agenta nie musi już korzystać z interfejsu w celu uzyskania dostępu do menedżera kolejek, te aplikacje stają się rozszerzeniem menedżera kolejek. Wartość ta jest definiowana przez opcję MQCNO_FASTPATH_BINDING w wywołaniu MQCONNX.

Należy połączyć zaufane aplikacje z wątkami bibliotek produktu WebSphere MQ . Instrukcje na temat sposobu konfigurowania aplikacji WebSphere MQ do uruchamiania jako zaufane zawiera sekcja Opcje MQCNO.

Ta opcja zapewnia najwyższą wydajność.

Uwaga: Ta opcja zapewnia zachowanie integralności menedżera kolejek: nie ma żadnej ochrony przed nadpisaniem jej pamięci masowej. Dotyczy to również sytuacji, gdy aplikacja zawiera błędy, które mogą być ujawnione dla komunikatów i innych danych w menedżerze kolejek. Przed skorzystaniem z tej opcji należy rozważyć następujące kwestie.

MQCNO_SHARED_BINDING

Tę opcję należy określić, aby aplikacja i agent lokalnego menedżera kolejek były uruchamiane w oddzielnych procesach. Zapewnia to zachowanie integralności menedżera kolejek, czyli chroni menedżer kolejek przed programami błędnymi. Jednak aplikacja i agent lokalny-menedżer kolejek współużytkują niektóre zasoby.

Ta opcja jest pośrednia między powiązaniem MQCNO_FASTPATH_BINDING i MQCNO_ISOLATED_BINDING, zarówno jeśli chodzi o ochronę integralności menedżera kolejek, jak i w zakresie wydajności wywołań MQI.

Wartość MQCNO_SHARED_BINDING jest ignorowana, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

Jeśli aplikacja nawiązała połączenie z lokalnym menedżerem kolejek za pomocą komendy MQCNO_SHARED_BINDING, menedżer kolejek może zostać zatrzymany, gdy aplikacja jest uruchomiona. Jeśli menedżer kolejek zostanie zrestartowany w czasie, gdy aplikacja nadal działa, próba uruchomienia menedżera kolejek zakończy się niepowodzeniem z błędem AMQ7018 , ponieważ aplikacja nadal wstrzymuje się do zasobów wymaganych przez menedżer kolejek.

Aby uruchomić menedżer kolejek, należy zatrzymać aplikację.

MQCNO_ISOLATED_BINDING

Tę opcję należy określić, aby aplikacja i agent lokalnego menedżera kolejek były uruchamiane w oddzielnych procesach, co w przypadku parametru MQCNO_SHARED_BINDING. Jednak w tym przypadku proces aplikacji i agent lokalnego menedżera kolejek są odizolowane od siebie, ponieważ nie współużytkują zasobów.

Jest to najbezpieczniejsza opcja zabezpieczania integralności menedżera kolejek, ale zapewnia najmniejszą wydajność wywołań MQI.

Wartość MQCNO_ISOLATED_BINDING jest ignorowana, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

MQCNO_CLIENT_BINDING

Tę opcję należy określić, aby aplikacja próbowała wykonać próbę nawiązania połączenia z klientem. Ta opcja ma następujące ograniczenia:

- Komenda MQCNO_CLIENT_BINDING została odrzucona w systemie z/OS z błędem MQRC_OPTIONS_ERROR.
- Wartość MQCNO_CLIENT_BINDING jest odrzucana za pomocą wywołania MQRC_OPTIONS_ERROR, jeśli jest ona określona z dowolną opcją powiązania MQCNO inną niż MQCNO_STANDARD_BINDING.
- Powiązanie MQCNO_CLIENT_BINDING nie jest dostępne dla produktu Java , ponieważ ma własne mechanizmy wyboru typu powiązania.
- V 7.5.0.7 Przed produktem IBM WebSphere MQ Version 7.5.0, pakiet poprawek 7powiązanie MQCNO_CLIENT_BINDING nie jest dostępne dla platformy .NET, ponieważ ma on własne mechanizmy wyboru typu powiązania. W produkcie Version 7.5.0, Fix Pack 7usuwane jest ograniczenie dotyczące korzystania z platformy .NET dla MQCNO_CLIENT_BINDING.
- Jeśli zmienna środowiskowa MQ_CONNECT_TYPE nie jest ustawiona, gdy wywołano wywołanie MQCONNX, podejmowana jest próba standardowego połączenia z serwerem przy użyciu domyślnego typu powiązania. Jeśli ładowanie biblioteki serwera nie powiedzie się, zostanie podjęta próba nawiązania połączenia z klientem.

MQCNO_LOCAL_BINDING

Tę opcję należy określić, aby aplikacja próbowała nawiązać połączenie z serwerem. Jeśli określono również parametr MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING lub MQCNO_SHARED_BINDING, to połączenie jest typu tego typu i jest udokumentowane w tej sekcji. W przeciwnym razie zostanie podjęta próba użycia standardowego połączenia z serwerem przy użyciu domyślnego typu powiązania. Parametr MQCNO_LOCAL_BINDING ma następujące ograniczenia:

- Parametr MQCNO_LOCAL_BINDING jest ignorowany w systemie z/OS.
- Wartość MQCNO_LOCAL_BINDING jest odrzucana za pomocą wywołania MQRC_OPTIONS_ERROR, jeśli jest ona określona z dowolną opcją ponownego połączenia MQCNO, inną niż MQCNO_RECONNECT_AS_DEF.
- Opcja MQCNO_LOCAL_BINDING nie jest dostępna dla produktu Java , ponieważ ma własne mechanizmy wyboru typu powiązania.
- V 7.5.0.7 Przed produktem IBM WebSphere MQ Version 7.5.0, pakiet poprawek 7powiązanie z opcją MQCNO_LOCAL_BINDING nie jest dostępne dla platformy .NET, ponieważ ma własne mechanizmy wyboru typu powiązania. W produkcie Version 7.5.0, Fix Pack 7usuwane jest ograniczenie dotyczące korzystania z platformy .NET dla MQCNO_LOCAL_BINDING.
- Jeśli zmienna środowiskowa MQ_CONNECT_TYPE nie jest ustawiona, gdy wywołano wywołanie MQCONNX, podejmowana jest próba standardowego połączenia z serwerem przy użyciu domyślnego typu powiązania. Jeśli ładowanie biblioteki serwera nie powiedzie się, zostanie podjęta próba nawiązania połączenia z klientem.

W systemie z/OS te opcje są tolerowane, ale wykonywane jest tylko połączenie standardowe. MQCNO, wersja 3, dla systemu z/OS, umożliwia dostęp do czterech alternatywnych opcji:

MQCNO_SERIALIZE_CONN_TAG_QSG

Umożliwia to aplikacji żądanie, aby w grupie współużytkowania kolejki tylko jedna instancja aplikacji została uruchomiona w dowolnym momencie. Jest to osiągane przez zarejestrowanie użycia znacznika połączenia z wartością, która została określona lub wyprowadzona przez aplikację. Znacznik to 128-bajtowy łańcuch znaków określony w MQCNO w wersji 3.

MQCNO_RESTRICT_CONN_TAG_QSG

Ta opcja jest używana w przypadku, gdy aplikacja składa się z więcej niż jednego procesu (lub TCB), z których każdy może połączyć się z menedżerem kolejek. Połączenie jest dozwolone tylko wtedy, gdy nie ma bieżącego użycia znacznika lub aplikacja żądająca znajduje się w tym samym zasięgu przetwarzania. Jest to przestrzeń adresowa MVS w obrębie tej samej grupy współużytkowania kolejki, co właściciel znacznika.

MQCNO_SERIALIZE_CONN_TAG_Q_MGR

Jest to podobne do wywołania MQCNO_SERIALIZE_CONN_TAG_QSG, ale tylko lokalny menedżer kolejek jest interrogowany, aby sprawdzić, czy żądany znacznik jest już używany.

MQCNO_RESTRICT_CONN_TAG_Q_MGR

Jest to podobne do wartości MQCNO_RESTRICT_CONN_TAG_QSG, ale tylko lokalny menedżer kolejek jest interrogowany, aby sprawdzić, czy żądany znacznik jest już używany.

Ograniczenia dotyczące zaufanych aplikacji

Do zaufanych aplikacji mają zastosowanie następujące ograniczenia:

- Należy jawnie odłączyć zaufane aplikacje od menedżera kolejek.
- Przed zakończeniem działania menedżera kolejek za pomocą komendy endmqm należy zatrzymać zaufane aplikacje.
- Nie wolno używać sygnałów asynchronicznych i przerwań zegara (takich jak `sigkill`) z opcją MQCNO_FASTPATH_BINDING.
- Na wszystkich platformach wątek w zaufanej aplikacji nie może połączyć się z menedżerem kolejek, gdy inny wątek w tym samym procesie jest połączony z innym menedżerem kolejek.
- W produkcie WebSphere MQ w systemach UNIX and Linux należy używać mqm jako efektywnego userID i groupID dla wszystkich wywołań MQI. Te identyfikatory można zmienić przed wywołaniem uwierzytelniania innego niż MQI wymagającym uwierzytelniania (na przykład otwarcie pliku), ale *należy* zmienić go z powrotem na mqm przed dokonaniem następnego wywołania MQI.
- W produkcie WebSphere MQ for HP-UXwielowątkowe aplikacje typu fast-path prawdopodobnie muszą ustawić większą wielkość stosu niż wartość domyślna. Użyj wielkości 256 kB.
- W produkcie WebSphere MQ for Windows zaufane 64-bitowe aplikacje nie są obsługiwane. W przypadku próby uruchomienia zaufanej 64-bitowej aplikacji zostanie ona obniżona do standardowego połączenia powiązanego.
- W produkcie WebSphere MQ w systemach UNIX and Linux zaufane 32-bitowe aplikacje nie są obsługiwane. W przypadku próby uruchomienia zaufanej 32-bitowej aplikacji zostanie ona obniżona do standardowego połączenia powiązanego.

Połączenia współużytkowane (niezależne od wątku) z produktem MQCONNX

Te informacje umożliwiają zapoznanie się ze współużytkowanymi połączeniami z tabelą MQCONNX i uwagami dotyczącymi użycia do rozważenia.

Uwaga: Nieobsługiwane w produkcie WebSphere MQ for z/OS.

W przypadku platform WebSphere MQ innych niż WebSphere MQ for z/OSpołączenie nawiązane z produktem `MQCONN` jest dostępne tylko dla wątku, który nawiąże połączenie. Opcje w wywołaniu MQCONNX pozwalają na utworzenie połączenia, które może być współużytkowane przez wszystkie wątki w procesie. Jeśli aplikacja jest uruchomiona w środowisku transakcyjnym, które wymaga wywołania MQI w tym samym wątku, należy użyć następującej opcji domyślnej:

MQCNO_HANDLE_SHARE_NONE
Tworzy połączenie niewspółużytkowane.

W większości innych środowisk można korzystać z jednego z następujących niezależnych, współużytkowanych opcji połączeń:

MQCNO_HANDLE_SHARE_BLOCK
Tworzy połączenie współużytkowane. W przypadku połączenia z serwerem MQCNO_HANDLE_SHARE_BLOCK , jeśli połączenie jest obecnie używane przez wywołanie MQI w innym wątku, wywołanie MQI oczekuje do czasu zakończenia bieżącego wywołania MQI.

MQCNO_HANDLE_SHARE_NO_BLOCK
Tworzy połączenie współużytkowane. W przypadku połączenia z serwerem MQCNO_HANDLE_SHARE_NO_BLOCK , jeśli połączenie jest obecnie używane przez wywołanie MQI w innym wątku, wywołanie MQI nie powiedzie się natychmiast z powodu MQRC_CALL_IN_PROGRESS.

Z wyjątkiem środowiska MTS (Microsoft Transaction Server), wartością domyślną jest MQCNO_HANDLE_SHARE_NONE. W środowisku MTS wartością domyślną jest MQCNO_HANDLE_SHARE_BLOCK.

Z wywołania MQCONNX zwracany jest uchwyt połączenia. Uchwyt może być używany przez kolejne wywołania MQI z dowolnego wątku w procesie, tworząc powiązanie tych wywołań z uchwytem zwróconego z serwera MQCONNX. Wywołania MQI używające pojedynczego współużytkowanego uchwytu są serializowane między wątkami.

Na przykład następująca sekwencja działań jest możliwa przy użyciu uchwytu współużytkowanego:

1. Wątek 1 wydaje MQCONNX i pobiera współużytkowany uchwyt *h1*
2. Wątek 1 otwiera kolejkę i wysyła żądanie pobrania za pomocą *h1*
3. Wątek 2 wysyła żądanie umieszczenia przy użyciu *h1*
4. Wątek 3 wysyła żądanie umieszczenia przy użyciu *h1*
5. Problemy z wątkiem 2 MQDISC przy użyciu *h1*

Mimo że uchwyt jest używany przez dowolny wątek, dostęp do połączenia jest niedostępny dla innych wątków. W sytuacji, gdy dopuszczalne jest, że wątek oczekuje na zakończenie dowolnego poprzedniego wywołania z innego wątku, należy użyć opcji MQCONNX z opcją MQCNO_HANDLE_SHARE_BLOCK.

Jednak blokowanie może powodować trudności. Przypuśćmy, że w kroku “2” na stronie 218wątek 1 wysyła żądanie pobrania, które oczekuje na komunikaty, które mogły nie zostać jeszcze odebrane (a get with wait). W tym przypadku wątki 2 i 3 również pozostawiane są na czas oczekiwania (zablokowane) tak długo, jak długo trwa żądanie pobrania wątku 1. Jeśli wolisz, aby wywołanie MQI zostało zwrócone z błędem, jeśli na tym uchwycie jest już uruchomione inne wywołanie MQI, użyj opcji MQCONNX z opcją MQCNO_HANDLE_SHARE_NO_BLOCK.

Uwagi dotyczące użycia współużytkowanego połączenia

1. Wszystkie uchwyty obiektów (Hobj) utworzone przez otwarcie obiektu są powiązane z Hconn; tak dla wspólnego Hconn, Hobjs są również współużytkowane i możliwe do wykorzystania przez dowolny wątek z użyciem Hconn. Podobnie każda jednostka pracy rozpoczęta pod Hconn jest powiązana z tym Hconn, więc ta również jest współużytkowana przez wątki ze współużytkaną Hconn.
2. *Dowolny* wątek może wywołać komendę MQDISC, aby odłączyć współużytkowane Hconn, a nie tylko wątek, który wywołał odpowiednią tabelę MQCONNX. Zmaterializowana tabela MQDISC kończy działanie programu Hconn, które nie jest dostępne dla wszystkich wątków.
3. Pojedynczy wątek może używać wielu współużytkowanych zasobów Hconns szeregowo, na przykład za pomocą komendy MQPUT można umieścić jeden komunikat pod jednym współużytkowanym Hconn, a następnie umieścić kolejny komunikat przy użyciu innego współużytkowanego Hconn, przy czym każda operacja jest wykonywana w innej lokalnej jednostce pracy.
4. Współużytkowane połączenia Hconns nie mogą być używane w ramach globalnej jednostki pracy.

Użycie opcji wywołania MQCONNX z wartością MQ_CONNECT_TYPE

Te informacje umożliwiają zrozumienie różnych opcji wywołania MQCONNX, w jaki sposób są one używane z typem MQ_CONNECT_TYPE.

W systemach WebSphere MQ for IBM i, WebSphere MQ for Windowsoraz WebSphere MQ w systemach UNIX and Linux można użyć zmiennej środowiskowej MQ_CONNECT_TYPE w połączeniu z typem powiązania określonym w polu *Options* struktury MQCNO używanej w wywołaniu MQCONNX.

Tabela 33. Zmienna środowiskowa MQ_CONNECT_TYPE

Opcja wywołania MQCONNX	**MQ_CONNECT_TYPE, zmienna środowiskowa**	**Wynik**
STANDARDOWA	UNDEFINED	STANDARDOWA
STANDARDOWA	STANDARDOWA	STANDARDOWA
STANDARDOWA	Krótka ścieżka	STANDARDOWA
STANDARDOWA	KLIENT	KLIENT
STANDARDOWA	LOKALNA	STANDARDOWA

Jeśli opcja MQCNO_STANDARD_BINDING nie jest określona, można użyć opcji MQCNO_NONE, która domyślnie ma wartość MQCNO_STANDARD_BINDING.

Rozłączanie programów z menedżera kolejek za pomocą MQDISC

Ta sekcja zawiera informacje na temat odłączania programów z menedżera kolejek za pomocą komendy MQDISC.

Gdy program, który nawiąże połączenie z menedżerem kolejek przy użyciu wywołania MQCONN lub MQCONNX, zakończy wszystkie interakcje z menedżerem kolejek, przerywa on połączenie za pomocą wywołania MQDISC, z wyjątkiem:

- W przypadku aplikacji CICS Transaction Server for z/OS , gdzie wywołanie jest opcjonalne, chyba że użyto komendy MQCONNX, a użytkownik chce usunąć znacznik połączenia przed zakończeniem aplikacji.
- W produkcie WebSphere MQ for IBM i , w którym po wylogowaniu się z systemu operacyjnego zostanie wykonane niejawne wywołanie MQDISC.

Jako dane wejściowe wywołania MQDISC należy podać uchwyt połączenia (Hconn), który został zwrócony przez komendę MQCONN lub MQCONNX po nawiązaniu połączenia z menedżerem kolejek.

Z wyjątkiem programu CICS w systemie z/OS, po wywołaniu komendy MQDISC uchwyt połączenia (Hconn) nie jest już poprawny i nie można wywoływać kolejnych wywołań MQI, dopóki nie zostanie ponownie wywołana funkcja MQCONN lub MQCONNX. Komenda MQDISC wykonuje niejawną operację MQCLOSE dla wszystkich obiektów, które nadal są otwarte za pomocą tego uchwytu.

Jeśli produkt MQCONNX jest używany do nawiązywania połączenia w produkcie WebSphere MQ for z/OS, produkt MQDISC kończy również zasięg znacznika połączenia utworzonego przez produkt MQCONNX. Jednak w przypadku aplikacji CICS, IMSlub RRS, jeśli istnieje aktywna jednostka odtwarzania powiązana ze znacznikiem połączenia, wartość MQDISC jest odrzucana z kodem przyczyny MQRC_CONN_TAG_NOT_RELEASED.

Opisy parametrów są podane w opisie wywołania MQDISC w MQDISC.

Jeśli nie zostanie wydana żadna wartość MQDISC

Standardowe, niewspółużytkowane połączenie (Hconn) jest czyszczone podczas końców tworzenia wątku. Połączenie współużytkowane jest tylko niejawnie wycofane i rozłączone, gdy cały proces kończy działanie. Jeśli wątek, który utworzył współużytkowany Hconn, kończy się, a Hconn nadal istnieje, to Hconn jest nadal użyteczny.

Sprawdzanie uprawnień

Wywołania MQCLOSE i MQDISC zwykle nie sprawdzają uprawnień.

W normalnym toku zdarzeń zadanie, które ma uprawnienie do otwierania lub nawiązywania połączenia z obiektem WebSphere MQ , zamyka się lub rozłącza się z tym obiektem. Nawet jeśli uprawnienie zadania, które nawiąże połączenie z obiektem WebSphere MQ lub go otworzyło, jest odwołane, wywołania MQCLOSE i MQDISC są akceptowane.

Otwieranie i zamykanie obiektów

Te informacje udostępniają wgląd w otwieranie i zamykanie obiektów WebSphere MQ .

Aby wykonać którekolwiek z poniższych operacji, należy najpierw *otworzyć* odpowiedni obiekt WebSphere MQ :

- Umieszczanie komunikatów w kolejce
- Pobieranie (przeglądanie lub pobieranie) komunikatów z kolejki
- Ustawianie atrybutów obiektu
- Zapytaj o atrybuty dowolnego obiektu

Użyj wywołania MQOPEN, aby otworzyć obiekt, korzystając z opcji wywołania w celu określenia, co ma być zrobione z obiektem. Jedynym wyjątkiem jest umieszczenie pojedynczego komunikatu w kolejce, a następnie natychmiastowe zamknięcie kolejki. W takim przypadku można pominąć etap *otwierania* za pomocą wywołania MQPUT1 (patrz “Umieszczanie jednego komunikatu w kolejce przy użyciu wywołania MQPUT1” na stronie 240).

Przed otwarniem obiektu za pomocą wywołania MQOPEN należy połączyć program z menedżerem kolejek. Jest to szczegółowo wyjaśnione w przypadku wszystkich środowisk w produkcie “Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego” na stronie 211.

Istnieją cztery typy obiektów produktu WebSphere MQ , które można otworzyć:

- Kolejka
- Lista nazw
- Definicja procesu
- Menedżer kolejek

Wszystkie te obiekty można otworzyć w podobny sposób, korzystając z wywołania MQOPEN. Więcej informacji na temat obiektów produktu WebSphere MQ zawiera sekcja Obiekty.

Ten sam obiekt można otworzyć więcej niż raz, a za każdym razem można uzyskać nowy uchwyt obiektu.Użytkownik może chcieć przeglądać komunikaty w kolejce za pomocą jednego uchwytu i usunąć komunikaty z tej samej kolejki za pomocą innego uchwytu. Umożliwia to składowanie przy użyciu zasobów do zamknięcia i ponownego otwarcia tego samego obiektu. Istnieje również możliwość otwarcia kolejki w celu przeglądania *i* usuwania komunikatów w tym samym czasie.

Ponadto istnieje możliwość otwarcia wielu obiektów za pomocą jednej operacji MQOPEN i zamknięcia ich za pomocą komendy MQCLOSE. Informacje o tym, jak to zrobić, zawiera sekcja “Lista dystrybucyjna” na stronie 241 .

Podczas próby otwarcia obiektu menedżer kolejek sprawdza, czy użytkownik jest uprawniony do otwarcia tego obiektu dla opcji określonych w wywołaniu MQOPEN.

Obiekty są zamykane automatycznie, gdy program rozłącza się z menedżerem kolejek. W środowisku IMS odłączenie jest wymuszane, gdy program rozpocznie przetwarzanie dla nowego użytkownika po wywołaniu jednostki GU (pobierz unikalne) IMS . Na platformie IBM i obiekty są zamykane automatycznie po zakończeniu zadania.

Dobrą praktyką programowania jest zamykanie otwartych obiektów. Aby to zrobić, należy użyć wywołania MQCLOSE.

Aby dowiedzieć się więcej na temat otwierania i zamykania obiektów, należy użyć następujących odsyłaczy:

- “Otwieranie obiektów za pomocą wywołania MQOPEN” na stronie 221
- “Tworzenie kolejek dynamicznych” na stronie 229
- “Otwieranie kolejek zdalnych” na stronie 230
- “Zamykanie obiektów przy użyciu wywołania MQCLOSE” na stronie 230

Pojęcia pokrewne

“Interfejs kolejki komunikatów-przegląd” na stronie 199
Informacje na temat komponentów interfejsu kolejek komunikatów (Message Queue Interface-MQI).

“Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego” na stronie 211
Aby można było korzystać z usług programistycznych produktu WebSphere MQ , program musi mieć połączenie z menedżerem kolejek. Informacje zawarte w tej sekcji umożliwiają zapoznanie się z informacjami na temat nawiązywania połączenia z menedżerem kolejek i z jego rozłączeniem.

“Umieszczanie komunikatów w kolejce” na stronie 231
Te informacje umożliwiają zapoznanie się z informacjami na temat umieszczania komunikatów w kolejce.

“Pobieranie komunikatów z kolejki” na stronie 246
Ta sekcja zawiera informacje na temat pobierania komunikatów z kolejki.

“Sprawdzanie i ustawianie atrybutów obiektu” na stronie 328
Atrybuty to właściwości, które definiują parametry obiektu WebSphere MQ .

“Zatwierdzanie i wycofywanie jednostek pracy” na stronie 331
W tej sekcji opisano sposób zatwierdzania i tworzenia kopii zapasowych wszystkich możliwych do odtworzenia operacji get i put, które wystąpiły w jednostce pracy.

“Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy” na stronie 338
Informacje na temat wyzwalaczy i sposobów uruchamiania aplikacji IBM WebSphere MQ przy użyciu wyzwalaczy.

“Praca z interfejsem MQI i klastrami” na stronie 355
Istnieją specjalne opcje dotyczące wywołań i kodów powrotu, które odnoszą się do technologii klastrowej.

Otwieranie obiektów za pomocą wywołania MQOPEN

Te informacje umożliwiają zapoznanie się z otwarciem obiektów za pomocą wywołania MQOPEN.

Jako dane wejściowe dla wywołania MQOPEN należy podać:

- Uchwyt połączenia. W przypadku aplikacji CICS w systemie z/OSmożna określić stałą MQHC_DEF_HCONN (która ma wartość zero), lub użyć uchwytu połączenia zwróconego przez wywołanie MQCONN lub MQCONNX. W przypadku innych programów zawsze należy używać uchwytu połączenia zwróconego przez wywołanie MQCONN lub MQCONNX.
- Opis obiektu, który ma zostać otwarty, przy użyciu struktury deskryptora obiektu (MQOD).
- Jedna lub więcej opcji, które sterują działaniem wywołania.

Dane wyjściowe komendy MQOPEN są następujące:

- Uchwyt obiektu, który reprezentuje dostęp do obiektu. Użyj tego parametru na wejściu do wszystkich kolejnych wywołań MQI.
- Zmodyfikowaną strukturę deskryptora obiektu, jeśli tworzona jest kolejka dynamiczna (i jest ona obsługiwana na używanej platformie).
- Kod zakończenia.
- Kod przyczyny.

Zasięg uchwytu obiektu

Zakres uchwytu obiektu (Hobj) jest taki sam, jak zasięg uchwytu połączenia (Hconn).

Jest to uwzględnione w “Zasięg MQCONN lub MQCONNX” na stronie 213 i “Połączenia współużytkowane (niezależne od wątku) z produktem MQCONNX” na stronie 217. W niektórych środowiskach istnieją jednak dodatkowe uwagi:

Program CICS
W programie CICS można używać uchwytu tylko w ramach tego samego zadania CICS , z którego nawiązałeś wywołanie MQOPEN.

Zadania wsadowe IMS i z/OS
W środowiskach IMS i wsadowych można używać uchwytu w ramach tego samego zadania, ale nie w ramach podzadań.

Opisy parametrów wywołania MQOPEN są podane w sekcji MQOPEN.

W poniższych sekcjach opisano informacje, które należy podać jako dane wejściowe dla komendy MQOPEN.

Identyfikowanie obiektów (struktura MQOD)

Użyj struktury MQOD, aby zidentyfikować obiekt, który ma zostać otwarty. Ta struktura jest parametrem wejściowym wywołania MQOPEN. (Struktura jest modyfikowana przez menedżer kolejek w przypadku użycia wywołania MQOPEN w celu utworzenia kolejki dynamicznej).

Szczegółowe informacje na temat struktury MQOD można znaleźć w sekcji MQOD.

Informacje na temat korzystania ze struktury MQOD dla list dystrybucyjnych zawiera sekcja “Korzystanie ze struktury MQOD” na stronie 242 w sekcji “Lista dystrybucyjna” na stronie 241.

Rozdzielczość nazwy

W jaki sposób wywołanie MQOPEN rozwiązuje nazwy kolejek i menedżerów kolejek.

Uwaga: Alias menedżera kolejek jest definicją kolejki zdalnej bez pola RNAME .

Po otwarciu kolejki produktu WebSphere MQ wywołanie MQOPEN wykonuje funkcję tłumaczenia nazw w podanej nazwie kolejki. Określa, w której kolejce menedżer kolejek wykonuje kolejne operacje. Oznacza to, że po określeniu nazwy kolejki aliasowej lub kolejki zdalnej w deskryptorze obiektu (MQOD) wywołanie jest tłumaczone na nazwę kolejki lokalnej lub kolejki transmisji. Jeśli kolejka jest otwierana dla dowolnego typu danych wejściowych, przeglądania lub ustawiania, jest ona tłumaczona na kolejkę lokalną, jeśli istnieje, i nie powiedzie się, jeśli nie ma ona jednego. Jest ona tłumaczona na kolejkę nielokalną tylko wtedy, gdy jest otwarta tylko dla danych wyjściowych, tylko do zapytania, lub tylko do wyjścia i zapytania. Więcej informacji na temat procesu rozstrzygania nazw zawiera sekcja Tabela 34 na stronie 223 . Nazwa dostarczona w produkcie *ObjectQMgrName* jest tłumaczana *przed* tym, że w *ObjectName*.

Tabela 34 na stronie 223 pokazuje również, w jaki sposób można użyć lokalnej definicji kolejki zdalnej w celu zdefiniowania aliasu dla nazwy menedżera kolejek. Pozwala to wybrać, która kolejka transmisji jest używana podczas umieszczania komunikatów w kolejce zdalnej, dzięki czemu można na przykład użyć pojedynczej kolejki transmisji dla komunikatów przeznaczonych dla wielu menedżerów kolejek zdalnych.

Aby użyć poniższej tabeli, należy najpierw odczytać dwie kolumny z lewej strony, pod nagłówkiem **Dane wejściowe do tabeli MQOD**, a następnie wybrać odpowiednią wielkość liter. Następnie należy przeczytać odpowiadający mu wiersz, postępując zgodnie z instrukcjami. Postępując zgodnie z instrukcjami w kolumnach **Rozstrzygnięte nazwy** , można powrócić do kolumn **Wejście do tabeli MQOD** i wstawić wartości zgodnie z instrukcjami, albo można wyjść z tabeli z dostarczonymi wynikami. Na przykład może być wymagane wprowadzenie danych wejściowych *ObjectName*.

Tabela 34. Rozstrzyganie nazw kolejek podczas korzystania z komendy MQOPEN

Dane wejściowe dla MQOD		**Rozwiązane nazwy**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Kolejka transmisji**
Pusty lub lokalny menedżer kolejek	Kolejka lokalna bez atrybutu CLUSTER	Lokalny menedżer kolejek	Wejście *ObjectName*	Nie dotyczy (używana kolejka lokalna)
Pusty menedżer kolejek	Kolejka lokalna z atrybutem CLUSTER	Wybrany menedżer kolejek klastra lub wybrany menedżer kolejek klastra wybrany na PUT zarządzania obciążeniem	Wejście *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE i użyta kolejka lokalna SYSTEM.QSG.TRANSMIT.QUEUE (patrz uwaga)
Lokalny menedżer kolejek	Kolejka lokalna z atrybutem CLUSTER	Lokalny menedżer kolejek	Wejście *ObjectName*	Nie dotyczy (używana kolejka lokalna)
Pusty lub lokalny menedżer kolejek	Kolejka modelowa	Lokalny menedżer kolejek	Nazwa generowana	Nie dotyczy (używana kolejka lokalna)
Pusty lub lokalny menedżer kolejek	Kolejka aliasowa z atrybutem CLUSTER lub bez niej	Ponownie wykonaj tłumaczenie nazw z nazwą *ObjectQMgrName* , a w obiekcie definicji kolejki aliasowej wpisz *ObjectName* ustawioną na wartość *BaseQName* . Nie może być tłumaczany na alias lokalnie zdefiniowany, jeśli określony jest parametr *ObjectQMgrName* , ale może on być tłumaczany na alias klastrowy (udostępniany w innych menedżerach kolejek), w którym wartość *ObjectQMgrName* jest pusta.		
Lokalny menedżer kolejek	Kolejka aliasowa z atrybutem CLUSTER	Alias nie może być tłumaczany na kolejkę klastra, która nie jest zdefiniowana lokalnie, lub nie może być kolejką klastra, która ma taką samą wartość *ObjectName* , co alias.		

Tabela 34. Rozstrzyganie nazw kolejek podczas korzystania z komendy MQOPEN (kontynuacja)

Dane wejściowe dla MQOD		**Rozwiązane nazwy**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Kolejka transmisji**
Pusty menedżer kolejek	Kolejka aliasowa z atrybutem CLUSTER	Alias może zostać rozstrzygany do kolejki klastra o takiej samej nazwie *ObjectName* , co alias.		
Pusty lub lokalny menedżer kolejek	lokalna definicja kolejki zdalnej	Ponownie wykonaj tłumaczenie nazw z parametrem *ObjectQMgrName* ustawionym na wartość *RemoteQMgrName*, a *ObjectName* ustaw na wartość *RemoteQName*. Nie można rozstrzygnąć kolejek zdalnych		Nazwa atrybutu *XmitQName* , jeśli nazwa inna niż blank; w przeciwnym razie *RemoteQMgrName* znajduje się w obiekcie definicji kolejki zdalnej. SYSTEM.QSG.TRANSMIT.QUEUE (patrz uwaga)
Pusty menedżer kolejek	Nie znaleziono zgodnego obiektu lokalnego; znaleziono kolejki klastra	Wybrany menedżer kolejek klastra lub wybrany menedżer kolejek klastra wybrany na PUT zarządzania obciążeniem	Wejście *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (patrz uwaga)
Pusty lub lokalny menedżer kolejek	Brak zgodnego obiektu lokalnego; nie znaleziono kolejki klastra		Błąd, kolejka nie została znaleziona	Nie dotyczy
Nazwa menedżera kolejek w tej samej grupie współużytkowania kolejki, co lokalny menedżer kolejek	Lokalna kolejka współużytkowana	Lokalny menedżer kolejek	Wejście *ObjectName*	Nie dotyczy
Nazwa lokalnej kolejki transmisji	(Nierozstrzygnięte)	Dane wejściowe *ObjectQMgrName*	Wejście *ObjectName*	Dane wejściowe *ObjectQMgrName* SYSTEM.QSG.TRANSMIT.QUEUE (patrz uwaga)
Definicja aliasu menedżera kolejek (*RemoteQMgrName* może być nazwą lokalnego menedżera kolejek)	(Nie rozstrzygnięto, kolejka zdalna)	Ponownie wykonaj tłumaczenie nazw z nazwą *ObjectQMgrName* ustawioną na *RemoteQMgrName*. Nie może być rozstrzygane do kolejek zdalnych	Wejście *ObjectName*	Nazwa atrybutu *XmitQName* , jeśli nazwa inna niż blank; w przeciwnym razie *RemoteQMgrName* znajduje się w obiekcie definicji kolejki zdalnej. SYSTEM.QSG.TRANSMIT.QUEUE (patrz uwaga)

Tabela 34. Rozstrzyganie nazw kolejek podczas korzystania z komendy MQOPEN (kontynuacja)				
Dane wejściowe dla MQOD		**Rozwiązane nazwy**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Kolejka transmisji**
Menedżer kolejek nie jest nazwą żadnego obiektu lokalnego; znaleziono menedżery kolejek klastra lub alias menedżera kolejek	(Nierozstrzygnięte)	*ObjectQMgrNazwa* lub konkretny menedżer kolejek klastra wybrany w operacji PUT	Wejście *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (patrz uwaga)
Menedżer kolejek nie jest nazwą żadnego obiektu lokalnego; nie znaleziono obiektów klastra	(Nierozstrzygnięte)	Dane wejściowe *ObjectQMgrName*	Wejście *ObjectName*	Atrybut *DefXmitQName* menedżera kolejek, w którym obsługiwany jest program *DefXmitQName* . SYSTEM.QSG.TRANSMIT.QUEUE (patrz uwaga)

Uwagi:

1. *BaseQName* to nazwa kolejki podstawowej z definicji kolejki aliasowej.
2. *RemoteQName* to nazwa kolejki zdalnej z lokalnej definicji kolejki zdalnej.
3. *RemoteQMgrName* to nazwa zdalnego menedżera kolejek z lokalnej definicji kolejki zdalnej.
4. *XmitQName* to nazwa kolejki transmisji z lokalnej definicji kolejki zdalnej.
5. W przypadku używania menedżerów kolejek produktu WebSphere MQ for z/OS , które są częścią grupy współużytkowania kolejek (QSG), zamiast nazwy lokalnego menedżera kolejek w programie Tabela 34 na stronie 223można użyć nazwy QSG.

   Jeśli lokalny menedżer kolejek nie może otworzyć kolejki docelowej lub umieścić komunikat w kolejce, komunikat jest przesyłany do określonej nazwy ObjectQMgr(w kolejkach wewnątrzgrupowych lub w kanale WebSphere MQ).
6. W kolumnie *ObjectName* w tabeli CLUSTER odnosi się zarówno do atrybutów CLUSTER, jak i CLUSNL kolejki.
7. SYSTEM.QSG.TRANSMIT.QUEUE jest używana, jeśli lokalne i zdalne menedżery kolejek znajdują się w tej samej grupie współużytkowania kolejki; kolejkowanie wewnątrz grupy jest włączone.
8. Jeśli do każdego kanału nadawczego klastra przypisana została inna kolejka transmisji klastra, SYSTEM.CLUSTER.TRANSMIT.QUEUE może nie być nazwą kolejki transmisji klastra. Więcej informacji na temat wielu kolejek transmisji klastra zawiera sekcja Łączenie w klastry: planowanie konfigurowania kolejek transmisji klastra .
9. W sytuacji, w której menedżer kolejek nie jest nazwą żadnego obiektu lokalnego, znaleziono menedżery kolejek klastra lub alias menedżera kolejek.

   Jeśli nazwa menedżera kolejek została podana przy użyciu produktu **ObjectQMgrName**, a istnieje wiele kanałów klastra o różnych nazwach klastrów znanych przez lokalny menedżer kolejek, które mogą osiągnąć ten cel, to dowolny z tych kanałów może zostać użyty do przeniesienia komunikatu, niezależnie od nazwy klastra kolejki docelowej.

   Może to być nieoczekiwane, jeśli komunikaty dla tej kolejki były przewidywane tylko w celu wysłania przez kanał o takiej samej nazwie klastra co kolejka.

   Jednak **ObjectQMgrName** ma pierwszeństwo w tym przypadku, a równoważenie obciążenia klastra uwzględnia wszystkie kanały, które mogą osiągnąć ten menedżer kolejek, niezależnie od nazwy klastra, w którym się znajdują.

Otwarcie kolejki aliasowej powoduje również otwarcie kolejki podstawowej, do której alias jest tłumaczący, a otwarcie kolejki zdalnej powoduje otwarcie kolejki transmisji. Z tego powodu nie można usunąć ani określonej kolejki, ani kolejki, do której jest ona tłumaczona, gdy druga jest otwarta.

Podczas gdy kolejka aliasowa nie może być przetłumaczana na inną lokalnie zdefiniowaną kolejkę aliasową (współużytkowana w klastrze lub nie), rozstrzyganie do zdalnej zdefiniowanej kolejki aliasowej klastra jest dozwolone i dlatego może zostać określone jako kolejka podstawowa.

Rozstrzygnięta nazwa kolejki i rozstrzygnięta nazwa menedżera kolejek są przechowywane w polach *ResolvedQName* i *ResolvedQMgrName* w MQOD.

Więcej informacji na temat rozstrzygania nazw w rozproszonym środowisku kolejkowania zawiera sekcja Co to jest rozdzielczość nazw kolejek?.

Korzystanie z opcji wywołania MQOPEN

W parametrze *Options* wywołania MQOPEN należy wybrać jedną lub więcej opcji, aby kontrolować dostęp nadawany obiektowi, który jest otwierany. Za pomocą tych opcji można:

- Otwórz kolejkę i określ, że wszystkie komunikaty umieszczone w tej kolejce muszą być kierowane do tej samej instancji.
- Otwórz kolejkę, aby umożliwić umieszczanie na niej komunikatów.
- Otwórz kolejkę, aby umożliwić przeglądanie komunikatów na niej
- Otwórz kolejkę, aby umożliwić usuwanie z niego komunikatów.
- Otwórz obiekt, aby umożliwić sprawdzenie i ustawienie jego atrybutów (ale można ustawić atrybuty tylko kolejek)
- Otwórz temat lub łańcuch tematu, aby opublikować w nim komunikaty.
- Wiązanie informacji kontekstowych z komunikatem
- Nominuj alternatywny identyfikator użytkownika, który ma być używany do sprawdzania zabezpieczeń
- Sterowanie wywołaniem, jeśli menedżer kolejek znajduje się w stanie wygaszania

Opcja MQOPEN dla kolejki klastra
Powiązanie używane dla uchwytu kolejki jest pobierane z atrybutu kolejki *DefBind* , który może mieć wartość MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXEDlub MQBND_BIND_ON_GROUP.

Aby skierować wszystkie komunikaty umieszczone w kolejce przy użyciu funkcji MQPUT do tego samego menedżera kolejek przy użyciu tej samej trasy, należy użyć opcji MQOO_BIND_ON_OPEN w wywołaniu funkcji MQOPEN .

Aby określić, że miejsce docelowe ma zostać wybrane w czasie MQPUT, czyli dla poszczególnych komunikatów, należy użyć opcji MQOO_BIND_NOT_FIXED w wywołaniu MQOPEN .

Aby określić, że wszystkie komunikaty w grupach komunikatów umieszczane w kolejce za pomocą funkcji MQPUT są przydzielane do tej samej instancji docelowej, należy użyć opcji MQOO_BIND_ON_GROUP w wywołaniu funkcji MQOPEN .

W przypadku korzystania z grup komunikatów z klastrami należy określić parametr MQOO_BIND_ON_OPEN lub MQOO_BIND_ON_GROUP , aby zapewnić, że wszystkie komunikaty w grupie będą przetwarzane w tym samym miejscu docelowym.

Jeśli żadna z tych opcji nie zostanie podana, zostanie użyta wartość domyślna MQOO_BIND_AS_Q_DEF.

Jeśli nazwa menedżera kolejek zostanie określona w programie MQOD, wybrana zostanie kolejka w tym menedżerze kolejek. Jeśli nazwa menedżera kolejek jest pusta, można wybrać dowolną instancję. Więcej informacji zawiera sekcja “MQOPEN i klastry” na stronie 356.

Jeśli kolejka klastra jest otwierana za pomocą definicji QALIAS , niektóre atrybuty kolejki są definiowane przez kolejkę aliasową, a nie przez kolejkę podstawową. Atrybuty klastra należą do atrybutów definicji kolejki podstawowej, które są nadpisywane przez kolejkę aliasową. Na przykład w poniższym fragmencie kodu kolejka klastra jest otwierana za pomocą kodu MQOO_BIND_NOT FIXED , a nie za pomocą

kodu MQOO_BIND_ON_OPEN. Definicja kolejki klastra jest ogłaszana w całym klastrze, a definicja kolejki aliasowej jest lokalna względem menedżera kolejek.

```
DEFINE QLOCAL(CLQ1)  CLUSTER(MYCLUSTER) DEFBIND(OPEN) REPLACE
DEFINE QALIAS(ACLQ1) TARGET(CLQ1) DEFBIND(NOTFIXED) REPLACE
```

Opcja MQOPEN dla umieszczania komunikatów
Aby otworzyć kolejkę lub temat w celu umieszczenia na nim komunikatów, należy użyć opcji MQOO_OUTPUT.

Opcja MQOPEN dla przeglądania komunikatów
Aby otworzyć kolejkę, aby możliwe było *przeglądanie* komunikatów na nim, należy użyć wywołania MQOPEN z opcją MQOO_BROWSE.

Spowoduje to utworzenie *kursora przeglądania* używanego przez menedżer kolejek w celu zidentyfikowania następnego komunikatu w kolejce. Więcej informacji na ten temat zawiera sekcja “Przeglądanie komunikatów w kolejce” na stronie 279.

Uwaga:

1. Nie można przeglądać komunikatów w kolejce zdalnej; nie należy otwierać kolejki zdalnej przy użyciu opcji MQOO_BROWSE.
2. Nie można określić tej opcji podczas otwierania listy dystrybucyjnej. Więcej informacji na temat list dystrybucyjnych zawiera sekcja “Lista dystrybucyjna” na stronie 241.
3. Użyj komendy MQOO_CO_OP w połączeniu z opcją MQOO_BROWSE, jeśli korzystasz z przeglądania kooperatywnego. Patrz Opcje .

Opcje MQOPEN służące do usuwania komunikatów
Trzy opcje sterują otwarciem kolejki w celu usunięcia z niej komunikatów.

W dowolnym wywołaniu MQOPEN można używać tylko jednego z nich. Te opcje definiują, czy program ma wyłączny lub współużytkowany dostęp do kolejki. *Wyłączny dostęp* oznacza, że do czasu zamknięcia kolejki,Tylko użytkownik może usunąć z niego komunikaty. Jeśli inny program podejmie próbę otwarcia kolejki w celu usunięcia komunikatów, jego wywołanie MQOPEN nie powiedzie się. *Dostęp współużytkowany* oznacza, że można usunąć więcej niż jeden programkomunikaty z kolejki.

Najbardziej zaleconym podejściem jest zaakceptowanie typu dostępu, który był przeznaczony dla kolejki, gdy kolejka została zdefiniowana. Definicja kolejki zaangażowana jest w ustawienie *Shareability* i*DefInputOpenOption* . Aby zaakceptować ten dostęp, należy skorzystać z opcji MQOO_INPUT_AS_Q_DEF. Informacje o tym, w jaki sposób ustawienie tych atrybutów ma wpływ na typ dostępu, który zostanie podany podczas korzystania z tej opcji, można znaleźć w sekcji Tabela 35 na stronie 227 .

Tabela 35. Sposób, w jaki atrybuty kolejki i opcje wywołania MQOPEN mają wpływ na dostęp do kolejek

Kolejka - atrybuty		**Typ dostępu z opcjami MQOPEN**		
Shareability	*DefInputOpenOption*	**AS_Q_DEF**	**Współużytkowane**	**EXCLUSIVE**
Do współużytkowania	Współużytkowane	shared (współużytkowany)	shared (współużytkowany)	wykluczająca
Do współużytkowania	EXCLUSIVE	wykluczająca	shared (współużytkowany)	wykluczająca
NOT_SHAREABLE*	SHARED*	wykluczająca	wykluczająca	wykluczająca
NOT_SHAREABLE	EXCLUSIVE	wykluczająca	wykluczająca	wykluczająca

Tabela 35. Sposób, w jaki atrybuty kolejki i opcje wywołania MQOPEN mają wpływ na dostęp do kolejek (kontynuacja)	
Kolejka - atrybuty	**Typ dostępu z opcjami MQOPEN**
Uwaga: * Mimo że można zdefiniować kolejkę tak, aby miała ona taką kombinację atrybutów, domyślna opcja otwierania danych wejściowych jest przesłaniana przez atrybut współużytkowalności.	

Lub:

- Jeśli wiadomo, że aplikacja może działać poprawnie, nawet jeśli inne programy mogą usunąć komunikaty z kolejki w tym samym czasie, należy użyć opcji MQOO_INPUT_SHARED. Tabela 35 na stronie 227 pokazuje, w jaki sposób, w niektórych przypadkach, użytkownik otrzyma wyłączny dostęp do kolejki, nawet przy użyciu tej opcji.
- Jeśli wiadomo, że aplikacja może działać poprawnie tylko wtedy, gdy inne programy nie usuwają komunikatów z kolejki w tym samym czasie, należy użyć opcji MQOO_INPUT_EXCLUSIVE.

Uwaga:

1. Nie można usunąć komunikatów z kolejki zdalnej. Z tego powodu nie można otworzyć zdalnej kolejki za pomocą żadnej z opcji MQOO_INPUT_ *.
2. Nie można określić tej opcji podczas otwierania listy dystrybucyjnej. Więcej informacji zawiera sekcja "Lista dystrybucyjna" na stronie 241.

Opcje MQOPEN służące do ustawiania i uzyskiwania informacji o atrybutach
Aby otworzyć kolejkę, tak aby można było ustawić jej atrybuty, należy użyć opcji MQOO_SET.

Nie można ustawić atrybutów żadnego innego typu obiektu (patrz sekcja "Sprawdzanie i ustawianie atrybutów obiektu" na stronie 328).

Aby otworzyć obiekt, aby można było dowiedzieć się o jego atrybutach, należy użyć opcji MQOO_INQUIRE.

Uwaga: Nie można określić tej opcji podczas otwierania listy dystrybucyjnej.

Opcje MQOPEN związane z kontekstem komunikatu
Aby możliwe było powiązanie informacji kontekstowych z komunikatem podczas umieszczania go w kolejce, należy użyć jednej z opcji kontekstu komunikatu podczas otwierania kolejki.

Opcje umożliwiają rozróżnianie informacji o kontekście, które odnoszą się do *użytkownika* , który wygenerował komunikat, a który odnosi się do *aplikacji* , która zainicjował dany komunikat. Ponadto można wybrać opcję ustawienia informacji kontekstowych podczas umieszczania komunikatu w kolejce lub wybrać opcję automatycznego uruchamiania kontekstowego z innego uchwytu kolejki.

Pojęcia pokrewne
"Kontekst komunikatu" na stronie 39
Informacja *Kontekst komunikatu* umożliwia aplikacji, która pobiera komunikat, w celu uzyskania informacji o inicjatorze komunikatu.

"Sterowanie informacjami kontekstową" na stronie 237
W przypadku użycia wywołania MQPUT lub MQPUT1 w celu umieszczenia komunikatu w kolejce można określić, że menedżer kolejek ma dodać pewne domyślne informacje kontekstu do deskryptora komunikatu. Aplikacje, które mają odpowiedni poziom uprawnień, mogą dodawać dodatkowe informacje kontekstowe. Do sterowania informacjami kontekstowych można użyć pola opcji w strukturze MQPMO.

Opcja MQOPEN dla alternatywnego uprawnienia użytkownika
Podczas próby otwarcia obiektu za pomocą wywołania MQOPEN menedżer kolejek sprawdza, czy użytkownik ma uprawnienia do otwierania tego obiektu. Jeśli użytkownik nie jest autoryzowany, wywołanie nie powiedzie się.

Jednak programy serwerowe mogą chcieć, aby menedżer kolejek sprawdzał autoryzację użytkownika, dla którego pracują, a nie uprawnienia serwera. W tym celu należy użyć opcji

MQOO_ALTERNATE_USER_AUTHORITY wywołania MQOPEN i podać alternatywny identyfikator użytkownika w polu *AlternateUserId* struktury MQOD. Zwykle serwer uzyskałby identyfikator użytkownika z informacji kontekstowych w komunikacie, który jest przetwarzany.

Opcja MQOPEN dla wygaszania menedżera kolejek
W środowisku CICS w systemie z/OS, jeśli używany jest wywołanie MQOPEN, gdy menedżer kolejek jest w stanie wygaszania, wywołanie zawsze nie powiedzie się.

W innych środowiskach z/OS , IBM i, Windows oraz w środowiskach systemów UNIX and Linux wywołanie kończy się niepowodzeniem, gdy menedżer kolejek jest wygaszany tylko wtedy, gdy używana jest opcja MQOO_FAIL_IF_QUIESCING wywołania MQOPEN.

Opcja MQOPEN dla rozstrzygania nazw kolejek lokalnych
Po otwarciu lokalnej kolejki aliasowej lub kolejki modelowej zwracana jest kolejka lokalna.

Jednak po otwarciu kolejki zdalnej lub kolejki klastra pola *ResolvedQName* i *ResolvedQMgrName* struktury MQOD są zapełniane nazwami kolejek zdalnych i zdalnego menedżera kolejek, które znajdują się w definicji kolejki zdalnej, lub w wybranej zdalnej kolejce klastra.

Użyj opcji MQOO_RESOLVE_LOCAL_Q w wywołaniu MQOPEN, aby wypełnić *ResolvedQName* w strukturze MQOD nazwą kolejki lokalnej, która została otwarta. *ResolvedQMgrName* jest podobnie wypełniany nazwą lokalnego menedżera kolejek udostępniającego kolejkę lokalną. To pole jest dostępne tylko dla wersji 3 struktury MQOD. Jeśli struktura jest mniejsza niż wersja 3, parametr MQOO_RESOLVE_LOCAL_Q jest ignorowany, jeśli nie zostanie zwrócony błąd.

Jeśli podczas otwierania zostanie podana wartość MQOO_RESOLVE_LOCAL_Q, na przykład kolejka zdalna, *ResolvedQName* to nazwa kolejki transmisji, do której będą umieszczane komunikaty. *ResolvedQMgrName*to nazwa lokalnego menedżera kolejek udostępniającego kolejkę transmisji.

Tworzenie kolejek dynamicznych

Kolejki dynamicznej należy używać, gdy nie jest potrzebna kolejka po zakończeniu aplikacji.

Na przykład można użyć kolejki dynamicznej w celu uzyskania kolejki odpowiedzi. Nazwę kolejki odpowiedzi należy określić w polu *ReplyToQ* struktury MQMD po umieszczniu komunikatu w kolejce (patrz sekcja “Definiowanie komunikatów przy użyciu struktury MQMD” na stronie 232).

Aby utworzyć kolejkę dynamiczną, należy użyć szablonu znanego jako kolejka modelowa wraz z wywołaniem MQOPEN. Kolejkę modelową tworzy się za pomocą komend produktu WebSphere MQ lub paneli operacji i sterowania. Tworzona kolejka dynamiczna przyjmuje atrybuty kolejki modelowej.

Po wywołaniu komendy MQOPEN należy określić nazwę kolejki modelowej w polu *ObjectName* struktury MQOD. Po zakończeniu wywołania pole *ObjectName* jest ustawione na nazwę utworzonej kolejki dynamicznej. Ponadto pole *ObjectQMgrName* jest ustawione na nazwę lokalnego menedżera kolejek.

Użytkownik może określić nazwę kolejki dynamicznej, która została utworzona na trzy sposoby:

- Podaj pełną nazwę w polu *DynamicQName* struktury MQOD.
- Podaj przedrostek nazwy (mniej niż 33 znaki) i pozwól menedżerowi kolejek na wygenerowanie pozostałej części nazwy. Oznacza to, że menedżer kolejek generuje unikalną nazwę, ale nadal istnieje kilka elementów sterujących (na przykład użytkownik może chcieć, aby każdy użytkownik mógł użyć określonego przedrostka, lub nadać specjalną klasyfikację zabezpieczeń kolejkom z określonym przedrostkiem w nazwie). Aby użyć tej metody, należy podać gwiazdkę (*) dla ostatniego niepustego znaku w polu *DynamicQName* . Nie należy podawać pojedynczej gwiazdki (*) dla nazwy kolejki dynamicznej.
- Należy zezwolić menedżerowi kolejek na wygenerowanie pełnej nazwy. Aby użyć tej metody, należy podać gwiazdkę (*) w pierwszej pozycji znaku w polu *DynamicQName* .

Więcej informacji na temat tych metod znajduje się w opisie pola DynamicQName .

Więcej informacji na temat kolejek dynamicznych zawiera sekcja Kolejki dynamiczne i modelowe .

Otwieranie kolejek zdalnych

Kolejka zdalna jest kolejką, której właścicielem jest menedżer kolejek inny niż ten, do którego aplikacja jest połączona.

Aby otworzyć zdalną kolejkę, należy użyć wywołania MQOPEN jako kolejki lokalnej. Nazwę kolejki można określić w następujący sposób:

1. W polu *ObjectName* struktury MQOD określ nazwę kolejki zdalnej, która jest znana jako *lokalny* menedżer kolejek.

   Uwaga: W tym przypadku pozostaw puste pole *ObjectQMgrName* .
2. W polu *ObjectName* struktury MQOD określ nazwę kolejki zdalnej, która jest znana jako *zdalny* menedżer kolejek. W polu *ObjectQMgrName* podaj jedną z następujących opcji:

   - Nazwa kolejki transmisji, która ma taką samą nazwę jak zdalny menedżer kolejek. Nazwa i wielkość liter (wielkie, małe lub mieszaninę) muszą być zgodne z *dokładnie*.
   - Nazwa obiektu aliasu menedżera kolejek, który jest tłumaczona na docelowy menedżer kolejek lub kolejkę transmisji.

   Powoduje to, że menedżer kolejek jest miejscem docelowym komunikatu, a także kolejką transmisji, którą należy umieścić w celu uzyskania informacji o tym, czy ma być ona w tym miejscu.
3. Jeśli program *DefXmitQname* jest obsługiwany, w polu *ObjectName* struktury MQOD należy określić nazwę kolejki zdalnej, która jest znana przez *zdalny* menedżer kolejek.

   Uwaga: W polu *ObjectQMgrName* należy ustawić nazwę zdalnego menedżera kolejek (w tym przypadku nie może być ona pusta).

Podczas wywoływania komendy MQOPEN sprawdzana jest poprawność tylko nazw lokalnych; ostatnia operacja sprawdzania oznacza, że istnieje kolejka transmisji, która ma być używana.

Te metody są podsumowane w produkcie Tabela 34 na stronie 223.

Zamykanie obiektów przy użyciu wywołania MQCLOSE

Aby zamknąć obiekt, należy użyć wywołania MQCLOSE.

Jeśli obiekt jest kolejką, należy zwrócić uwagę na następujące informacje:

- Przed zamkniętą kolejką dynamiczną nie ma potrzeby tworzenia pustej kolejki dynamicznej.

  Po zamknięciu tymczasowej kolejki dynamicznej kolejka jest usuwana wraz z komunikatami, które mogą nadal być w niej dostępne. Jest to prawda, nawet jeśli nie istnieją niezatwierdzone wywołania MQGET, MQPUT lub MQPUT1 dla kolejki.
- W przypadku produktu WebSphere MQ for z/OS, jeśli istnieją żądania MQGET z opcją MQGMO_SET_SIGNAL dla tej kolejki, są one anulowane.
- Jeśli kolejka została otwarta za pomocą opcji MQOO_BROWSE, kursor przeglądania zostanie zniszczony.

Zamknięcie nie jest powiązane z punktem synchronizacji, dlatego można zamknąć kolejki przed lub po punkcie synchronizacji.

Jako dane wejściowe dla wywołania MQCLOSE należy podać:

- Uchwyt połączenia. Użyj tego samego uchwytu połączenia, który został użyty do jego otwarcia, lub w przypadku aplikacji CICS w systemie z/OSmożna określić stałą MQHC_DEF_HCONN (która ma wartość zero).
- Uchwyt obiektu, który ma zostać zamknięty. Pobierz to z danych wyjściowych wywołania MQOPEN.
- MQCO_NONE w polu *Options* (chyba że zamykano stałą kolejkę dynamiczną).
- Opcja sterująca, aby określić, czy menedżer kolejek powinien usunąć kolejkę, nawet jeśli nadal istnieją na niej komunikaty (przy zamykaniu trwałej kolejki dynamicznej).

Dane wyjściowe komendy MQCLOSE są następujące:

- Kod zakończenia
- Kod przyczyny
- Uchwyt obiektu, zresetuj do wartości MQHO_UNUSABLE_HOBJ

Opisy parametrów wywołania MQCLOSE są podane w tabeli MQCLOSE.

Umieszczanie komunikatów w kolejce

Te informacje umożliwiają zapoznanie się z informacjami na temat umieszczania komunikatów w kolejce.

Użyj wywołania MQPUT, aby umieścić komunikaty w kolejce. Za pomocą komendy MQPUT można wielokrotnie umieszczać wiele komunikatów w tej samej kolejce, po początkowym wywołaniu MQOPEN. Po zakończeniu umieszczania wszystkich komunikatów w kolejce należy wywołać komendę MQCLOSE.

Jeśli chcesz umieścić pojedynczy komunikat w kolejce i natychmiast zamknąć kolejkę, można użyć wywołania MQPUT1 . MQPUT1 wykonuje te same funkcje, co w przypadku następującej sekwencji wywołań:

- MQOPEN
- MQPUT
- MQCLOSE

Jeśli jednak istnieje więcej niż jeden komunikat do umieszczenia w kolejce, jest on bardziej wydajny, aby można było używać wywołania MQPUT. Zależy to od wielkości komunikatu i platformy, nad którą pracuje użytkownik.

Aby dowiedzieć się więcej na temat umieszczania komunikatów w kolejce, należy użyć następujących odsyłaczy:

- “Umieszczanie komunikatów w kolejce lokalnej przy użyciu wywołania MQPUT” na stronie 232
- “Umieszczanie komunikatów w kolejce zdalnej” na stronie 237
- “Ustawianie właściwości komunikatu” na stronie 237
- “Sterowanie informacjami kontekstową” na stronie 237
- “Umieszczanie jednego komunikatu w kolejce przy użyciu wywołania MQPUT1” na stronie 240
- “Lista dystrybucyjna” na stronie 241
- “Niektóre przypadki, w których wywołania put nie powiodły się” na stronie 246

Pojęcia pokrewne

“Interfejs kolejki komunikatów-przegląd” na stronie 199
Informacje na temat komponentów interfejsu kolejek komunikatów (Message Queue Interface-MQI).

“Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego” na stronie 211
Aby można było korzystać z usług programistycznych produktu WebSphere MQ , program musi mieć połączenie z menedżerem kolejek. Informacje zawarte w tej sekcji umożliwiają zapoznanie się z informacjami na temat nawiązywania połączenia z menedżerem kolejek i z jego rozłączeniem.

“Otwieranie i zamykanie obiektów” na stronie 220
Te informacje udostępniają wgląd w otwieranie i zamykanie obiektów WebSphere MQ .

“Pobieranie komunikatów z kolejki” na stronie 246
Ta sekcja zawiera informacje na temat pobierania komunikatów z kolejki.

“Sprawdzanie i ustawianie atrybutów obiektu” na stronie 328
Atrybuty to właściwości, które definiują parametry obiektu WebSphere MQ .

“Zatwierdzanie i wycofywanie jednostek pracy” na stronie 331
W tej sekcji opisano sposób zatwierdzania i tworzenia kopii zapasowych wszystkich możliwych do odtworzenia operacji get i put, które wystąpiły w jednostce pracy.

“Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy” na stronie 338

Informacje na temat wyzwalaczy i sposobów uruchamiania aplikacji IBM WebSphere MQ przy użyciu wyzwalaczy.

"Praca z interfejsem MQI i klastrami" na stronie 355
Istnieją specjalne opcje dotyczące wywołań i kodów powrotu, które odnoszą się do technologii klastrowej.

Umieszczanie komunikatów w kolejce lokalnej przy użyciu wywołania MQPUT

Ta sekcja zawiera informacje na temat umieszczania komunikatów w kolejce lokalnej przy użyciu wywołania MQPUT.

Jako dane wejściowe wywołania MQPUT należy podać:

- Uchwyt połączenia (Hconn).
- Uchwyt kolejki (Hobj).
- Opis komunikatu, który ma zostać umieszczony w kolejce. Ma to postać struktury deskryptora komunikatu (MQMD).
- Informacje sterujące w postaci struktury opcji put-message (put-message options-MQPMO).
- Długość danych zawartych w komunikacie (MQLONG).
- Same dane komunikatu.

Dane wyjściowe wywołania MQPUT są następujące:

- Kod przyczyny (MQLONG)
- Kod zakończenia (MQLONG)

Jeśli wywołanie zakończy się pomyślnie, to zwróci również strukturę opcji i strukturę deskryptora komunikatu. Wywołanie modyfikuje strukturę opcji w celu wyświetlenia nazwy kolejki i menedżera kolejek, do którego komunikat został wysłany. Jeśli użytkownik zażądał, aby menedżer kolejek wygenerował unikalną wartość identyfikatora wprowadzanego komunikatu (przez podanie wartości zero binarnej w polu *MsgId* struktury MQMD), wywołanie wstawia wartość w polu *MsgId* przed zwróceniem tej struktury do użytkownika. Przed wydaniem kolejnej operacji MQPUT należy zresetować tę wartość.

W tabeli MQPUTznajduje się opis wywołania MQPUT.

Więcej informacji na temat informacji wymaganych jako dane wejściowe dla wywołania MQPUT można znaleźć w następujących odsyłaczach:

- "Określanie uchwytów" na stronie 232
- "Definiowanie komunikatów przy użyciu struktury MQMD" na stronie 232
- "Określanie opcji przy użyciu struktury MQPMO" na stronie 233
- "Dane w komunikacie" na stronie 236
- "Umieszczanie komunikatów: korzystanie z uchwytów komunikatów" na stronie 237

Określanie uchwytów

Dla uchwytu połączenia (*Hconn*) w programie CICS w aplikacjach z/OS można określić stałą MQHC_DEF_HCONN (która ma wartość zero) lub uchwyt połączenia zwracany przez wywołanie MQCONN lub MQCONNX. W przypadku innych aplikacji zawsze należy używać uchwytu połączenia zwróconego przez wywołanie MQCONN lub MQCONNX.

Niezależnie od środowiska, w którym pracuje użytkownik, należy użyć tego samego uchwytu kolejki (*Hobj*), który jest zwracany przez wywołanie MQOPEN.

Definiowanie komunikatów przy użyciu struktury MQMD

Struktura deskryptora komunikatu (MQMD) jest parametrem wejściowym/wyjściowym dla wywołań MQPUT i MQPUT1 . Użyj go do zdefiniowania komunikatu umieszczanego w kolejce.

Jeśli wartość MQPRI_PRIORITY_AS_Q_DEF lub MQPER_PERSISTENCE_AS_Q_DEF została określona dla komunikatu, a kolejka jest kolejką klastra, użyte wartości są wartościami kolejki, do której komenda MQPUT jest tłumaczona. Jeśli ta kolejka jest wyłączona dla operacji MQPUT, wywołanie nie powiedzie się. Więcej informacji na ten temat zawiera sekcja Konfigurowanie klastra menedżera kolejek .

Uwaga: Przed umieszczniem nowego komunikatu należy użyć wartości MQPMO_NEW_MSG_ID i MQPMO_NEW_CORREL_ID, aby upewnić się, że *MsgId* i *CorrelId* są unikalne. Wartości w tych polach są zwracane w przypadku pomyślnej operacji MQPUT.

Istnieje wprowadzenie do właściwości komunikatu, które opisano w tabeli MQMD w produkcie “Komunikaty produktu IBM WebSphere MQ” na stronie 10, a opis samej struktury znajduje się w MQMD.

Określanie opcji przy użyciu struktury MQPMO

Struktura MQPMO (Put Message Option) służy do przekazywania opcji do wywołań MQPUT i MQPUT1 .

Poniższe sekcje dają pomoc przy wypełnianiu pól tej struktury. Opis struktury znajduje się w MQPMO.

Struktura obejmuje następujące pola:

- *StrucId*
- *Version*
- *Options*
- *Context*
- *ResolvedQName*
- *ResolvedQMgrName*
- *RecsPresent*
- *PutMsgRecsFields*
- *ResponseRecOffset and ResponseRecPtr*
- *OriginalMsgHandle*
- *NewMsgHandle*
- *Action*
- *PubLevel*

Zawartość tych pól jest następująca:

StrucId
Identyfikuje strukturę jako strukturę opcji komunikatów put. Jest to 4-znakowe pole. Zawsze należy podać identyfikator MQPMO_STRUC_ID.

Wersja
W tym temacie opisano numer wersji struktury. Wartość domyślna to MQPMO_VERSION_1. Jeśli zostanie wprowadzona wartość MQPMO_VERSION_2, można użyć list dystrybucyjnych (patrz “Lista dystrybucyjna” na stronie 241). Jeśli zostanie wprowadzona wartość MQPMO_VERSION_3, można użyć uchwytów komunikatów i właściwości komunikatów. Jeśli zostanie wprowadzona wartość MQPMO_CURRENT_VERSION, aplikacja jest ustawiana zawsze w celu użycia najnowszej wersji.

Opcje
Steruje on następującymi elementami:

- Informacja o tym, czy operacja put jest uwzględniona w jednostce pracy
- Informacje o tym, ile informacji kontekstowych jest powiązanych z komunikatem
- Miejsce, z którego pochodzą informacje o kontekście
- Określa, czy wywołanie nie powiedzie się, jeśli menedżer kolejek jest w stanie wygaszania.
- Określa, czy grupowanie lub segmentacja jest dozwolona
- Generowanie nowego identyfikatora komunikatu i identyfikatora korelacji
- Kolejność umieszczania komunikatów i segmentów w kolejce.

- Określenie, czy należy rozstrzygnąć nazwy kolejek lokalnych

Jeśli pole *Options* zostanie pozostaw ustawione na wartość domyślną (MQPMO_NONE), zostanie wyświetlony komunikat zawierający domyślne informacje o kontekście.

Ponadto sposób, w jaki połączenie działa z punktami synchronizacji, jest określany przez platformę. Domyślnym elementem sterującym punktu synchronizacji jest yes w systemie z/OS. W przypadku innych platform nie jest on dostępny.

Kontekst
Określa nazwę uchwytu kolejki, z którego mają być kopiowane informacje kontekstowe (jeśli jest to wymagane w polu *Options*).

Informacje na temat wprowadzenia do kontekstu komunikatów zawiera sekcja "Kontekst komunikatu" na stronie 39. Informacje na temat używania struktury MQPMO w celu sterowania informacjami o kontekście w komunikacie zawiera sekcja "Sterowanie informacjami kontekstową" na stronie 237.

ResolvedQName
Nazwa ta zawiera nazwę (po rozstrzygnięcie dowolnej nazwy aliasu) kolejki, która została otwarta w celu odebrania komunikatu. To jest pole wyjściowe.

Nazwa ResolvedQMgr
Nazwa ta zawiera nazwę (po rozstrzygnięcie dowolnej nazwy aliasu) menedżera kolejek, który jest właścicielem kolejki w programie *ResolvedQName*. To jest pole wyjściowe.

Zmaterializowana tabela zapytania (MQPMO) może również zawierać pola wymagane dla list dystrybucyjnych (patrz "Lista dystrybucyjna" na stronie 241). Jeśli ma być używany ten obiekt, używana jest wersja 2 struktury MQPMO. Obejmuje to następujące pola:

RecsPresent
To pole zawiera liczbę kolejek znajdujących się na liście dystrybucyjnej, to znaczy liczbę obecnych rekordów operacji umieszczania komunikatów (MQPMR) i odpowiadających im rekordów odpowiedzi (MQRR).

Wprowadzona wartość może być taka sama, jak liczba rekordów obiektów udostępnionych w tabeli MQOPEN. Jeśli jednak wartość jest mniejsza niż liczba rekordów obiektów udostępnionych w wywołaniu MQOPEN lub jeśli użytkownik nie udostępni żadnych rekordów umieszczania komunikatów, wartości tych kolejek, które nie są zdefiniowane, są pobierane z wartości domyślnych udostępnianych przez deskryptor komunikatu. Ponadto, jeśli wartość jest większa niż liczba udostępnionych rekordów obiektów, nadmiarowe rekordy umieszczania komunikatów są ignorowane.

Zaleca się, aby wykonać jedną z następujących czynności:

- Jeśli chcesz otrzymywać raport lub odpowiedź z każdego miejsca docelowego, wprowadź tę samą wartość, która jest wyświetlana w strukturze MQOR i użyj pól MQPMRs zawierających pola *MsgId* . Zainicjuj te pola *MsgId* na zero lub podaj MQPMO_NEW_MSG_ID.

  Po umieszczniu komunikatu w kolejce wartości *MsgId* utworzone przez menedżer kolejek stają się dostępne w produkcie MQPMRs;, aby określić, które miejsce docelowe jest powiązane z każdym raportem lub odpowiadając na odpowiedź.
- Jeśli nie chcesz otrzymywać raportów lub odpowiedzi, wybierz jedną z następujących opcji:

  1. Aby zidentyfikować miejsca docelowe, które nie powiodły się natychmiast, można w dalszym ciągu wprowadzić tę samą wartość w polu *RecsPresent* , co jest wyświetlane w strukturze MQOR i zapewnić produktom MQRRs identyfikowanie tych miejsc docelowych. Nie należy określać żadnych obiektów MQPMRs.
  2. Jeśli nie chcesz określać miejsc docelowych, które nie powiodły się, wprowadź wartość zero w polu *RecsPresent* i nie dostarczaj tabel MQPMR ani MQRRs.

Uwaga: Jeśli używana jest wartość MQPUT1, liczba wskaźników rekordów odpowiedzi i przesunięć rekordów odpowiedzi musi wynosić zero.

Pełny opis rekordów Put Message Records (MQPMR) i Response Records (MQRR) można znaleźć w sekcji MQPMR i MQRR.

PutMsgRecFields
Wskazuje to, które pola są obecne w każdym rekordzie komunikatu umieszczania komunikatów (MQPMR). Listę tych pól można znaleźć w sekcji “Korzystanie ze struktury MQPMR” na stronie 245.

PutMsgRecOffset* i *PutMsgRecPtr
Wskaźniki (zwykle w języku C) i offsety (zwykle w języku COBOL) są używane do obsługi rekordów umieszczania komunikatów (patrz sekcja “Korzystanie ze struktury MQPMR” na stronie 245 , aby uzyskać przegląd struktury MQPMR).

Użyj pola *PutMsgRecPtr* , aby określić wskaźnik do pierwszego rekordu umieszczonego komunikatu, lub pole *PutMsgRecOffset* , aby określić przesunięcie pierwszego rekordu umieszczania komunikatu. Jest to przesunięcie od początku wywołania MQPMO. W zależności od pola *PutMsgRecFields* wprowadź wartość inną niż NULL dla *PutMsgRecOffset* lub *PutMsgRecPtr*.

ResponseRecPrzesunięcie i ResponseRecPtr
W celu uzyskania dalszych informacji na temat rekordów odpowiedzi można również użyć wskaźników i przesunięć w celu zaadresowania rekordów odpowiedzi (więcej informacji można znaleźć w sekcji “Korzystanie ze struktury MQRR” na stronie 244).

Użyj pola *ResponseRecPtr* , aby określić wskaźnik do pierwszego rekordu odpowiedzi, lub pole *ResponseRecOffset* , aby określić przesunięcie pierwszego rekordu odpowiedzi. Jest to przesunięcie od początku struktury MQPMO. Wprowadź wartość inną niż NULL dla opcji *ResponseRecOffset* lub *ResponseRecPtr*.

Uwaga: Jeśli do umieszczania komunikatów na liście dystrybucyjnej używany jest parametr MQPUT1 , wartość *ResponseRecPtr* musi mieć wartość NULL lub wartość zero, a wartość *ResponseRecOffset* musi wynosić zero.

W wersji 3 struktury MQPMO dodatkowo znajdują się następujące pola:

Uchwyt komunikatu OriginalMsg
Użycie tego pola może być uzależnione od wartości pola *Działanie* . W przypadku umieszczania nowego komunikatu z powiązanymi właściwościami komunikatu należy ustawić to pole na uchwyt komunikatu, który został wcześniej utworzony, i ustawić właściwości na. W przypadku przekazywania, odpowiadania na raport lub generowania raportu w odpowiedzi na wcześniej pobrany komunikat, pole to zawiera uchwyt komunikatu tego komunikatu.

Uchwyt NewMsg
Jeśli zostanie określony uchwyt *NewMsg(Nowy komunikat)*, wszystkie właściwości powiązane z właściwościami nadpisania uchwytu są powiązane z uchwytem *OriginalMsgHandle*. Więcej informacji na ten temat zawiera sekcja Działanie (MQLONG).

Działanie
To pole służy do określania typu wykonywanego zadania. Możliwe wartości i ich znaczenia są następujące:

MQACTP_NEW
Jest to nowy komunikat niezwiązany z żadnym innym.

MQACTP_FORWARD
Ten komunikat został pobrany wcześniej i jest teraz przekazywany.

ODPOWIEDŹ MQACTP_REPLY
Ten komunikat jest odpowiedzią na wcześniej pobrany komunikat.

RAPORT MQACTP_REPORT
Ten komunikat jest raportem wygenerowanym w wyniku wcześniej pobranego komunikatu.

Więcej informacji na ten temat zawiera sekcja Działanie (MQLONG).

PubLevel
Jeśli ten komunikat jest publikacją, można ustawić to pole w celu określenia, które subskrypcje mają być odbierane. Ta publikacja otrzyma tylko subskrypcje o wartości *SubLevel* mniejszej lub równej tej wartości. Wartością domyślną jest 9, która jest najwyższym poziomem i oznacza, że subskrypcje z dowolnym *SubLevel* mogą otrzymać tę publikację.

Dane w komunikacie

Należy podać adres buforu, który zawiera dane w parametrze *Buffer* wywołania MQPUT.
W komunikatach można umieścić dowolne dane w komunikatach. Ilość danych w komunikatach wpływa jednak na wydajność aplikacji, która je przetwarza.

Maksymalna wielkość danych jest określana przez:

- Atrybut *MaxMsgLength* menedżera kolejek
- Atrybut *MaxMsgLength* kolejki, w której jest wstawiany komunikat.
- Wielkość nagłówka komunikatu dodanego przez produkt WebSphere MQ (w tym nagłówek dead-letter, MQDLH i nagłówek listy dystrybucyjnej, MQDH)

Atrybut *MaxMsgLength* menedżera kolejek przechowuje wielkość komunikatu, który może być przetwarzany przez menedżer kolejek. Wartość domyślna to 100 MB dla wszystkich produktów WebSphere MQ na poziomie V6 lub nowszym.

Aby określić wartość tego atrybutu, należy użyć wywołania MQINQ w obiekcie menedżera kolejek.
W przypadku dużych komunikatów można zmienić tę wartość.

Atrybut *MaxMsgLength* kolejki określa maksymalną wielkość komunikatu, który można umieścić w kolejce. W przypadku próby umieszczenia komunikatu o wielkości większej niż wartość tego atrybutu wywołanie MQPUT nie powiedzie się. Jeśli komunikat jest umieszczany w kolejce zdalnej, maksymalna wielkość komunikatu, którą można pomyślnie umieścić, jest określana przez atrybut *MaxMsgLength* kolejki zdalnej, wszystkich pośrednich kolejek transmisji, które komunikat jest umieszczany na trasie, do miejsca docelowego i używanych kanałów.

W przypadku operacji MQPUT wielkość komunikatu musi być mniejsza lub równa atrybutowi *MaxMsgLength* zarówno w kolejce, jak i w menedżerze kolejek. Wartości tych atrybutów są niezależne, ale zalecane jest ustawienie *MaxMsgLength* kolejki na wartość mniejszą lub równą wartości tego menedżera kolejek.

Produkt WebSphere MQ dodaje informacje nagłówka do komunikatów w następujących okolicznościach:

- Po umieszczniu komunikatu w kolejce zdalnej produkt WebSphere MQ dodaje do komunikatu strukturę nagłówka transmisji (MQXQH). Ta struktura obejmuje nazwę kolejki docelowej i jej menedżera kolejek, do którego należy.
- Jeśli produkt WebSphere MQ nie może dostarczyć komunikatu do kolejki zdalnej, podejmuje próbę umieszczenia komunikatu w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów). Dodaje ona strukturę MQDLH do komunikatu. Struktura ta obejmuje nazwę kolejki docelowej oraz przyczynę umieszczenia komunikatu w kolejce niedostarczonych komunikatów.
- Aby wysłać komunikat do wielu kolejek docelowych, produkt WebSphere MQ doda do komunikatu nagłówek MQDH. Opisuje dane znajdujące się w komunikacie, należącym do listy dystrybucyjnej, w kolejce transmisji. Należy wziąć pod uwagę to, wybierając optymalną wartość dla maksymalnej długości komunikatu.
- Jeśli komunikat jest segmentem lub komunikatem w grupie, produkt WebSphere MQ może dodać produkt MQMDE.

Struktury te są opisane w tabelach MQDH i MQMDE.

Jeśli komunikaty mają maksymalną wielkość dozwoloną dla tych kolejek, dodanie tych nagłówków oznacza, że operacje put nie powiodą się, ponieważ komunikaty są teraz zbyt duże. Aby zmniejszyć prawdopodobieństwo niepowodzenia operacji put:

- Wielkość komunikatów musi być mniejsza niż wartość atrybutu *MaxMsgLength* kolejek transmisji i niedostarczonych komunikatów. Zezwól co najmniej na wartość stałej MQ_MSG_HEADER_LENGTH (więcej dla dużych list dystrybucyjnych).
- Upewnij się, że atrybut *MaxMsgLength* w kolejce niedostarczonych komunikatów jest ustawiony na wartość taką samą, jak *MaxMsgLength* menedżera kolejek, który jest właścicielem kolejki niedostarczonych komunikatów.

Atrybuty dla menedżera kolejek i stałe kolejkowania komunikatów są opisane w sekcji Atrybuty dla menedżera kolejek.

Umieszczanie komunikatów: korzystanie z uchwytów komunikatów

Dwa uchwyty komunikatów są dostępne w strukturze MQPMO, *Uchwyt OriginalMsg* i *Uchwyt NewMsg*. Relacja między tymi uchwytami komunikatów jest definiowana przez wartość w polu *Działanie* MQPMO.

Szczegółowe informacje na ten temat zawiera sekcja Działanie (MQLONG). Uchwyt komunikatu nie musi być wymagany w celu umieszczenia komunikatu. Jego celem jest powiązanie właściwości z komunikatem, więc jest on wymagany tylko wtedy, gdy używane są właściwości komunikatu.

Umieszczanie komunikatów w kolejce zdalnej

Aby umieścić komunikat w kolejce zdalnej (czyli w kolejce należnej do menedżera kolejek innego niż ten, do którego aplikacja jest podłączona), a nie do kolejki lokalnej, jedynym dodatkowym zagadnieniem jest sposób określania nazwy kolejki po jej otwarciu.Jest to opisane w sekcji “Otwieranie kolejek zdalnych” na stronie 230. Nie ma żadnych zmian w sposobie użycia wywołania MQPUT lub MQPUT1 dla kolejki lokalnej.

Więcej informacji na temat używania kolejek zdalnych i kolejek transmisji zawiera sekcja Techniki rozproszonego przesyłania komunikatów programu WebSphere MQ .

Ustawianie właściwości komunikatu

Dla każdej właściwości, która ma zostać ustawiona, wywołaj komendę MQSETMP. Po umieszczonym w komunikacie uchwycie komunikatu i polach działania struktury MQPMO.

Aby powiązać właściwości z komunikatem, komunikat musi mieć uchwyt komunikatu. Utwórz uchwyt komunikatu przy użyciu wywołania funkcji MQCRTMH. Wywołaj komendę MQSETMP, podając ten uchwyt komunikatu dla każdej właściwości, która ma zostać ustawiona. Dostępny jest przykładowy program amqsstma.cilustrujący użycie komendy MQSETMP.

Jeśli jest to nowy komunikat, po umieszczniu go w kolejce przy użyciu komendy MQPUT lub MQPUT1należy ustawić pole uchwytu OriginalMsgw MQPMO na wartość tego uchwytu komunikatu, a następnie ustawić pole MQPMO Action na wartość MQACTP_NEW (jest to wartość domyślna).

Jeśli jest to komunikat, który został wcześniej pobrany, a następnie przekazujesz lub odpowiadasz na niego lub wysyłając raport w odpowiedzi na ten komunikat, umieść ten oryginalny uchwyt komunikatu w polu OriginalMsguchwytu MQPMO i nowym uchwycie komunikatu w polu Uchwyt NewMsg. W zależności od potrzeb ustaw pole Działanie odpowiednio na MQACTP_FORWARD, MQACTP_REPLY lub MQACTP_REPORT.

Jeśli użytkownik ma właściwości w nagłówku MQRFH2 z wcześniej pobranego komunikatu, można przekształcić je w właściwości uchwytu komunikatu przy użyciu wywołania MQBUFMH.

Jeśli komunikat jest umieszczany w kolejce menedżera kolejek na poziomie wcześniejszym niż WebSphere MQ 7.0, który nie może przetwarzać właściwości komunikatu, można ustawić parametr PropertyControl w definicji kanału, aby określić sposób, w jaki właściwości mają być traktowane.

Sterowanie informacjami kontekstową

W przypadku użycia wywołania MQPUT lub MQPUT1 w celu umieszczenia komunikatu w kolejce można określić, że menedżer kolejek ma dodać pewne domyślne informacje kontekstu do deskryptora komunikatu. Aplikacje, które mają odpowiedni poziom uprawnień, mogą dodawać dodatkowe informacje kontekstowe. Do sterowania informacjami kontekstowych można użyć pola opcji w strukturze MQPMO.

Aby sterować informacjami o kontekście, należy użyć pola *Options* w strukturze MQPMO.

W przeciwnym razie menedżer kolejek nadpisuje informacje kontekstu, które mogą znajdować się już w deskryptorze komunikatu, wraz z informacjami o tożsamości i kontekście wygenerowanymi dla komunikatu. Jest to takie samo, jak określenie opcji MQPMO_DEFAULT_CONTEXT.Te domyślne informacje o kontekście mogą być potrzebne podczas tworzenia nowego komunikatu (na przykład podczas przetwarzania danych wejściowych użytkownika z ekranu uzyskiwania informacji).

Jeśli z komunikatem nie ma powiązanych informacji o kontekście, należy skorzystać z opcji MQPMO_NO_CONTEXT. Podczas umieszczania komunikatu bez kontekstu wszystkie operacje sprawdzania uprawnień dokonywane przez produkt IBM WebSphere MQ są wykonywane przy użyciu pustego ID użytkownika. Pusty identyfikator użytkownika nie może być przypisany do jawnego uprawnienia do zasobów IBM WebSphere MQ , ale jest traktowany jako członek grupy specjalnej 'nobody'. Więcej informacji na temat grupy specjalnej nobodyzawiera sekcja Informacje uzupełniające dotyczące interfejsu usług instalowalnych.

Jeśli z komunikatem nie ma powiązanych informacji o kontekście, należy skorzystać z opcji MQPMO_NO_CONTEXT.

W poniższych sekcjach tego tematu wyjaśniono sposób użycia kontekstu tożsamości, kontekstu użytkownika i całego kontekstu.

Przekazywanie kontekstu tożsamości

Ogólnie programy powinny przekazywać informacje kontekstu tożsamości z komunikatu do komunikatu wokół aplikacji, dopóki dane nie dotrą do miejsca docelowego.

Programy powinny zmieniać informacje o kontekście pochodzenia za każdym razem, gdy zmieniają one dane. Jednak aplikacje, które chcą zmienić lub ustawić dowolne informacje kontekstowe, muszą mieć odpowiedni poziom uprawnień. Menedżer kolejek sprawdza to uprawnienie, gdy aplikacje otwierają kolejki; muszą mieć uprawnienia do korzystania z odpowiednich opcji kontekstu dla wywołania MQOPEN.

Jeśli aplikacja pobiera komunikat, przetwarza dane z komunikatu, a następnie umieszcza zmienione dane w innym komunikacie (może to być możliwe do przetworzenia przez inną aplikację), aplikacja musi przekazać informacje kontekstu tożsamości z oryginalnego komunikatu do nowego komunikatu. Użytkownik może zezwolić menedżerowi kolejek na tworzenie informacji o kontekście pochodzenia.

Aby zapisać informacje o kontekście z oryginalnego komunikatu, należy użyć opcji MQOO_SAVE_ALL_CONTEXT podczas otwierania kolejki w celu uzyskania komunikatu. Jest to uzupełnienie wszystkich innych opcji, które są używane z wywołaniem MQOPEN. Należy jednak pamiętać o tym, że nie można zapisać informacji kontekstowych, jeśli tylko użytkownik przegląda komunikat.

Podczas tworzenia drugiego komunikatu:

- Otwórz kolejkę za pomocą opcji MQOO_PASS_IDENTITY_CONTEXT (dodatkowo do opcji MQOO_OUTPUT).
- W polu *Context* struktury opcji put-message należy podać uchwyt kolejki, z której zostały zapisane informacje o kontekście.
- W polu *Options* w strukturze opcji put-message określ opcję MQPMO_PASS_IDENTITY_CONTEXT.

Przekazywanie kontekstu użytkownika

Nie można wybrać tylko do przekazania kontekstu użytkownika. Aby przekazać kontekst użytkownika podczas umieszczania komunikatu, należy podać parametr MQPMO_PASS_ALL_CONTEXT. Wszystkie właściwości w kontekście użytkownika są przekazywane w taki sam sposób, jak kontekst źródłowy.

Gdy trwa operacja MQPUT lub MQPUT1 , a kontekst jest przekazywany, wszystkie właściwości w kontekście użytkownika są przekazywane z pobranego komunikatu do komunikatu umieszczonego w komunikacie. Wszystkie właściwości kontekstu użytkownika, które zostały zmodyfikowane przez aplikację, są umieszczane razem z ich oryginalnymi wartościami. Wszystkie właściwości kontekstu użytkownika, które zostały usunięte przez umieszczanie aplikacji, są odtwarzane w komunikacie

umieszczonym. Wszystkie właściwości kontekstu użytkownika, które aplikacja dodała do wiadomości, są zachowywane.

Przekazywanie całego kontekstu

Jeśli aplikacja pobiera komunikat i umieszcza dane komunikatu (bez zmian) w innym komunikacie, aplikacja musi przekazać wszystkie informacje o kontekście (tożsamość, pochodzenie i użytkownika) z oryginalnego komunikatu do nowego komunikatu. Przykładem aplikacji, która może to zrobić, jest narzędzie przenoszenia komunikatów, które przenosi komunikaty z jednej kolejki do innej.

Wykonaj tę samą procedurę, co w przypadku przekazywania kontekstu tożsamości, z tym wyjątkiem, że używana jest opcja MQOPEN MQOO_PASS_ALL_CONTEXT, a opcja put-message MQPMO_PASS_ALL_CONTEXT.

Ustawianie kontekstu tożsamości

Jeśli chcesz ustawić informacje o kontekście tożsamości dla komunikatu:

- Otwórz kolejkę przy użyciu opcji MQOO_SET_IDENTITY_CONTEXT.
- Umieść komunikat w kolejce, podając opcję MQPMO_SET_IDENTITY_CONTEXT. W deskryptorze komunikatu określ wymagane informacje o kontekście tożsamości.

Uwaga: W przypadku ustawienia niektórych (ale nie wszystkich) pól kontekstu tożsamości za pomocą opcji MQOO_SET_IDENTITY_CONTEXT i opcji MQPMO_SET_IDENTITY_CONTEXT należy pamiętać, że menedżer kolejek nie ustawia żadnego z pozostałych pól.

Aby zmodyfikować dowolne opcje kontekstu komunikatu, należy mieć odpowiednie autoryzacje do wystawienia połączenia. Na przykład, aby można było używać funkcji MQOO_SET_IDENTITY_CONTEXT lub MQPMO_SET_IDENTITY_CONTEXT, użytkownik musi mieć uprawnienie `+setid` .

Ustawianie kontekstu użytkownika

Aby ustawić właściwość w kontekście użytkownika, należy ustawić pole Kontekst deskryptora właściwości komunikatu (MQPD) na wartość MQPD_USER_CONTEXT podczas wywoływania wywołania MQSETMP.

Aby ustawić właściwość w kontekście użytkownika, nie trzeba mieć żadnych uprawnień specjalnych. Kontekst użytkownika nie ma opcji kontekstu MQOO_SET_ * lub MQPMO_SET_ *.

Ustawianie całego kontekstu

Aby ustawić zarówno tożsamość, jak i informacje o kontekście pochodzenia dla komunikatu:

1. Otwórz kolejkę za pomocą opcji MQOO_SET_ALL_CONTEXT.
2. Umieść komunikat w kolejce, podając opcję MQPMO_SET_ALL_CONTEXT. W deskryptorze komunikatu określ informacje o tożsamości i pochodzeniu, które są wymagane.

Dla każdego typu ustawienia kontekstu wymagane jest odpowiednie uprawnienie.

Pojęcia pokrewne

“Kontekst komunikatu” na stronie 39
Informacja *Kontekst komunikatu* umożliwia aplikacji, która pobiera komunikat, w celu uzyskania informacji o inicjatorze komunikatu.

Odsyłacze pokrewne

“Opcje MQOPEN związane z kontekstem komunikatu” na stronie 228
Aby możliwe było powiązanie informacji kontekstowych z komunikatem podczas umieszczania go w kolejce, należy użyć jednej z opcji kontekstu komunikatu podczas otwierania kolejki.

Umieszczanie jednego komunikatu w kolejce przy użyciu wywołania MQPUT1

Użyj wywołania MQPUT1 , jeśli chcesz zamknąć kolejkę natychmiast po umieszczonym na nim pojedynczym komunikacie. Na przykład aplikacja serwera może używać wywołania MQPUT1 , gdy wysyła odpowiedź do każdej z różnych kolejek.

Parametr MQPUT1 jest funkcjonalnie równoważny z wywołaniem komendy MQOPEN, po której następuje wywołanie MQPUT, a następnie MQCLOSE. Jedyna różnica w składni wywołań MQPUT i MQPUT1 jest taka, że dla operacji MQPUT należy określić uchwyt obiektu, natomiast dla parametru MQPUT1 należy określić strukturę deskryptora obiektu (MQOD) zgodnie z definicją w tabeli MQOPEN (patrz sekcja “Identyfikowanie obiektów (struktura MQOD)” na stronie 222). Jest to spowodowane tym, że należy podać informacje dla wywołania MQPUT1 dotyczące kolejki, która ma zostać otwarta, podczas gdy podczas wywoływania operacji MQPUT kolejka musi być już otwarta.

Jako dane wejściowe dla wywołania MQPUT1 należy podać:

- Uchwyt połączenia.
- Opis obiektu, który ma zostać otwarty. Ma to postać struktury deskryptora obiektu (MQOD).
- Opis komunikatu, który ma zostać umieszczony w kolejce. Ma to postać struktury deskryptora komunikatu (MQMD).
- Informacje sterujące w postaci struktury opcji komunikatu put (put-message options structure- MQPMO).
- Długość danych zawartych w komunikacie (MQLONG).
- Adres danych komunikatu.

Dane wyjściowe komendy MQPUT1 są następujące:

- Kod zakończenia
- Kod przyczyny

Jeśli wywołanie zakończy się pomyślnie, to zwróci również strukturę opcji i strukturę deskryptora komunikatu. Wywołanie modyfikuje strukturę opcji w celu wyświetlenia nazwy kolejki i menedżera kolejek, do którego komunikat został wysłany. Jeśli użytkownik zażądał, aby menedżer kolejek wygenerował unikalną wartość dla identyfikatora wprowadzanego komunikatu (przez podanie wartości zero w polu *MsgId* struktury MQMD), wywołanie wstawia wartość w polu *MsgId* przed zwróceniem tej struktury do użytkownika.

Uwaga: Nie można użyć komendy MQPUT1 z nazwą kolejki modelowej, jednak po otwarciu kolejki modelowej można wydać komendę MQPUT1 do kolejki dynamicznej.

Sześć parametrów wejściowych dla parametru MQPUT1 to:

Hconn
To jest uchwyt połączenia. W przypadku aplikacji CICS można określić stałą MQHC_DEF_HCONN (która ma wartość zero), lub użyć uchwytu połączenia zwróconego przez wywołanie MQCONN lub MQCONNX. W przypadku innych programów zawsze należy używać uchwytu połączenia zwróconego przez wywołanie MQCONN lub MQCONNX.

ObjDesc
Jest to struktura deskryptora obiektu (MQOD).

W polach *ObjectName* i *ObjectQMgrName* podaj nazwę kolejki, w której ma zostać umieszczony komunikat, oraz nazwę menedżera kolejek, który jest właścicielem tej kolejki.

Pole *DynamicQName* jest ignorowane dla wywołania MQPUT1 , ponieważ nie może korzystać z kolejek modelowych.

Użyj pola *AlternateUserId* , aby nominować alternatywny identyfikator użytkownika, który ma być używany do testowania uprawnień do otwarcia kolejki.

MsgDesc
Jest to struktura deskryptora komunikatu (MQMD). Podobnie jak w przypadku wywołania MQPUT, ta struktura służy do definiowania komunikatu umieszczanego w kolejce.

PutMsgOpts
Jest to struktura opcji komunikatów typu put (put-message options) (MQPMO). Należy go używać w taki sam sposób, jak w przypadku wywołania MQPUT (patrz sekcja "Określanie opcji przy użyciu struktury MQPMO" na stronie 233).

Gdy pole *Options* jest ustawione na zero, menedżer kolejek używa własnego ID użytkownika podczas wykonywania testów dla uprawnień dostępu do kolejki. Ponadto menedżer kolejek ignoruje dowolny alternatywny identyfikator użytkownika podany w polu *AlternateUserId* struktury MQOD.

BufferLength
Jest to długość wiadomości.

Buffer
Jest to obszar buforu, który zawiera tekst komunikatu.

Jeśli używane są klastry, komenda MQPUT1 działa tak, jakby w efekcie działa MQOO_BIND_NOT_FIXED. Aplikacje muszą używać rozstrzygniętych pól w strukturze MQPMO, a nie struktury MQOD w celu określenia miejsca, w którym komunikat został wysłany. Więcej informacji na ten temat zawiera sekcja Konfigurowanie klastra menedżera kolejek .

W pliku MQPUT1znajduje się opis wywołania MQPUT1 .

Lista dystrybucyjna

Nie jest obsługiwane w produkcie WebSphere MQ for z/OS. Listy dystrybucyjne umożliwiają umieszczenie komunikatu w wielu miejscach docelowych w pojedynczym wywołaniu MQPUT lub MQPUT1 .Pojedyncze wywołanie MQOPEN może otworzyć wiele kolejek, a pojedyncze wywołanie MQPUT może następnie umieścić komunikat dla każdej z tych kolejek. Niektóre informacje ogólne ze struktur MQI używanych dla tego procesu mogą zostać zastąpione konkretnymi informacjami dotyczącymi poszczególnych miejsc docelowych znajdujących się na liście dystrybucyjnej.

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.

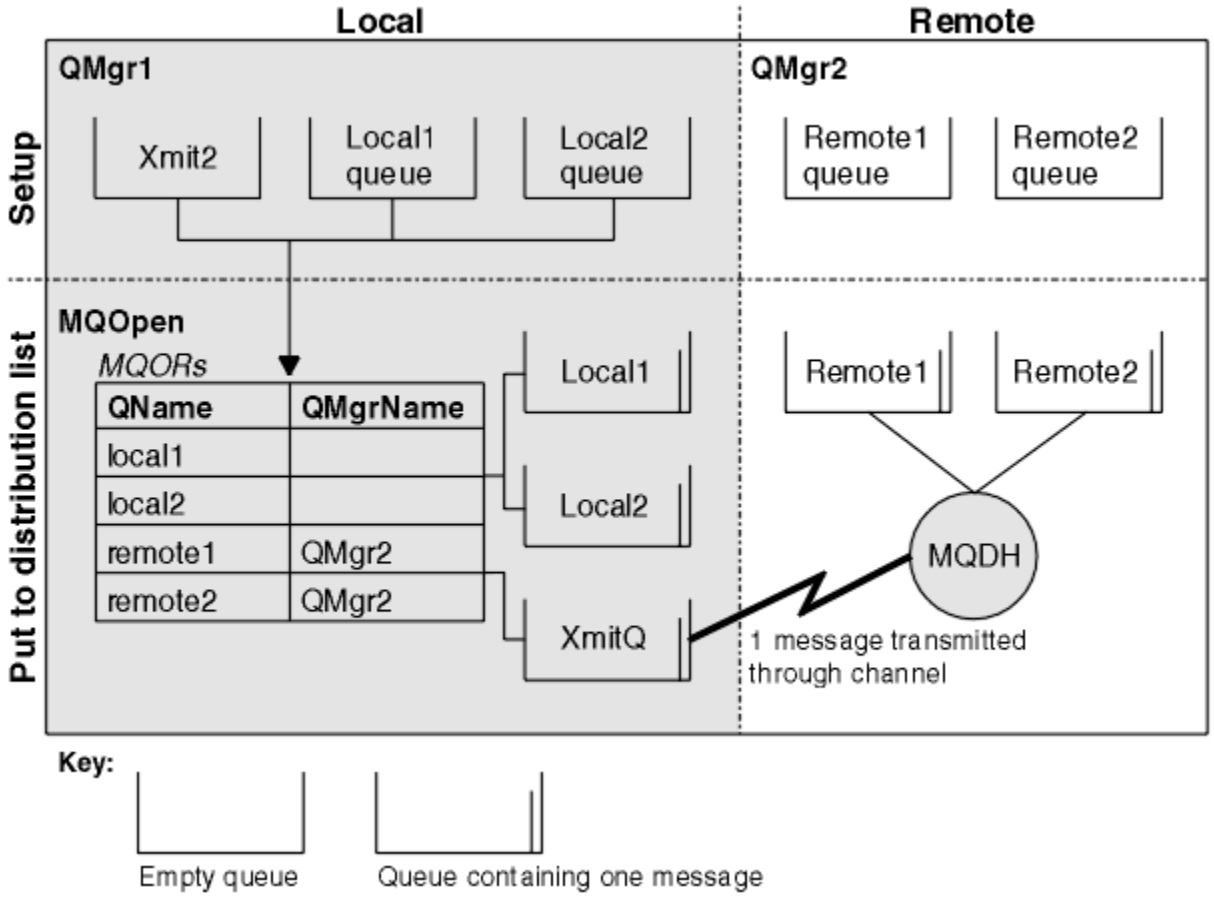
Po wywołaniu wywołania MQOPEN z deskryptora obiektu (MQOD) pobierane są ogólne informacje. Jeśli w polu *Version* zostanie podana wartość MQOD_VERSION_2 , a w polu *RecsPresent* wartość większa od zera, to *Hobj* może zostać zdefiniowany jako uchwyt listy (jednej lub większej liczby kolejek), a nie kolejki. W tym przypadku konkretne informacje są podawane za pośrednictwem rekordów obiektów (MQORs), które nadają szczegółowe informacje na temat miejsca docelowego (to znaczy *ObjectName* i *ObjectQMgrName*).

Uchwyt obiektu (*Hobj*) jest przekazywany do wywołania MQPUT, co pozwala na umieszczenie listy, a nie do pojedynczej kolejki.

Gdy komunikat jest umieszczany w kolejkach (MQPUT), informacje ogólne są pobierane ze struktury opcji umieszczania komunikatów (Put Message Option structure-MQPMO) i deskryptora komunikatu (Message Descriptor-MQMD). Konkretne informacje są podane w postaci rekordów komunikatu umieszczania komunikatów (Put Message Records-MQPMRs).

Rekordy odpowiedzi (MQRR) mogą otrzymać kod zakończenia i kod przyczyny specyficzne dla każdej kolejki docelowej.

Rysunek 29 na stronie 242 pokazuje, w jaki sposób działają listy dystrybucyjne.



Rysunek 29. Sposób działania list dystrybucyjnych

Otwieranie list dystrybucyjnych

Użyj wywołania MQOPEN, aby otworzyć listę dystrybucyjną, a następnie użyj opcji wywołania, aby określić, co ma być używane z listą.

Jako dane wejściowe dla komendy MQOPEN należy podać:

- Uchwyt połączenia (patrz “Umieszczanie komunikatów w kolejce” na stronie 231 , aby uzyskać opis)
- Informacje ogólne w strukturze deskryptora obiektu (MQOD)
- Nazwa każdej kolejki, która ma być otwarta, przy użyciu struktury rekordów obiektów (MQOR)

Dane wyjściowe komendy MQOPEN są następujące:

- Uchwyt obiektu, który reprezentuje dostęp użytkownika do listy dystrybucyjnej
- Ogólny kod zakończenia
- Ogólny kod przyczyny
- Rekordy odpowiedzi (opcjonalne), zawierające kod zakończenia i przyczynę dla każdego miejsca docelowego

Korzystanie ze struktury MQOD

Użyj struktury MQOD, aby zidentyfikować kolejki, które mają zostać otwarte.

Aby zdefiniować listę dystrybucyjną, w polu *Version* należy podać wartość MQOD_VERSION_2 , wartość większą od zera w polu *RecsPresent* oraz wartość MQOT_Q w polu *ObjectType* . Sekcja MQOD zawiera opis wszystkich pól struktury MQOD.

Korzystanie ze struktury MQOR

Podaj strukturę MQOR dla każdego miejsca docelowego.

Struktura zawiera nazwy kolejek docelowych i menedżerów kolejek. Pola *ObjectName* i *ObjectQMgrName* w tabeli MQOD nie są używane dla list dystrybucyjnych. Musi istnieć jeden lub więcej rekordów obiektów. Jeśli pole *ObjectQMgrName* pozostanie puste, zostanie użyty lokalny menedżer kolejek. Więcej informacji na temat tych pól można znaleźć w sekcji ObjectName i ObjectQMgrName (Nazwa obiektu ObjectQMgr).

Kolejki docelowe można określić na dwa sposoby:

- Za pomocą pola przesunięcia *ObjectRecOffset*.

  W takim przypadku aplikacja musi zadeklarować własną strukturę zawierającą strukturę MQOD, po której następuje tablica rekordów MQOR (wraz z wymaganą wieloma elementami tablicy), a następnie ustaw *ObjectRecOffset* na przesunięcie pierwszego elementu w tablicy od początku tabeli MQOD. Upewnij się, że przesunięcie jest poprawne.

  Zaleca się korzystanie z wbudowanych urządzeń udostępnianych przez język programowania, jeśli są one dostępne we wszystkich środowiskach, w których działa aplikacja. Poniższy kod ilustruje tę technikę dla języka programowania COBOL:

```
01  MY-OPEN-DATA.
    02 MY-MQOD.
       COPY CMQODV.
    02 MY-MQOR-TABLE OCCURS 100 TIMES.
       COPY CMQORV.
    MOVE LENGTH OF MY-MQOD TO MQOD-OBJECTRECOFFSET.
```

  Alternatywnie można użyć stałej MQOD_CURRENT_LENGTH, jeśli język programowania nie obsługuje niezbędnych wbudowanych urządzeń we wszystkich środowiskach, których to dotyczy. Poniższy kod ilustruje tę technikę:

```
01  MY-MQ-CONSTANTS.
    COPY CMQV.
01  MY-OPEN-DATA.
    02 MY-MQOD.
       COPY CMQODV.
    02 MY-MQOR-TABLE OCCURS 100 TIMES.
       COPY CMQORV.
    MOVE MQOD-CURRENT-LENGTH TO MQOD-OBJECTRECOFFSET.
```

  Jednak działa to poprawnie tylko wtedy, gdy struktura MQOD i tablica rekordów MQOR są ciągłe; jeśli kompilator wstawia pomijanie bajtów między tabelą MQOD i tablicą MQOR, należy je dodać do wartości zapisanej w składowaniu *ObjectRecOffset*.

  Użycie produktu *ObjectRecOffset* jest zalecane w przypadku języków programowania, które nie obsługują typu danych wskaźnika, lub które implementują typ danych wskaźnika w sposób, który nie jest przenośny dla różnych środowisk (na przykład w języku programowania COBOL).

- Za pomocą pola wskaźnika *ObjectRecPtr*.

  W takim przypadku aplikacja może zadeklarować tablicę struktur MQOR niezależnie od struktury MQOD, a następnie ustawić wartość *ObjectRecPtr* na adres tablicy. Poniższy kod ilustruje tę technikę dla języka programowania C:

```
MQOD MyMqod;
MQOR MyMqor[100];
MyMqod.ObjectRecPtr = MyMqor;
```

  Użycie produktu *ObjectRecPtr* jest zalecane w przypadku języków programowania obsługujących typ danych wskaźnika w sposób przenośny dla różnych środowisk (na przykład język programowania w języku C).

Niezależnie od wybranej techniki, należy użyć jednego z produktów *ObjectRecOffset* i *ObjectRecPtr*; wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_RECORDS_ERROR, jeśli oba są zerowe, albo oba są niezerowe.

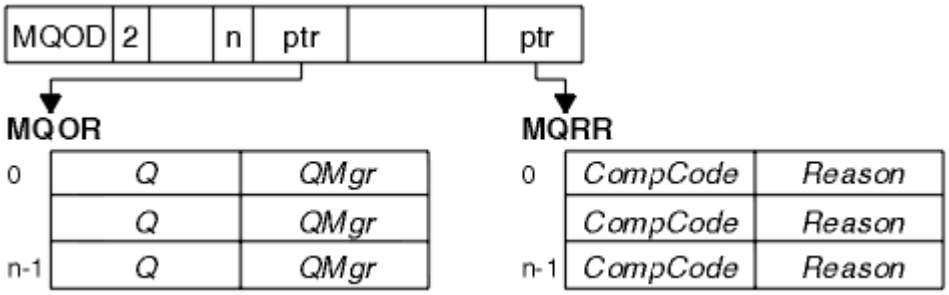
Korzystanie ze struktury MQRR

Te struktury są specyfiką miejsca docelowego; każdy rekord odpowiedzi zawiera pole *CompCode* i *Reason* dla każdej kolejki listy dystrybucyjnej. Tej struktury należy użyć, aby umożliwić rozróżnianie miejsc, w których występują problemy.

Jeśli na przykład zostanie wyświetlony kod przyczyny komendy MQRC_MULTIPLE_REASON, a lista dystrybucyjna zawiera pięć kolejek docelowych, nie będzie wiadomo, do których kolejek mają być stosowane problemy, jeśli ta struktura nie zostanie użyta. Jeśli jednak kod zakończenia i kod przyczyny dla każdego miejsca docelowego są dostępne, można łatwiej znaleźć błędy.

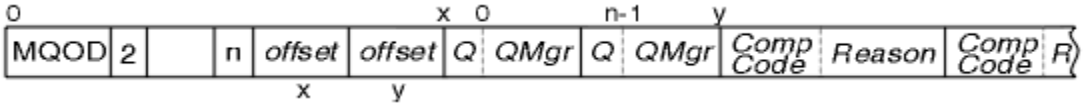
Więcej informacji na temat struktury MQRR zawiera sekcja MQRR .

Rysunek 30 na stronie 244 pokazuje, w jaki sposób można otworzyć listę dystrybucyjną w języku C.



Rysunek 30. Otwieranie listy dystrybucyjnej w języku C

Rysunek 31 na stronie 244 pokazuje, jak można otworzyć listę dystrybucyjną w języku COBOL.



Rysunek 31. Otwieranie listy dystrybucyjnej w języku COBOL

Korzystanie z opcji MQOPEN

Podczas otwierania listy dystrybucyjnej można określić następujące opcje:

- MQOO_OUTPUT
- MQOO_FAIL_IF_QUIESCING (opcjonalnie)
- MQOO_ALTERNATE_USER_AUTHORITY (opcjonalne)
- MQOO_ * _CONTEXT (opcjonalnie)

Opis tych opcji można znaleźć w sekcji “Otwieranie i zamykanie obiektów” na stronie 220 .

Umieszczanie komunikatów na liście dystrybucyjnej

Aby umieścić komunikaty na liście dystrybucyjnej, można użyć komendy MQPUT lub MQPUT1.

Jako dane wejściowe należy podać:

- Uchwyt połączenia (opis zawiera sekcja “Umieszczanie komunikatów w kolejce” na stronie 231).
- Uchwyt obiektu. Jeśli lista dystrybucyjna zostanie otwarta za pomocą komendy MQOPEN, program *Hobj* umożliwia tylko umieszczenie listy dystrybucyjnej na liście.
- Struktura deskryptora komunikatu (MQMD). Opis tej struktury można znaleźć w sekcji MQMD .

- Informacje sterujące w postaci struktury opcji komunikatu put (put-message option structure-MQPMO). Więcej informacji na temat wypełniania pól struktury MQPMO zawiera sekcja “Określanie opcji przy użyciu struktury MQPMO” na stronie 233 .
- Informacje sterujące w postaci rekordów umieszczania komunikatów (Put Message Records-MQPMR).
- Długość danych zawartych w komunikacie (MQLONG).
- Same dane komunikatu.

Dane wyjściowe:

- Kod zakończenia
- Kod przyczyny
- Rekordy odpowiedzi (opcjonalne)

Korzystanie ze struktury MQPMR

Ta struktura jest opcjonalna i udostępnia informacje specyficzne dla miejsca docelowego dla niektórych pól, które mogą być różne od tych, które zostały już zidentyfikowane w deskryptywnym MQMD.

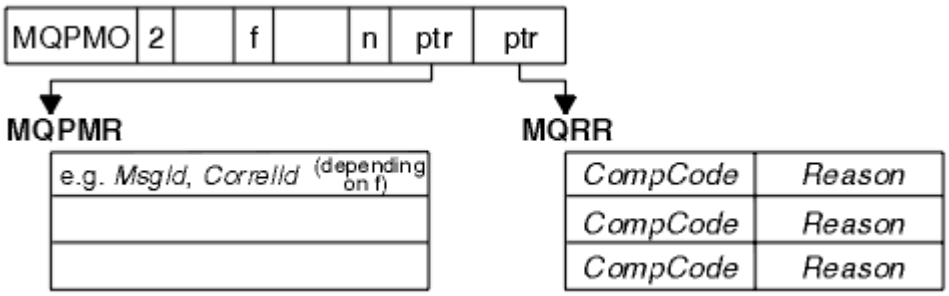
Opis tych pól znajduje się w sekcji MQPMR.

Zawartość każdego rekordu zależy od informacji podanych w polu *PutMsgRecFields* obiektu MQPMO. Na przykład w przykładowym programie AMQSPTL0.C (w sekcji “Przykładowy program listy dystrybucyjnej” na stronie 131 opis) pokazujący użycie list dystrybucyjnych, przykład ten wybiera wartości dla produktów *MsgId* i *CorrelId* w tabeli MQPMR. Ta sekcja przykładowego programu wygląda następująco:

```
typedef struct
{
MQBYTE24 MsgId;
MQBYTE24 CorrelId;
} PutMsgRec;
...
/**********************
MQLONG PutMsgRecFields=MQPMRF_MSG_ID | MQPMRF_CORREL_ID;
```

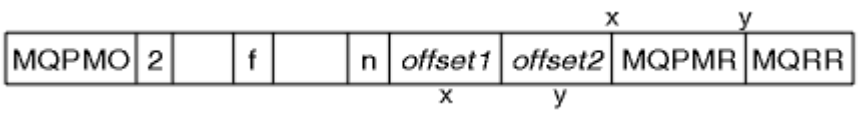
Oznacza to, że *MsgId* i *CorrelId* są dostępne dla każdego miejsca docelowego listy dystrybucyjnej. Rekordy umieszczania komunikatów są udostępniane jako tablica.

Rysunek 32 na stronie 245 pokazuje, w jaki sposób można umieścić komunikat w liście dystrybucyjnej w C.



Rysunek 32. Umieszczanie komunikatu na liście dystrybucyjnej w języku C

Program Rysunek 33 na stronie 245 pokazuje, jak można umieścić komunikat w liście dystrybucyjnej w języku COBOL.



Rysunek 33. Umieszczanie komunikatu na liście dystrybucyjnej w języku COBOL

Korzystanie z komendy MQPUT1

Jeśli używany jest protokół MQPUT1, należy wziąć pod uwagę następujące kwestie:

1. Wartości pól *ResponseRecOffset* i *ResponseRecPtr* muszą mieć wartość null lub zero.
2. Rekordy odpowiedzi, jeśli jest to wymagane, muszą być adresowane z tabeli MQOD.

Niektóre przypadki, w których wywołania put nie powiodły się

Jeśli niektóre atrybuty kolejki zostaną zmienione za pomocą opcji FORCE w komendzie w okresie między wywołaniem komendy MQOPEN i wywołaniem MQPUT, wywołanie MQPUT nie powiedzie się i zwróci kod przyczyny MQRC_OBJECT_CHANGED.

Menedżer kolejek oznacza, że uchwyt obiektu nie jest już poprawny. Dzieje się tak również wtedy, gdy zmiany są wprowadzane w czasie przetwarzania wywołania MQPUT1 lub gdy zmiany mają zastosowanie do każdej kolejki, do której nazwa kolejki jest tłumaczona. Atrybuty, które mają wpływ na uchwyt w ten sposób, są wymienione w opisie wywołania MQOPEN w produkcie MQOPEN. Jeśli wywołanie zwróci kod przyczyny MQRC_OBJECT_CHANGED, zamknij kolejkę, ponownie go otwórz, a następnie spróbuj ponownie umieścić komunikat.

Jeśli operacje put są zablokowane dla kolejki, w której podjęto próbę umieszczenia komunikatów (lub dowolnej kolejki, do której nazwa kolejki jest tłumaczona), wywołanie MQPUT lub MQPUT1 nie powiedzie się i zwróci kod przyczyny MQRC_PUT_INHIBITED. Może być możliwe pomyślne umieszczenie komunikatu w przypadku podjęcia próby połączenia w późniejszym czasie, jeśli projekt aplikacji jest taki, że inne programy regularnie zmieniają atrybuty kolejek.

Furthemore, jeśli kolejka, na której próbujesz umieścić komunikat, jest pełna, wywołanie MQPUT lub MQPUT1 nie powiedzie się i zwróci wartość MQRC_Q_FULL.

Jeśli usunięto kolejkę dynamiczną (tymczasową lub trwałą), wywołania MQPUT korzystające z wcześniej uzyskanego uchwytu obiektu nie powiodą się i zwróć kod przyczyny MQRC_Q_DELETED. W tej sytuacji dobrą praktyką jest zamknięcie uchwytu obiektu, ponieważ nie jest on już używany.

W przypadku list dystrybucyjnych w pojedynczym żądaniu może wystąpić wiele kodów zakończenia i kodów przyczyny. Nie można ich obsłużyć przy użyciu tylko pól wyjściowych *CompCode* i *Reason* w tabelach MQOPEN i MQPUT.

Jeśli listy dystrybucyjne są używane w celu umieszczania komunikatów w wielu miejscach docelowych, rekordy odpowiedzi zawierają określone *CompCode* i *Reason* dla każdego miejsca docelowego. Jeśli zostanie wyświetlony kod zakończenia MQCC_FAILED, żaden komunikat nie zostanie pomyślnie umieszczony w kolejce docelowej. Jeśli kod zakończenia to MQCC_WARNING, to komunikat zostanie pomyślnie umieszczony w jednej lub większej liczby kolejek docelowych. Jeśli zostanie wyświetlony kod powrotu z MQRC_MULTIPLE_POWODÓW, kody przyczyny nie są takie same dla każdego miejsca docelowego. Dlatego zaleca się użycie struktury MQRR, aby można było określić, która kolejka lub kolejki spowodowały błąd, a także przyczyny każdego z nich.

Pobieranie komunikatów z kolejki

Ta sekcja zawiera informacje na temat pobierania komunikatów z kolejki.

Komunikaty z kolejki można pobrać na dwa sposoby:

1. Można usunąć komunikat z kolejki, tak aby inne programy nie mogły go już oglądać.
2. Można skopiować wiadomość, pozostawiając oryginalny komunikat w kolejce. Jest to określane jako *przeglądanie*. Można usunąć ten komunikat po jego przeglądowaniu.

W obu przypadkach można użyć wywołania MQGET, ale najpierw aplikacja musi być połączona z menedżerem kolejek i należy użyć wywołania MQOPEN w celu otwarcia kolejki (dla danych wejściowych, przeglądania lub obu tych elementów). Operacje te są opisane w produktach “Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego” na stronie 211 i “Otwieranie i zamykanie obiektów” na stronie 220.

Po otwarciu kolejki można wielokrotnie korzystać z wywołania MQGET w celu przeglądania lub usuwania komunikatów w tej samej kolejce. Wywołaj komendę MQCLOSE po zakończeniu pobierania wszystkich komunikatów, które mają być wyświetlane w kolejce.

Użyj poniższych odsyłaczy, aby dowiedzieć się więcej na temat pobierania komunikatów z kolejki:

- "Pobieranie komunikatów z kolejki przy użyciu wywołania MQGET" na stronie 247
- "Kolejność, w jakiej komunikaty są pobierane z kolejki" na stronie 251
- "Uzyskiwanie określonego komunikatu" na stronie 264
- "Zwiększanie wydajności nietrwałych komunikatów" na stronie 265
- "Obsługa komunikatów o długości większej niż 4 MB" na stronie 269
- "Oczekiwanie na komunikaty" na stronie 275
-
- "Pomijanie wycofania" na stronie 275
- "Konwersja danych aplikacji" na stronie 278
- "Przeglądanie komunikatów w kolejce" na stronie 279
- "Niektóre przypadki, w których wywołanie MQGET nie powiodło się" na stronie 285

Pojęcia pokrewne

"Interfejs kolejki komunikatów-przegląd" na stronie 199
Informacje na temat komponentów interfejsu kolejek komunikatów (Message Queue Interface-MQI).

"Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego" na stronie 211
Aby można było korzystać z usług programistycznych produktu WebSphere MQ , program musi mieć połączenie z menedżerem kolejek. Informacje zawarte w tej sekcji umożliwiają zapoznanie się z informacjami na temat nawiązywania połączenia z menedżerem kolejek i z jego rozłączeniem.

"Otwieranie i zamykanie obiektów" na stronie 220
Te informacje udostępniają wgląd w otwieranie i zamykanie obiektów WebSphere MQ .

"Umieszczanie komunikatów w kolejce" na stronie 231
Te informacje umożliwiają zapoznanie się z informacjami na temat umieszczania komunikatów w kolejce.

"Sprawdzanie i ustawianie atrybutów obiektu" na stronie 328
Atrybuty to właściwości, które definiują parametry obiektu WebSphere MQ .

"Zatwierdzanie i wycofywanie jednostek pracy" na stronie 331
W tej sekcji opisano sposób zatwierdzania i tworzenia kopii zapasowych wszystkich możliwych do odtworzenia operacji get i put, które wystąpiły w jednostce pracy.

"Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy" na stronie 338
Informacje na temat wyzwalaczy i sposobów uruchamiania aplikacji IBM WebSphere MQ przy użyciu wyzwalaczy.

"Praca z interfejsem MQI i klastrami" na stronie 355
Istnieją specjalne opcje dotyczące wywołań i kodów powrotu, które odnoszą się do technologii klastrowej.

Pobieranie komunikatów z kolejki przy użyciu wywołania MQGET

Wywołanie MQGET pobiera komunikat z otwartej kolejki lokalnej. Nie może on pobrać komunikatu z kolejki w innym systemie.

Jako dane wejściowe wywołania MQGET należy podać:

- Uchwyt połączenia.
- Uchwyt kolejki.
- Opis komunikatu, który ma zostać zwrócony z kolejki. Ma to postać struktury deskryptora komunikatu (MQMD).
- Sterowanie informacjami w postaci struktury MQGMO (Get Message Options).
- Wielkość buforu, który został przypisany do przechowywania komunikatu (MQLONG).

- Adres pamięci masowej, w której ma zostać umieszczony komunikat.

Dane wyjściowe komendy MQGET są następujące:

- Kod przyczyny
- Kod zakończenia
- Komunikat w określonym obszarze buforu, jeśli wywołanie zakończy się pomyślnie.
- Struktura opcji została zmodyfikowana w celu wyświetlenia nazwy kolejki, z której został pobrany komunikat.
- Struktura deskryptora komunikatu wraz z zawartością pól zmodyfikowanych w celu opisania komunikatu, który został pobrany
- Długość komunikatu (MQLONG)

W tabeli MQGETznajduje się opis wywołania MQGET.

W poniższych sekcjach opisano informacje, które należy podać jako dane wejściowe dla wywołania MQGET.

- “Określanie uchwytów połączeń” na stronie 248
- “Opisywanie komunikatów przy użyciu struktury MQMD i wywołania MQGET” na stronie 248
- “Określanie opcji MQGET przy użyciu struktury MQGMO” na stronie 249
- “Określanie wielkości obszaru buforu” na stronie 251

Określanie uchwytów połączeń

W przypadku programu CICS w aplikacjach z/OS można określić stałą MQHC_DEF_HCONN (która ma wartość zero), lub użyć uchwytu połączenia zwróconego przez wywołanie MQCONN lub MQCONNX. W przypadku innych aplikacji zawsze należy używać uchwytu połączenia zwróconego przez wywołanie MQCONN lub MQCONNX.

Użyj uchwytu kolejki (*Hobj*), który jest zwracany w przypadku wywołania komendy MQOPEN.

Opisywanie komunikatów przy użyciu struktury MQMD i wywołania MQGET

Aby zidentyfikować komunikat, który ma zostać wygenerowany z kolejki, należy użyć struktury deskryptora komunikatu (MQMD).

Jest to parametr wejściowy/wyjściowy dla wywołania MQGET. Istnieje wprowadzenie do właściwości komunikatu, które zawiera opis MQMD w produkcie “Komunikaty produktu IBM WebSphere MQ” na stronie 10, a także opis samej struktury w produkcie MQMD.

Jeśli wiadomo, który komunikat ma zostać wygenerowany z kolejki, należy zapoznać się z “Uzyskiwanie określonego komunikatu” na stronie 264.

Jeśli konkretna wiadomość nie zostanie określona, komenda MQGET pobiera komunikat *pierwszy* w kolejce. “Kolejność, w jakiej komunikaty są pobierane z kolejki” na stronie 251 opisuje sposób, w jaki priorytet komunikatu, atrybut *MsgDeliverySequence* kolejki oraz opcja MQGMO_LOGICAL_ORDER określają kolejność komunikatów w kolejce.

Uwaga: Aby użyć komendy MQGET więcej niż raz (na przykład w celu wykonania kroku przez komunikaty w kolejce), po każdym wywołaniu należy ustawić pola *MsgId* i *CorrelId* tej struktury na wartość NULL. Powoduje to wyczyszczenie tych pól identyfikatorów pobranych komunikatów.

Jeśli jednak chcesz zgrupować swoje wiadomości, *GroupId* musi być taka sama dla wiadomości w tej samej grupie, tak aby wywołanie wyszukało komunikat o tych samych identyfikatorach co poprzedni komunikat w celu wykonania całej grupy.

Określanie opcji MQGET przy użyciu struktury MQGMO

Struktura MQGMO jest zmienną wejścia/wyjścia dla przekazywania opcji do wywołania MQGET. W poniższych sekcjach można wykonać niektóre z pól tej struktury.

W produkcie MQGMOznajduje się opis struktury MQGMO.

StrucId
StrucId jest 4-znakowym polem używanym do identyfikowania struktury jako struktury opcji komunikatów get. Zawsze należy podać identyfikator MQGMO_STRUC_ID.

Version
*Version*Opisuje numer wersji struktury. MQGMO_VERSION_1 jest wartością domyślną. Aby użyć pól w wersji 2 lub pobrać komunikaty w kolejności logicznej, należy określić wartość MQGMO_VERSION_2. Aby użyć pól w wersji 3 lub pobrać komunikaty w kolejności logicznej, należy określić wartość MQGMO_VERSION_3. MQGMO_CURRENT_VERSION ustawia aplikację w taki sposób, aby była używana na ostatnim poziomie.

Options
W obrębie kodu można wybrać opcje w dowolnej kolejności; każda opcja jest reprezentowana przez bit w polu *Options* .

Pole *Options* steruje:

- Określa, czy wywołanie MQGET oczekuje na dotarcie komunikatu do kolejki przed jej zakończeniem (patrz “Oczekiwanie na komunikaty” na stronie 275)
- Informacja o tym, czy operacja pobierania jest uwzględniona w jednostce pracy.
- Określa, czy nietrwały komunikat jest pobierany poza punktem synchronizacji, co pozwala na szybkie przesyłanie komunikatów.
- W produkcie WebSphere MQ for z/OS, czy pobrany komunikat jest oznaczony jako pomijanie wycofania (patrz sekcja “Pomijanie wycofania” na stronie 275)
- Informacja o tym, czy komunikat jest usuwany z kolejki, czy tylko przeglądane
- Czy wybrać komunikat przy użyciu kursora przeglądania lub innego kryterium wyboru
- Określa, czy wywołanie powiedzie się nawet wtedy, gdy komunikat jest dłuższy niż bufor.
- W produkcie WebSphere MQ for z/OS, określ, czy zezwolić na zakończenie połączenia. Ta opcja ustawia również sygnał wskazujący, że użytkownik chce być powiadamiany o nadejściu komunikatu.
- Określa, czy wywołanie nie powiedzie się, jeśli menedżer kolejek jest w stanie wygaszania.
- W produkcie WebSphere MQ for z/OS, czy wywołanie nie powiedzie się, jeśli połączenie jest w stanie wygaszania.
- Określa, czy wymagana jest konwersja danych komunikatu aplikacji (patrz sekcja “Konwersja danych aplikacji” na stronie 278)
- Kolejność, w jakiej komunikaty i (z wyjątkiem produktu WebSphere MQ for z/OS) są pobierane z kolejki.
- Z wyjątkiem WebSphere MQ for z/OS, bez względu na to, czy są kompletne, komunikaty logiczne są dostępne tylko do pobrania.
- Określa, czy komunikaty w grupie mogą być pobierane tylko w przypadku, gdy dostępne są *wszystkie* komunikaty w grupie.
- Z wyjątkiem produktu WebSphere MQ for z/OS, czy segmenty w komunikacie logicznym mogą być pobierane tylko w przypadku, gdy dostępne są *wszystkie* segmenty w komunikacie logicznym.

Jeśli pole *Options* zostanie pozostaw ustawione na wartość domyślną (MQGMO_NO_WAIT), wywołanie MQGET będzie działać w ten sposób:

- Jeśli nie ma komunikatu zgodnego z kryteriami wyboru w kolejce, wywołanie nie czeka na nadejście komunikatu, ale zostanie zakończone natychmiast. Ponadto w produkcie WebSphere MQ for z/OSwywołanie nie ustawia sygnału żądającego powiadomienia, gdy pojawi się taki komunikat.
- Sposób, w jaki połączenie działa z punktami synchronizacji jest określany przez platformę:

Platforma	Pod kontrolą punktu synchronizacji
IBM i	Nie
Systemy UNIX and Linux	Nie
z/OS	Tak
Systemy Windows	Nie

- W produkcie WebSphere MQ for z/OSpobrany komunikat nie jest oznaczany jako pominięcie wycofania.
- Wybrany komunikat jest usuwany z kolejki (nie jest przeglądane).
- Konwersja danych komunikatu aplikacji nie jest wymagana.
- Wywołanie nie powiedzie się, jeśli komunikat jest dłuższy niż bufor.

WaitInterval
Pole *WaitInterval* określa maksymalny czas (w milisekundach), przez jaki wywołanie MQGET oczekuje na dotarcie komunikatu do kolejki w przypadku użycia opcji MQGMO_WAIT. Jeśli w czasie określonym w *WaitInterval*nie pojawi się żaden komunikat, wywołanie zakończy się i zwróci kod przyczyny wskazujący, że nie było komunikatu zgodnego z kryteriami wyboru w kolejce.

W produkcie WebSphere MQ for z/OS, jeśli używana jest opcja MQGMO_SET_SIGNAL, pole *WaitInterval* określa czas, dla którego sygnał jest ustawiany.

Więcej informacji na temat tych opcji można znaleźć w sekcji “Oczekiwanie na komunikaty” na stronie 275.

Signal1
Element **Signal1 jest obsługiwany w produkcie WebSphere MQ dla systemów z/OS i MQSeries tylko dla produktu HP Integrity NonStop Server**.

Jeśli opcja MQGMO_SET_SIGNAL jest używana do żądania powiadomienia o aplikacji po nadejściu odpowiedniego komunikatu, należy określić typ sygnału w polu *Signal1* (Typ sygnału). W produkcie WebSphere MQ na wszystkich innych platformach pole *Signal1* jest zastrzeżone, a jego wartość nie jest znacząca.

Signal2
Pole *Signal2* jest zarezerwowane na wszystkich platformach, a jego wartość nie jest znacząca.

ResolvedQName
ResolvedQName to pole wyjściowe, w którym menedżer kolejek zwraca nazwę kolejki (po rozstrzygnięcie dowolnego aliasu), z której został pobrany komunikat.

MatchOptions
Produkt *MatchOptions* steruje kryteriami wyboru dla komendy MQGET.

GroupStatus
GroupStatus wskazuje, czy pobrany komunikat znajduje się w grupie.

SegmentStatus
SegmentStatus wskazuje, czy pobrany element jest segmentem komunikatu logicznego.

Segmentation
Segmentation wskazuje, czy segmentacja jest dozwolona dla pobranego komunikatu.

MsgToken

MsgToken Jednoznacznie identyfikuje komunikat.

ReturnedLength
ReturnedLength to pole wyjściowe, w którym menedżer kolejek zwraca długość zwracanych danych komunikatu (w bajtach).

MsgHandle
Uchwyt do komunikatu, który ma zostać zapełniony właściwościami komunikatu pobieranego z kolejki. Uchwyt został wcześniej utworzony za pomocą wywołania MQCRTMH. Wszystkie właściwości, które są już powiązane z uchwytem, są czyszczone przed pobraniem komunikatu.

Określanie wielkości obszaru buforu

W parametrze *BufferLength* wywołania MQGET określ wielkość obszaru buforu, w którym mają być przechowywane pobierane dane komunikatu. Decydujesz, jak duże powinno być to na trzy sposoby:

1. Użytkownik może już wiedzieć, jaka długość komunikatów ma być oczekiwana w tym programie. Jeśli tak, należy określić bufor o tej wielkości.

   Można jednak użyć opcji MQGMO_ACCEPT_TRUNCATED_MSG w strukturze MQGMO, jeśli wywołanie MQGET ma być zakończone nawet wtedy, gdy komunikat jest zbyt duży dla buforu. W tym przypadku:

   - Bufor jest wypełniany tak dużą, jak i może być wstrzymana
   - Wywołanie zwraca kod zakończenia ostrzeżenia
   - Komunikat jest usuwany z kolejki (odrzucaj pozostałą część komunikatu) lub kursor przeglądania jest zaawansowany (jeśli przeglądasz kolejkę)
   - Rzeczywista długość komunikatu jest zwracana w produkcie *DataLength* .

   Bez tej opcji wywołanie nadal kończy się ostrzeżeniem, ale nie usuwa komunikatu z kolejki (lub wyprzedzają kursor przeglądania).
2. Oszacuj wielkość buforu (lub nawet podaj wielkość zero bajtów), a *nie* należy użyć opcji MQGMO_ACCEPT_TRUNCATED_MSG. Jeśli wywołanie MQGET nie powiedzie się (na przykład, ponieważ bufor jest za mały), długość komunikatu jest zwracana w parametrze *DataLength* wywołania. (Bufor jest nadal zapełniany tak, jak większość komunikatu może być wstrzymana, ale przetwarzanie wywołania nie zostało zakończone.) Zapisz *MsgId* tego komunikatu, a następnie powtórz wywołanie MQGET, określając obszar buforu o prawidłowej wielkości, a *MsgId* , który został odnotowane od pierwszego wywołania.

   Jeśli program obsługuje kolejkę, która jest również obsługiwana przez inne programy, jeden z tych innych programów może usunąć komunikat, który ma zostać usunięty, zanim program będzie mógł wydać kolejne wywołanie MQGET. Program może tracić czas na wyszukiwanie wiadomości, która już nie istnieje. Aby tego uniknąć, należy najpierw przeglądać kolejkę aż do znalezienia komunikatu, określając wartość *BufferLength* równą zero i używając opcji MQGMO_ACCEPT_TRUNCATED_MSG. Spowoduje to umieszczenie kursora w komunikacie, który ma zostać wyświetlony. Następnie można pobrać komunikat, wywołując komendę MQGET ponownie, podając opcję MQGMO_MSG_UNDER_CURSOR. Jeśli inny program usunie komunikat między wywołaniami przeglądania i usuwania, druga operacja MQGET nie powiedzie się natychmiast (bez przeszukiwania całej kolejki), ponieważ pod kursorem przeglądania nie ma żadnego komunikatu.
3. Atrybut *MaxMsgLength kolejka* określa maksymalną długość komunikatów akceptowanych dla tej kolejki. Atrybut *MaxMsgLength menedżer kolejek* określa maksymalną długość komunikatów akceptowanych dla tego menedżera kolejek. Jeśli nie wiadomo, jaka długość komunikatu jest oczekiwana, można uzyskać informacje na temat atrybutu *MaxMsgLength* (za pomocą wywołania MQINQ), a następnie określić bufor o tej wielkości.

   Aby uniknąć zmniejszonej wydajności, należy podjąć próbę jak najbliżej rzeczywistej wielkości buforu.

   Więcej informacji na temat atrybutu *MaxMsgLength* zawiera sekcja “Zwiększanie maksymalnej długości komunikatu” na stronie 269.

Kolejność, w jakiej komunikaty są pobierane z kolejki

Użytkownik może sterować kolejką, w której pobierane są komunikaty z kolejki. Ta sekcja zawiera informacje na temat opcji.

Priorytet

Program może przypisać priorytet do komunikatu, gdy umieszcza komunikat w kolejce (patrz “Priorytety komunikatów” na stronie 17). Komunikaty o równym priorytecie są przechowywane w kolejce w celu ich przybycia, a nie kolejności, w jakiej są zatwierdzane.

Menedżer kolejek przechowuje kolejki w ścisłej sekwencji FIFO (najpierw w pierwszej kolejności) lub w FIFO w kolejności priorytetów. Zależy to od ustawienia atrybutu *MsgDeliverySequence* kolejki.Po nadejściu komunikatu do kolejki jest on wstawiany bezpośrednio po ostatnim komunikacie, który ma ten sam priorytet.

Programy mogą otrzymać pierwszy komunikat z kolejki lub mogą otrzymać określony komunikat z kolejki, ignorując priorytet tych komunikatów. Program może na przykład chcieć przetworzyć odpowiedź na konkretny komunikat, który został przesłany wcześniej. Więcej informacji na ten temat zawiera sekcja “Uzyskiwanie określonego komunikatu” na stronie 264.

Jeśli aplikacja umieszcza sekwencję komunikatów w kolejce, inna aplikacja może pobrać te komunikaty w tej samej kolejności, w jakiej zostały umieszczone, pod warunkiem, że:

- Wszystkie komunikaty mają ten sam priorytet
- Wszystkie komunikaty zostały umieszczone w tej samej jednostce pracy lub zostały umieszczone na zewnątrz jednostki pracy.
- Kolejka jest lokalna względem umieszczonej aplikacji

Jeśli warunki te nie są spełnione, a aplikacje są zależne od komunikatów pobieranych w określonej kolejności, aplikacje muszą zawierać informacje o kolejności w danych komunikatu lub ustanowić sposób potwierdzania odbioru komunikatu przed wysłaniem następnego komunikatu.

Uporządkowanie logiczne i fizyczne

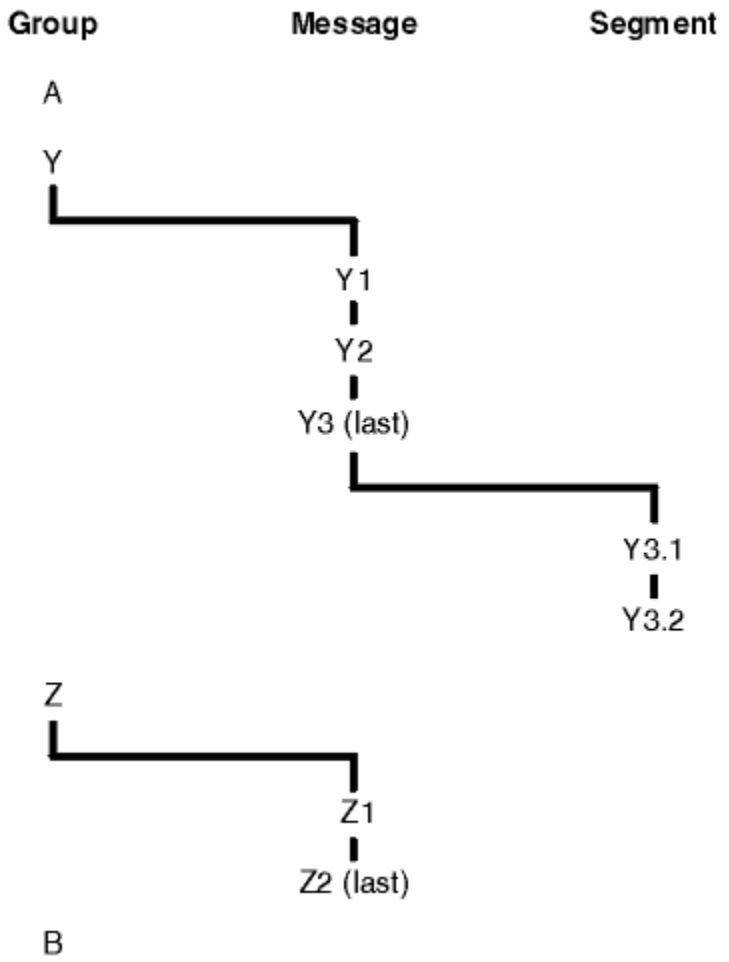
Komunikaty w kolejkach mogą występować (w obrębie każdego poziomu priorytetu) w kolejności *fizycznej* lub *logicznej* .

Porządek fizyczny jest to kolejność, w jakiej komunikaty docierają do kolejki. Kolejność logiczna polega na tym, że wszystkie komunikaty i segmenty w grupie znajdują się w ich sekwencji logicznej, obok siebie, w pozycji określonej przez fizyczne położenie pierwszej pozycji należącej do grupy.

Opis grup, komunikatów i segmentów zawiera sekcja “Grupy komunikatów” na stronie 36. Te zlecenia fizyczne i logiczne mogą się różnić, ponieważ:

- Grupy mogą przybyć do miejsca docelowego w podobnych sytuacjach z różnych aplikacji, a tym samym utracić dowolny odrębny porządek fizyczny.
- Nawet w obrębie jednej grupy komunikaty mogą być wysyłane poza kolejną kolejką, ponieważ są one przekierowujące lub opóźnione w przypadku niektórych komunikatów w grupie.

Na przykład kolejność logiczna może wyglądać na rysunku Rysunek 34 na stronie 253:

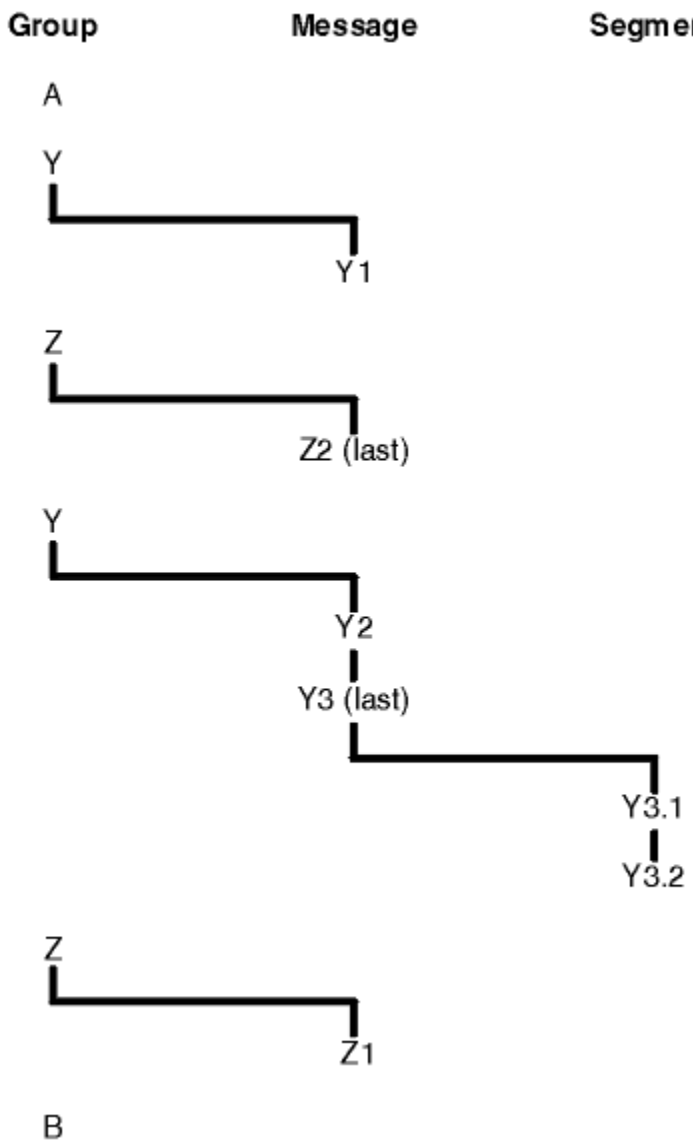


Rysunek 34. Porządek logiczny w kolejce

Komunikaty te mogą wystąpić w następującej kolejności logicznej w kolejce:

1. Komunikat A (nie w grupie)
2. Komunikat logiczny 1 grupy Y
3. Komunikat logiczny 2 grupy Y
4. Segment 1 of (ostatni) komunikat logiczny 3 grupy Y
5. (Ostatni) segment 2 z (ostatniego) komunikatu logicznego 3 grupy Y
6. Komunikat logiczny 1 grupy Z
7. (Ostatni) komunikat logiczny 2 grupy Z
8. Komunikat B (nie znajduje się w grupie)

Porządek fizyczny może być jednak zupełnie inny. Pozycja fizyczna elementu *first* w każdej grupie określa pozycję logiczną całej grupy. Na przykład, jeśli grupy Y i Z dotarły do podobnych czasów, a komunikat 2 grupy Z wyprzedzał komunikat 1 z tej samej grupy, porządek fizyczny będzie wyglądał podobnie do rysunku Rysunek 35 na stronie 254:



Rysunek 35. Porządek fizyczny w kolejce

Komunikaty te pojawiają się w następującej kolejności fizycznej w kolejce:

1. Komunikat A (nie w grupie)
2. Komunikat logiczny 1 grupy Y
3. Komunikat logiczny 2 grupy Z
4. Komunikat logiczny 2 grupy Y
5. Segment 1 of (ostatni) komunikat logiczny 3 grupy Y
6. (Ostatni) segment 2 z (ostatniego) komunikatu logicznego 3 grupy Y
7. Komunikat logiczny 1 grupy Z
8. Komunikat B (nie znajduje się w grupie)

Uwaga: W systemie IBM WebSphere MQ for z/OSfizyczna kolejność komunikatów w kolejce nie jest gwarantowana, jeśli kolejka jest indeksowana przez GROUPID.

Podczas pobierania komunikatów można określić MQGMO_LOGICAL_ORDER, aby pobierać komunikaty w porządku logicznym, a nie w porządku fizycznym.

Jeśli wywoła wywołanie MQGET z MQGMO_BROWSE_FIRST i MQGMO_LOGICAL_ORDER, kolejne wywołania MQGET z MQGMO_BROWSE_NEXT muszą również określać parametr

MQGMO_LOGICAL_ORDER. I odwrotnie, jeśli MQGET z MQGMO_BROWSE_FIRST nie określa MQGMO_LOGICAL_ORDER, nie należy podawać następujących operacji MQGETs z MQGMO_BROWSE_NEXT.

Informacje o grupach i segmentach, które menedżer kolejek zachowuje dla wywołań MQGET, które przeglądną komunikaty w kolejce, są oddzielone od informacji o grupach i segmentach, które menedżer kolejek zachowuje dla wywołań MQGET, które usuwają komunikaty z kolejki. Jeśli zostanie określona wartość MQGMO_BROWSE_FIRST, menedżer kolejek ignoruje informacje o grupie i segmencie w celu przeglądania, a następnie skanuje kolejkę tak, jakby nie było żadnej bieżącej grupy i nie ma bieżącego komunikatu logicznego.

Uwaga: Nie należy używać wywołania MQGET w celu przeglądania *poza końcem* grupy komunikatów (lub komunikatu logicznego nie w grupie) bez określania parametru MQGMO_LOGICAL_ORDER. Na przykład, jeśli ostatni komunikat w grupie *poprzedza* pierwszy komunikat w grupie w kolejce, za pomocą komendy MQGMO_BROWSE_NEXT w celu przejrzenia poza koniec grupy, podanie wartości MQMO_MATCH_MSG_SEQ_NUMBER z parametrem *MsgSeqNumber* ustawionym na 1 (aby znaleźć pierwszy komunikat następnej grupy) ponownie zwróci pierwszy komunikat w grupie, który został już przeglądane. Może to nastąpić natychmiast lub kilka wywołań MQGET w późniejszym czasie (jeśli istnieją grupy, które się do tego zdarzają).

Aby uniknąć możliwości pętli nieskończonej, należy otworzyć kolejkę *dwukrotnie* w celu wyszukania:

- Użyj pierwszego uchwytu, aby przeglądać tylko pierwszy komunikat w każdej grupie.
- Użyj drugiego uchwytu, aby przeglądać tylko komunikaty należące do określonej grupy.
- Użyj opcji MQMO_ *, aby przenieść drugi kursor przeglądania na pozycję pierwszego kursora przeglądania, przed przeglądaniem komunikatów w grupie.
- Nie należy używać funkcji przeglądania MQGMO_BROWSE_NEXT poza końcem grupy.

Więcej informacji na ten temat można znaleźć w sekcji MQGET, MQMDi Reguły sprawdzania poprawności opcji MQI.

W przypadku większości aplikacji można wybrać porządkowanie logiczne lub fizyczne podczas przeglądania. Jeśli jednak użytkownik chce przełączać się między tymi trybami, należy pamiętać, że podczas pierwszego wydania przeglądania za pomocą komendy MQGMO_LOGICAL_ORDER, położenie w sekwencji logicznej jest ustanawiane.

Jeśli pierwszy element w grupie nie jest obecny w tym czasie, grupa, w której użytkownik nie jest w stanie, nie jest uznawana za część sekwencji logicznej.

Gdy kursor przeglądania znajduje się w grupie, może on być kontynuowany w obrębie tej samej grupy, nawet jeśli pierwszy komunikat zostanie usunięty. Początkowo nigdy nie można przenieść do grupy za pomocą komendy MQGMO_LOGICAL_ORDER, w której pierwszy element nie jest obecny.

MQPMO_LOGICAL_ORDER

Opcja MQPMO informuje menedżera kolejek o tym, w jaki sposób aplikacja umieszcza komunikaty w grupach i segmentach komunikatów logicznych. Można ją określić tylko w wywołaniu MQPUT. Nie jest ona poprawna w wywołaniu metody MQPUT1 .

Jeśli zostanie określona opcja MQPMO_LOGICAL_ORDER, oznacza to, że aplikacja używa kolejnych wywołań MQPUT w następujących celach:

1. Umieszczenie segmentów w każdym komunikacie logicznym w kolejności rosnącego przesunięcia segmentu, począwszy od 0, bez przerw.
2. Umieszczenie wszystkich segmentów w jednym komunikacie logicznym przed umieszczeniem segmentów w następnym komunikacie logicznym.
3. Umieszczenie komunikatów logicznych w każdej grupie komunikatów w kolejności rosnących numerów kolejnych komunikatów, począwszy od 1, bez przerw.IBM WebSphere MQ automatycznie zwiększa numer kolejny komunikatu.
4. Umieszczenie wszystkich komunikatów logicznych w jednej grupie komunikatów przed umieszczeniem komunikatów logicznych w następnej grupie komunikatów.

Ponieważ aplikacja opowiedziała menedżerowi kolejek, w jaki sposób umieszcza komunikaty w grupach i segmentach komunikatów logicznych, aplikacja nie musi obsługiwać i aktualizować informacji o grupach i segmentach na temat poszczególnych wywołań MQPUT, ponieważ menedżer kolejek przechowuje i aktualizuje te informacje. W szczególności oznacza to, że aplikacja nie musi ustawiać pól *GroupId*, *MsgSeqNumber*i *Offset* w strukturze MQMD, ponieważ menedżer kolejek ustawia te pola na odpowiednie wartości. Aplikacja musi ustawić pole *MsgFlags* tylko w deskryptywie MQMD, aby wskazać, kiedy komunikaty należą do grup lub są segmentami komunikatów logicznych, a także aby wskazać ostatni komunikat w grupie lub ostatnim segmencie komunikatu logicznego.

Po uruchomieniu grupy komunikatów lub komunikatu logicznego kolejne wywołania MQPUT muszą określać odpowiednie opcje MQMF_ * w produkcie *MsgFlags* w strukturze MQMD. Jeśli aplikacja podejmie próbę umieszczenia komunikatu, który nie znajduje się w grupie, gdy istnieje niezakończona grupa komunikatów lub komunikat, który nie jest segmentem, jeśli istnieje niezakończony komunikat logiczny, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_INCOMPLETE_GROUP lub MQRC_INCOMPLETE_MSG, w zależności od przypadku. Menedżer kolejek zachowuje jednak informacje na temat bieżącej grupy komunikatów lub bieżącego komunikatu logicznego, a aplikacja może je zakończyć, wysyłając komunikat (prawdopodobnie bez danych komunikatu aplikacji), określając odpowiednio MQMF_LAST_MSG_IN_GROUP lub MQMF_LAST_SEGMENT, przed ponownym wywołaniem wywołania MQPUT w celu umieszczenia komunikatu, który nie znajduje się w grupie, albo nie jest to segment.

Rysunek 35 na stronie 254 przedstawia kombinacje opcji i flag, które są poprawne, oraz wartości pól *GroupId*, *MsgSeqNumber*i *Offset* używanych przez menedżer kolejek w każdym przypadku. Kombinacje opcji i opcji, które nie są wyświetlane w tabeli, są niepoprawne. Kolumny w tabeli mają następujące znaczenia: albo Tak, albo Nie:

PROTOKÓŁ ORD
Określa, czy opcja MQPMO_LOGICAL_ORDER jest określona w wywołaniu.

MIG
Określa, czy w wywołaniu jest określona opcja MQMF_MSG_IN_GROUP lub MQMF_LAST_MSG_IN_GROUP.

SEG
Określa, czy w wywołaniu jest określona opcja MQMF_SEGMENT lub MQMF_LAST_SEGMENT.

SEG OK
Określa, czy opcja MQMF_SEGMENTATION_ALLOWED jest określona w wywołaniu.

Cur grp
Określa, czy bieżąca grupa komunikatów istnieje przed wywołaniem.

Komunikat dziennika Cur
Określa, czy bieżący komunikat logiczny istnieje przed wywołaniem.

Pozostałe kolumny
Pokaż wartości używane przez menedżer kolejek. Poprzednio oznaczono wartość użytą dla pola w poprzednim komunikacie dla uchwytu kolejki.

Tabela 36. Opcje MQPUT odnoszące się do komunikatów w grupach i segmentach komunikatów logicznych

Opcje określone przez użytkownika				**Status grupy i dziennika-komunikat przed wywołaniem**		**Wartości, których używa menedżer kolejek**		
PROTOKÓŁ ORD	**MIG**	**SEG**	**SEG OK**	**Grp**	**Komunikat dziennika Cur**	***GroupId***	***MsgSeqNumber***	***Offset***
Tak	Nie	Nie	Nie	Nie	Nie	MQGI_NONE	1	0
Tak	Nie	Nie	Tak	Nie	Nie	Identyfikator nowej grupy	1	0
Tak	Nie	Tak	Albo	Nie	Nie	Identyfikator nowej grupy	1	0
Tak	Nie	Tak	Albo	Nie	Tak	Poprzedni identyfikator grupy	1	Poprzednie przesunięcie + długość poprzedniego segmentu
Tak	Tak	Albo	Albo	Nie	Nie	Identyfikator nowej grupy	1	0
Tak	Tak	Albo	Albo	Tak	Nie	Poprzedni identyfikator grupy	Poprzedni numer kolejny + 1	0
Tak	Tak	Tak	Albo	Tak	Tak	Poprzedni identyfikator grupy	Poprzedni numer kolejny	Poprzednie przesunięcie + długość poprzedniego segmentu
Nie	Nie	Nie	Nie	Albo	Albo	MQGI_NONE	1	0
Nie	Nie	Nie	Tak	Albo	Albo	Identyfikator nowej grupy, jeśli wartość MQGI_NONE, wartość else w polu	1	0
Nie	Nie	Tak	Albo	Albo	Albo	Identyfikator nowej grupy, jeśli wartość MQGI_NONE, wartość else w polu	1	Wartość w polu
Nie	Tak	Nie	Albo	Albo	Albo	Identyfikator nowej grupy, jeśli wartość MQGI_NONE, wartość else w polu	Wartość w polu	0

Tabela 36. Opcje MQPUT odnoszące się do komunikatów w grupach i segmentach komunikatów logicznych (kontynuacja)

Opcje określone przez użytkownika				**Status grupy i dziennika-komunikat przed wywołaniem**		**Wartości, których używa menedżer kolejek**		
PROTOKÓŁ ORD	**MIG**	**SEG**	**SEG OK**	**Grp**	**Komunikat dziennika Cur**	***GroupId***	***MsgSeqNumber***	***Offset***
Nie	Tak	Tak	Albo	Albo	Albo	Identyfikator nowej grupy, jeśli wartość MQGI_NONE, wartość else w polu	Wartość w polu	Wartość w polu

Uwaga:

- Parametr MQPMO_LOGICAL_ORDER nie jest poprawny w wywołaniu MQPUT1 .
- W przypadku pola *MsgId* menedżer kolejek generuje nowy identyfikator komunikatu, jeśli określono wartość MQPMO_NEW_MSG_ID lub MQMI_NONE, a w przeciwnym razie używa wartości w polu.
- W przypadku pola *CorrelId* menedżer kolejek generuje nowy identyfikator korelacji, jeśli określono wartość MQPMO_NEW_CORREL_ID, a w przeciwnym razie używa wartości w polu.

Jeśli określono parametr MQPMO_LOGICAL_ORDER, menedżer kolejek wymaga, aby wszystkie komunikaty w grupie i segmentach w komunikacie logicznym były umieszczane z taką samą wartością w polu *Persistence* w strukturze MQMD, co oznacza, że wszystkie muszą być trwałe lub wszystkie muszą być nietrwałe. Jeśli ten warunek nie jest spełniony, wywołanie MQPUT nie powiedzie się z kodem przyczyny MQRC_INCONSISTENT_PERSISTENCE.

Opcja MQPMO_LOGICAL_ORDER wpływa na jednostki pracy w następujący sposób:

- Jeśli pierwszy komunikat fizyczny w grupie lub komunikat logiczny jest umieszczany w jednostce pracy, wszystkie pozostałe komunikaty fizyczne w grupie lub komunikacie logicznym muszą być umieszczone w jednostce pracy, o ile ten sam uchwyt kolejki jest używany. Nie muszą one jednak być umieszczane w tej samej jednostce pracy, co pozwala na dzielenie grupy komunikatów lub komunikatów logicznych składającej się z wielu komunikatów fizycznych na dwie lub więcej kolejnych jednostek pracy dla uchwytu kolejki.
- Jeśli pierwszy komunikat fizyczny w grupie lub komunikacie logicznym nie jest umieszczany w jednostce pracy, żaden inny komunikat fizyczny w grupie lub komunikat logiczny nie może zostać umieszczony w jednostce pracy, jeśli ten sam uchwyt kolejki jest używany.

Jeśli te warunki nie są spełnione, wywołanie MQPUT kończy się niepowodzeniem z kodem przyczyny MQRC_INCONSISTENT_UOW.

Jeśli określono parametr MQPMO_LOGICAL_ORDER, wartość MQMD podana w wywołaniu MQPUT nie może być mniejsza niż MQMD_VERSION_2. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_WRONG_MD_VERSION.

Jeśli parametr MQPMO_LOGICAL_ORDER nie jest określony, komunikaty w grupach i segmentach komunikatów logicznych mogą być umieszczane w dowolnej kolejności i nie jest konieczne umieszczanie pełnych grup komunikatów ani kompletnych komunikatów logicznych. Obowiązkiem aplikacji jest zapewnienie, że pola *GroupId*, *MsgSeqNumber*, *Offset*i *MsgFlags* mają odpowiednie wartości.

Za pomocą tej techniki można restartować grupę komunikatów lub komunikat logiczny w środku, po wystąpieniu awarii systemu. Po zrestartowaniu systemu aplikacja może ustawić pola *GroupId*, *MsgSeqNumber*, *Offset*, *MsgFlags*i *Persistence* na odpowiednie wartości, a następnie wywołać wywołanie MQPUT z zestawem MQPMO_SYNCPOINT lub MQPMO_NO_SYNCPOINT zgodnie z wymaganiami, ale bez określania parametru MQPMO_LOGICAL_ORDER. Jeśli to wywołanie powiedzie się, menedżer kolejek zachowuje informacje o grupie i segmencie, a kolejne wywołania MQPUT używające tego uchwytu kolejki mogą określać parametr MQPMO_LOGICAL_ORDER jako normalny.

Informacje o grupach i segmentach, które menedżer kolejek zachowuje dla wywołania MQPUT, są oddzielone od informacji o grupach i segmentach, które są zachowane dla wywołania MQGET.

W przypadku dowolnego uchwytu kolejki aplikacja może łączyć wywołania MQPUT, które określają parametr MQPMO_LOGICAL_ORDER z wywołaniami MQPUT, które nie są, ale należy zwrócić uwagę na następujące punkty:

- Jeśli parametr MQPMO_LOGICAL_ORDER nie zostanie określony, każde pomyślne wywołanie MQPUT powoduje, że menedżer kolejek ustawia informacje o grupach i segmentach dla uchwytu kolejki na wartości określone przez aplikację, zastępując istniejące informacje o grupach i segmentach zachowane przez menedżer kolejek dla uchwytu kolejki.
- Jeśli parametr MQPMO_LOGICAL_ORDER nie zostanie określony, wywołanie nie powiedzie się, jeśli istnieje bieżąca grupa komunikatów lub komunikat logiczny. Wywołanie może zakończyć się powodzeniem z kodem zakończenia MQCC_WARNING. Tabela 37 na stronie 259 pokazuje różne przypadki, które mogą wystąpić. W takich przypadkach, jeśli kod zakończenia nie jest kodem MQCC_OK, kod przyczyny jest jednym z następujących (odpowiednio):
  - MQRC_INCOMPLETE_GROUP
  - MQRC_INCOMPLETE_MSG
  - MQRC_INCONSISTENT_PERSISTENCE
  - MQRC_INCONSISTENT_UOW

  Uwaga: Menedżer kolejek nie sprawdza informacji o grupie i segmencie dla wywołania MQPUT1 .

Tabela 37. Wynik, gdy wywołanie MQPUT lub MQCLOSE nie jest spójne z informacjami o grupach i segmentach

Bieżące połączenie to	**Poprzednie wywołanie było MQPUT z MQPMO_LOGICAL_ORDER**	**Poprzednie wywołanie było MQPUT bez MQPMO_LOGICAL_ORDER**
MQPUT z MQPMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED
MQPUT bez MQPMO_LOGICAL_ORDER	MQCC_WARNING,	MQCC_OK
MQCLOSE z niezakończonym komunikatem grupowym lub logicznym	MQCC_WARNING,	MQCC_OK

W przypadku aplikacji, które umieszczają komunikaty i segmenty w porządku logicznym, należy określić parametr MQPMO_LOGICAL_ORDER, ponieważ jest to najprostsza opcja do użycia. Ta opcja zwalnia aplikację z potrzeby zarządzania informacjami o grupach i segmentach, ponieważ menedżer kolejek zarządza tą informacją. Jednak wyspecjalizowane aplikacje mogą wymagać większej kontroli niż określona przez opcję MQPMO_LOGICAL_ORDER, która może zostać osiągnięta przez niepodanie tej opcji. Jeśli tak jest, należy upewnić się, że pola*GroupId*, *MsgSeqNumber*, *Offset*i *MsgFlags* w strukturze MQMD są ustawione poprawnie, przed każdą wywołaniem MQPUT lub MQPUT1 .

Na przykład aplikacja, która chce przekazywać komunikaty fizyczne, które otrzymuje, bez względu na to, czy te komunikaty znajdują się w grupach, czy w segmentach komunikatów logicznych, nie może określać parametru MQPMO_LOGICAL_ORDER z dwóch powodów:

- Jeśli komunikaty są pobierane i umieszczane w porządku, podanie wartości MQPMO_LOGICAL_ORDER powoduje przypisanie do komunikatów nowego identyfikatora grupy, co może utrudnić lub uniemożliwić inicjatorowi komunikaty korelowanie wszystkich komunikatów odpowiedzi lub raportów, które wynikają z grupy komunikatów.
- W złożonej sieci z wieloma ścieżkami między wysyłającym i odbierającym menedżery kolejek komunikaty fizyczne mogą być odbierane poza kolejnością. Nie podając wartości MQPMO_LOGICAL_ORDER i MQGMO_LOGICAL_ORDER w wywołaniu MQGET, aplikacja przekazujący może pobrać i przesłać każdy komunikat fizyczny zaraz po jego nadejściu, bez oczekiwania na nadejście kolejnego w kolejności logicznej.

Aplikacje, które generują komunikaty raportów w przypadku komunikatów w grupach lub segmentach komunikatów logicznych, nie mogą również określać parametru MQPMO_LOGICAL_ORDER podczas umieszczania komunikatu raportu.

Parametr MQPMO_LOGICAL_ORDER można określić przy użyciu dowolnej z pozostałych opcji MQPMO_ *.

Umieszczanie grup uporządkowanych logicznie w kolejce klastrowej (MQOO_BIND_ON_GROUP)

Opcja MQOO_BIND_ON_OPEN zapewnia, że wszystkie komunikaty z tej aplikacji, a więc wszystkie grupy, są kierowane do pojedynczej instancji. Ma to miejsce, w którym ruch aplikacji nie jest równoważeniem obciążenia w wielu instancjach kolejki klastra. Aby włączyć równoważenie obciążenia przy jednoczesnym zachowaniu grup komunikatów w stanie nienaruszonym, należy ustawić następujące opcje:

- Wywołanie MQPUT musi określać MQPMO_LOGICAL_ORDER.
- Wywołanie MQOPEN musi określać jedną z dwóch następujących opcji:
  - MQOO_BIND_ON_GROUP
  - MQOO_BIND_AS_Q_DEF, a definicja kolejki musi określać DEFBIND (GROUP)

Równoważenie obciążenia jest następnie sterowane *między grupami* komunikatów bez konieczności użycia komendy MQCLOSE i MQOPEN w kolejce. *Między grupami* oznacza, że parametr MQMF_MSG_IN_GROUP jest ustawiony w strukturze MQMD (v2) lub MQMDE, a w toku nie istnieje częściowa kompletna grupa. Gdy grupa jest w toku, rozstrzygnięty menedżer kolejek i nazwa kolejki w uchwycie obiektu są ponownie wykorzystywane.

Jeśli poprzedni komunikat miał ustawioną wartość MQPMO_LOGICAL_ORDER i/lub MQMF_MSG_IN_GROUP, ale bieżący komunikat nie jest częścią grupy, wywołanie PUT kończy się niepowodzeniem z parametrem MQRC_INCOMPLETE_GROUP.

Jeśli pojedyncza operacja MQPUT nie określa parametru MQPMO_LOGICAL_ORDER, a żadna bieżąca grupa nie jest aktywna, to równoważenie obciążenia jest kierowane dla tego komunikatu (tak jakby wywołanie MQOPEN określone zostało określone w tabeli MQOO_BIND_NOT_FIXED).

Nie ma miejsca do ponownego przydzielenia komunikatów powiązanych z miejscem docelowym za pomocą komendy MQOO_BIND_ON_GROUP. Aby uzyskać więcej informacji na temat ponownego przydzielenia, zobacz: “Grupy komunikatów” na stronie 36.

Grupowanie komunikatów logicznych

Istnieją dwie główne przyczyny korzystania z komunikatów logicznych w grupie:

- Może być konieczne przetworzenie komunikatów w określonej kolejności.
- Może być konieczne przetworzenie każdego komunikatu w grupie w pokrewny sposób.

W każdym z tych przypadków należy pobrać całą grupę z tą samą instancją aplikacji pobierając.

Załóżmy na przykład, że grupa składa się z czterech komunikatów logicznych. Umieszczenie aplikacji wygląda następująco:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT
```

```
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP

MQCMIT
```

Aplikacja pobierający określa opcję MQGMO_ALL_MSGS_AVAILABLE dla pierwszego komunikatu w grupie. Zapewnia to, że przetwarzanie nie zostanie uruchomione, dopóki nie zostaną odebrane wszystkie komunikaty w grupie. Opcja MQGMO_ALL_MSGS_AVAILABLE jest ignorowana w przypadku kolejnych komunikatów w grupie.

Po pobraniu pierwszego komunikatu logicznego grupy można użyć komendy MQGMO_LOGICAL_ORDER, aby upewnić się, że pozostałe komunikaty logiczne grupy są pobierane w kolejności.

Tak więc, pobieranie aplikacji wygląda następująco:

```
/* Wait for the first message in a group, or a message not in a group */
GMO.Options = MQGMO_SYNCPOINT | MQGMO_WAIT
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_LOGICAL_ORDER
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Process each remaining message in the group */
   ...

MQCMIT
```

Więcej informacji na temat grupowania komunikatów można znaleźć w sekcji “Segmentacja aplikacji komunikatów logicznych” na stronie 272 i “Umieszczanie i pobieranie grupy, która obejmuje jednostki pracy” na stronie 261.

Więcej informacji na temat zezwalania aplikacji na żądanie, aby grupa komunikatów była przydzielona do tej samej instancji docelowej dla kolejek klastra, należy zapoznać się z informacjami w sekcji DefBind.

Umieszczanie i pobieranie grupy, która obejmuje jednostki pracy

W poprzednim przypadku komunikaty lub segmenty nie mogą zostać uruchomione w celu opuszczenia węzła (jeśli jego miejsce docelowe jest zdalne) lub uruchomienie do pobrania do momentu umieszczenia całej grupy i zatwierdzenia jednostki pracy. Może to nie być to, czego chcesz, jeśli zajmuje dużo czasu, aby umieścić całą grupę, lub jeśli obszar kolejki jest ograniczony w węźle. Aby przezwyciężyć to, należy umieścić grupę w kilku jednostkach pracy.

Jeśli grupa jest umieszczana w wielu jednostkach pracy, niektóre z grup mogą być zatwierdzane nawet wtedy, gdy wprowadzanie aplikacji nie powiedzie się. W związku z tym aplikacja musi zapisywać informacje o statusie, zatwierdzane przy użyciu każdej jednostki pracy, którą może użyć po restarcie w celu wznowienia niekompletnej grupy. Najprostsze miejsce do zarejestrowania tych informacji znajduje się w kolejce STATUS. Jeśli kompletna grupa została pomyślnie wstawiona, kolejka STATUS jest pusta.

Jeśli segmentacja jest zaangażowana, logika jest podobna. W tym przypadku element StatusInfo musi zawierać *Offset* .

Poniżej przedstawiono przykład umieszczenia grupy w kilku jednostkach pracy:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

/* First UOW */

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
StatusInfo = GroupId,MsgSeqNumber from MQMD
MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
MQCMIT

/* Next and subsequent UOWs */
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
```

```
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
StatusInfo = GroupId,MsgSeqNumber from MQMD
MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
MQCMIT

/* Last UOW */
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
MQCMIT
```

Jeśli wszystkie jednostki pracy zostały zatwierdzone, cała grupa została pomyślnie ułożona, a kolejka STATUS jest pusta. Jeśli nie, grupa musi zostać wznowiona w punkcie wskazanym przez informacje o statusie. MQPMO_LOGICAL_ORDER nie może być użyty do pierwszego umieszczenia, ale może później.

Ponowne uruchomienie przetwarzania wygląda następująco:

```
MQGET (StatusInfo from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
if (Reason == MQRC_NO_MSG_AVAILABLE)
   /* Proceed to normal processing */
   ...

else
   /* Group was terminated prematurely */
   Set GroupId, MsgSeqNumber in MQMD to values from Status message
   PMO.Options = MQPMO_SYNCPOINT
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP

   /* Now normal processing is resumed.
      Assume this is not the last message */
   PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
   StatusInfo = GroupId,MsgSeqNumber from MQMD
   MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
   MQCMIT
```

Z poziomu aplikacji pobierających można rozpocząć przetwarzanie komunikatów w grupie, zanim cała grupa zostanie nadesłana. Powoduje to zwiększenie czasów odpowiedzi w komunikatach w grupie, a także oznacza, że pamięć nie jest wymagana dla całej grupy. W celu zrealizowania korzyści należy użyć kilku jednostek pracy dla każdej grupy komunikatów. Ze względów odzyskiwania konieczne jest pobranie każdego komunikatu w ramach jednostki pracy.

Podobnie jak w przypadku aplikacji wprowadzania danych, wymagane jest, aby informacje o statusie były zapisywane w dowolnym miejscu automatycznie, ponieważ każda jednostka pracy jest zatwierdzana. Najprostsze miejsce rejestrowania tych informacji znajduje się w kolejce STATUS. Jeśli kompletna grupa została pomyślnie przetworzona, kolejka STATUS jest pusta.

Uwaga: W przypadku pośrednich jednostek pracy można uniknąć wywołań MQGET z kolejki STATUS, określając, że każda operacja MQPUT dla kolejki statusu jest segmentem komunikatu (czyli przez ustawienie flagi MQMF_SEGMENT), zamiast umieszczania kompletnego nowego komunikatu dla każdej jednostki pracy. W ostatniej jednostce pracy segment końcowy jest umieszczany w kolejce statusu z parametrem MQMF_LAST_SEGMENT, a następnie informacje o statusie są kasowane za pomocą komendy MQGET, która określa parametr MQGMO_COMPLETE_MSG.

Podczas przetwarzania restartu zamiast pojedynczej operacji MQGET, aby uzyskać możliwy komunikat o statusie, należy przejrzeć kolejkę statusu za pomocą komendy MQGMO_LOGICAL_ORDER, aż do ostatniego segmentu (czyli do momentu, w którym nie zostaną zwrócone żadne kolejne segmenty). W pierwszej jednostce pracy po restarcie należy również określić przesunięcie jawnie podczas umieszczania segmentu statusu.

W poniższym przykładzie rozważamy tylko komunikaty w grupie, zakładając, że bufor aplikacji jest zawsze na tyle duży, aby pomieścić cały komunikat, niezależnie od tego, czy komunikat został podzielony na segmenty. W związku z tym w każdej operacji MQGET określono parametr MQGMO_COMPLETE_MSG. Te same zasady mają zastosowanie w przypadku podziału segmentacji (w tym przypadku element StatusInfo musi zawierać *Offset*).

Dla prostoty zakładamy, że maksymalnie 4 wiadomości są pobierane w ramach jednej jednostki pracy:

```
msgs = 0    /* Counts messages retrieved within UOW */
/* Should be no status message at this point */

/* Retrieve remaining messages in the group */
do while ( GroupStatus == MQGS_MSG_IN_GROUP )

   /* Process up to 4 messages in the group */
   GMO.Options = MQGMO_SYNCPOINT | MQGMO_WAIT
               | MQGMO_LOGICAL_ORDER
   do while ( (GroupStatus == MQGS_MSG_IN_GROUP) && (msgs < 4) )
      MQGET
      msgs = msgs + 1
      /* Process this message */
      ...
   /* end while

   /* Have retrieved last message or 4 messages  */
   /* Update status message if not last in group */
   MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
   if ( GroupStatus == MQGS_MSG_IN_GROUP )
      StatusInfo = GroupId,MsgSeqNumber from MQMD
      MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
   MQCMIT
   msgs = 0
/* end while

if ( msgs > 0 )
   /* Come here if there was only 1 message in the group */
   MQCMIT
```

Jeśli wszystkie jednostki pracy zostały zatwierdzone, cała grupa została pomyślnie wczytana, a kolejka STATUS jest pusta. Jeśli nie, grupa musi zostać wznowiona w punkcie wskazanym przez informacje o statusie. Nie można użyć komendy MQGMO_LOGICAL_ORDER dla pierwszego pobrania, ale może to być później.

Ponowne uruchomienie przetwarzania wygląda następująco:

```
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
if (Reason == MQRC_NO_MSG_AVAILABLE)
   /* Proceed to normal processing */
   ...

else
   /* Group was terminated prematurely */
   /* The next message on the group must be retrieved by matching
      the sequence number and group id with those retrieved from the
      status information. */
   GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT | MQGMO_WAIT
   MQGET GMO.MatchOptions = MQMO_MATCH_GROUP_ID | MQMO_MATCH_MSG_SEQ_NUMBER,
         MQMD.GroupId      = value from Status message,
         MQMD.MsgSeqNumber = value from Status message plus 1
   msgs = 1
   /* Process this message */
   ...

   /* Now normal processing is resumed */
   /* Retrieve remaining messages in the group */
   do while ( GroupStatus == MQGS_MSG_IN_GROUP )

      /* Process up to 4 messages in the group */
      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT | MQGMO_WAIT
                  | MQGMO_LOGICAL_ORDER
      do while ( (GroupStatus == MQGS_MSG_IN_GROUP) && (msgs < 4) )
         MQGET
         msgs = msgs + 1
         /* Process this message */
         ...

      /* Have retrieved last message or 4 messages  */
      /* Update status message if not last in group */
      MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
      if ( GroupStatus == MQGS_MSG_IN_GROUP )
         StatusInfo = GroupId,MsgSeqNumber from MQMD
         MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
      MQCMIT
      msgs = 0
```

Uzyskiwanie określonego komunikatu

Istnieje wiele sposobów uzyskiwania określonego komunikatu z kolejki. Są to następujące elementy: wybór opcji MsgId i CorrelId, wybór opcji GroupId, MsgSeqNumber i Offset, a następnie wybranie opcji MsgToken. Po otwarciu kolejki można również użyć łańcucha wyboru.

Aby uzyskać konkretny komunikat z kolejki, należy użyć pól *MsgId* i *CorrelId* struktury MQMD. Jednak aplikacje mogą jawnie ustawić te pola, tak więc podane wartości mogą nie identyfikować unikalnego komunikatu. Tabela 38 na stronie 264 pokazuje, który komunikat jest pobierany dla możliwych ustawień tych pól. Te pola są ignorowane na wejściu, jeśli określono wartość MQGMO_MSG_UNDER_CURSOR w parametrze *GetMsgOpts* wywołania MQGET.

Tabela 38. Korzystanie z identyfikatorów komunikatów i korelacji

Aby pobrać ...	***MsgId***	***CorrelId***
Pierwszy komunikat w kolejce	MQMI_NONE	MQCI_NONE
Pierwszy komunikat zgodny z *MsgId*	Niezerowe	MQCI_NONE
Pierwszy komunikat zgodny z *CorrelId*	MQMI_NONE	Niezerowe
Pierwszy komunikat zgodny z *MsgId* i *CorrelId*	Niezerowe	Niezerowe

W każdym przypadku wartość *pierwsza* oznacza pierwszy komunikat, który spełnia kryteria wyboru (chyba że określono parametr MQGMO_BROWSE_NEXT, gdy oznacza to *następny* komunikat w sekwencji spełniający kryteria wyboru).

Po powrocie wywołanie MQGET ustawia pola *MsgId* i *CorrelId* na identyfikatory komunikatów i korelacji zwróconego komunikatu (jeśli istnieje).

Jeśli w polu *Version* struktury MQMD zostanie ustawiona wartość 2, można użyć pól *GroupId*, *MsgSeqNumber*i *Offset* . Tabela 39 na stronie 264 informuje, który komunikat jest pobierany dla możliwych ustawień tych pól.

Tabela 39. Korzystanie z identyfikatora grupy

Aby pobrać ...	**opcje dopasowywania**
Pierwszy komunikat w kolejce	MQMO_NONE
Pierwszy komunikat zgodny z *MsgId*	MQMO_MATCH_MSG_ID
Pierwszy komunikat zgodny z *CorrelId*	MQMO_MATCH_KORELID
Pierwszy komunikat zgodny z *GroupId*	MQMO_MATCH_GROUP_ID
Pierwszy komunikat zgodny z *MsgSeqNumber*	MQMO_MATCH_MSG_SEQ_NUMBER
Pierwszy komunikat zgodny z *MsgToken*	MQMO_MATCH_MSG_TOKEN
Pierwszy komunikat zgodny z *Offset*	MQMO_MATCH_OFFSET

Uwagi:

1. MQMO_MATCH_XXX oznacza, że pole *XXX* w strukturze MQMD jest ustawione na wartość, która ma zostać dopasowana.
2. Opcje MQMO mogą być używane w połączeniu. Na przykład: MQMO_MATCH_GROUP_ID, MQMO_MATCH_MSG_SEQ_NUMBER i MQMO_MATCH_OFFSET mogą być używane razem w celu nadania segmentu identyfikowanego za pomocą pól *GroupId*, *MsgSeqNumber*i *Offset* .
3. Jeśli zostanie określona wartość MQGMO_LOGICAL_ORDER, wpłynie to na komunikat, który ma zostać pobrany, ponieważ opcja zależy od informacji o stanie kontrolowanej przez uchwyt kolejki. Więcej informacji na ten temat zawierają “Uporządkowanie logiczne i fizyczne” na stronie 252 i Opcje.

Wywołanie MQGET zwykle pobiera pierwszy komunikat z kolejki. Jeśli podczas korzystania z wywołania MQGET zostanie określony konkretny komunikat, menedżer kolejek musi przeszukac kolejkę, dopóki nie znajdzie tego komunikatu. Może to mieć wpływ na wydajność aplikacji.

Jeśli używana jest wersja 2 lub nowsza struktury MQGMO i nie określono opcji MQMO_MATCH_MSG_ID lub MQMO_MATCH_CORREL_ID, nie ma potrzeby resetowania pól *MsgId* lub *CorrelId* między MQGET.

Konkretny komunikat można pobrać z kolejki, określając jego element MsgToken oraz element MatchOption MQMO_MATCH_MSG_TOKEN w strukturze MQGMO. Element MsgToken jest zwracany przez wywołanie MQPUT, które pierwotnie umieło umieścić ten komunikat w kolejce, lub przez poprzednie operacje MQGET i pozostaje stałe, chyba że menedżer kolejek zostanie zrestartowany.

Jeśli interesujesz się tylko podzbiorem komunikatów w kolejce, możesz określić, które komunikaty mają być przetwarzane przy użyciu łańcucha wyboru z wywołaniem MQOPEN lub MQSUB. Komenda MQGET pobiera następnie następny komunikat, który spełnia ten łańcuch wyboru. Więcej informacji na temat łańcuchów wyboru zawiera sekcja “Selektory” na stronie 22.

Zwiększanie wydajności nietrwałych komunikatów

Gdy klient wymaga komunikatu z serwera, wysyła żądanie do serwera. Wysyła on osobne żądanie dla każdego z komunikatów, które konsumuje. Aby zwiększyć wydajność klientów korzystających z nietrwałych komunikatów przez uniknięcie konieczności wysyłania tych komunikatów żądań, klient może być skonfigurowany do używania *odczytu z wyprzedzeniem*. Odczyt z wyprzedzeniem umożliwia wysyłanie komunikatów do klienta bez konieczności żądania ich żądania.

Gdy opcja odczytu z wyprzedzeniem jest włączona, komunikaty są wysyłane do buforu pamięci na kliencie o nazwie *bufor odczytu z wyprzedzeniem*. Klient będzie miał bufor odczytu z wyprzedzeniem dla każdej kolejki, która została otwarta z włączoną obsługą odczytu z wyprzedzeniem. Komunikaty w buforze odczytu z wyprzedzeniem nie są utrwalane. Klient okresowo aktualizuje serwer, podając informacje o ilości zużywanej przez niego danych.

Po wywołaniu komendy MQOPEN z produktem MQOO_READ_AHEAD, klient WebSphere MQ umożliwia tylko odczyt z wyprzedzeniem, jeśli spełnione są określone warunki. Są one następujące:

- Zarówno klient, jak i menedżer kolejek zdalnych muszą być w wersji WebSphere MQ 7 lub nowszej.
- Aplikacja kliencka musi być skompilowana i powiązana z wątkami bibliotek klienta MQI produktu WebSphere MQ .
- Kanał klienta musi używać protokołu TCP/IP.
- Ustawienie SharingConversations (SHARECNV) kanału musi mieć wartość niezerową w definicji kanału zarówno klienta, jak i serwera.

Użycie odczytu z wyprzedzeniem może zwiększyć wydajność podczas korzystania z nietrwałych komunikatów z aplikacji klienckiej. Ta poprawa wydajności jest dostępna zarówno dla aplikacji MQI, jak i JMS. Aplikacje klienckie używające wywołania MQGET lub asynchronicznego będą korzystać z udoskonaleń wydajności podczas korzystania z nietrwałych komunikatów.

Nie wszystkie projekty aplikacji klienckich nadają się do korzystania z odczytu z wyprzedzeniem, ponieważ nie wszystkie opcje są obsługiwane w przypadku odczytu z wyprzedzeniem, a niektóre opcje są wymagane w celu zachowania spójności między wywołaniami MQGET, gdy funkcja odczytu z wyprzedzeniem jest włączona. Jeśli klient zmienia kryteria wyboru między wywołaniami MQGET, komunikaty zapisywane w buforze odczytu z wyprzedzeniem pozostaną bez zmian w buforze odczytu z wyprzedzeniem klienta.

Jeśli zaległy dziennik komunikatów z poprzednimi kryteriami wyboru nie jest już wymagany, można ustawić konfigurowalny przedział czasu czyszczenia na kliencie, aby automatycznie wyczyścić te komunikaty od klienta. Przedział czasu czyszczenia jest jedną z grup opcji strojenia odczytu z wyprzedzeniem określonych przez klienta. Możliwe jest dostrojenie tych opcji w celu spełnienia wymagań.

Jeśli aplikacja kliencka zostanie zrestartowana, komunikaty w buforze odczytu z wyprzedzeniem mogą zostać utracone. I odwrotnie, komunikat, który został przeniesiony do buforu odczytu z wyprzedzeniem,

mógł zostać usunięty z kolejki bazowej. Nie powoduje to usunięcia go z buforu, dlatego wywołanie MQGET z użyciem odczytu z wyprzedzeniem może zwrócić komunikat, który już nie istnieje.

Odczyt z wyprzedzeniem jest wykonywany tylko dla powiązań klienta. Atrybut jest ignorowany dla wszystkich innych powiązań.

Odczyt z wyprzedzeniem nie ma wpływu na wyzwalanie. Komunikat wyzwalacza nie jest generowany, gdy komunikat jest odczytywawy przed klientem. Odczyt z wyprzedzeniem nie generuje informacji o rachunkach i statystykach, gdy jest on włączony.

Korzystanie z odczytu z wyprzedzeniem przy użyciu funkcji przesyłania komunikatów subskrypcji

Gdy aplikacja subskrybująca określa kolejkę docelową, do której wysyłane są publikacje, wartość DEFREADA określonej kolejki jest używana jako domyślna wartość odczytu z wyprzedzeniem.

Gdy aplikacja subskrybująca żąda, aby produkt WebSphere MQ zarządzał miejscem docelowym, do którego są wysyłane publikacje, kolejka zarządzana jest tworzona jako kolejka dynamiczna oparta na predefiniowanej kolejce modelowej. Jest to wartość DEFREADA kolejki modelowej, która jest używana jako domyślna wartość odczytu z wyprzedzeniem. Domyślne kolejki modelowe SYSTEM.DURABLE.PUBLICATIONS.MODEL lub SYSTEM.NONDURABLE.PUBLICATIONS.MODEL jest używany, chyba że kolejka modelowa jest zdefiniowana dla tego lub nadrzędnego tematu.

Pojęcia pokrewne

“Strojenie wydajności dla nietrwałych komunikatów w systemie AIX” na stronie 268
Jeśli używany jest produkt AIX V5.3 lub nowszy, należy rozważyć ustawienie parametru strojenia w celu użycia pełnej wydajności dla nietrwałych komunikatów.

Zadania pokrewne

“Włączanie i wyłączanie odczytu z wyprzedzeniem” na stronie 267
Domyślnie funkcja odczytu z wyprzedzeniem jest wyłączona. Istnieje możliwość włączenia odczytu z wyprzedzeniem na poziomie kolejki lub aplikacji.

Odsyłacze pokrewne

“Opcje MQGET i odczyt z wyprzedzeniem” na stronie 266
Nie wszystkie opcje MQGET są obsługiwane, jeśli funkcja odczytu z wyprzedzeniem jest włączona. Niektóre opcje są wymagane w celu zachowania spójności między wywołaniami MQGET.

Opcje MQGET i odczyt z wyprzedzeniem

Nie wszystkie opcje MQGET są obsługiwane, jeśli funkcja odczytu z wyprzedzeniem jest włączona. Niektóre opcje są wymagane w celu zachowania spójności między wywołaniami MQGET.

Po wywołaniu komendy MQOPEN z produktem MQOO_READ_AHEAD, klient WebSphere MQ umożliwia tylko odczyt z wyprzedzeniem, jeśli spełnione są określone warunki. Są one następujące:

- Zarówno klient, jak i menedżer kolejek zdalnych muszą być w wersji WebSphere MQ 7 lub nowszej.
- Aplikacja kliencka musi być skompilowana i powiązana z wątkami bibliotek klienta MQI produktu WebSphere MQ .
- Kanał klienta musi używać protokołu TCP/IP.
- Ustawienie SharingConversations (SHARECNV) kanału musi mieć wartość niezerową w definicji kanału zarówno klienta, jak i serwera.

Poniższa tabela zawiera informacje o tym, które opcje są obsługiwane w przypadku odczytu z wyprzedzeniem oraz czy można je zmieniać między wywołaniami MQGET.

Tabela 40. Opcje MQGET i odczyt z wyprzedzeniem

	Dozwolone, gdy funkcja odczytu z wyprzedzeniem jest włączona i może być zmieniana między wywołaniami MQGET[5]	**Dozwolone, gdy funkcja odczytu z wyprzedzeniem jest włączona, ale nie może być zmieniana między wywołaniami MQGET[1]**	**Opcje MQGET, które nie są dozwolone, gdy funkcja odczytu z wyprzedzeniem jest włączona[2]**
Wartości MQGET MQMD	MsgId[3] CorrelId[3]	Kodowanie CodedCharSetId	

Tabela 40. Opcje MQGET i odczyt z wyprzedzeniem (kontynuacja)

	Dozwolone, gdy funkcja odczytu z wyprzedzeniem jest włączona i może być zmieniana między wywołaniami MQGET[5]	**Dozwolone, gdy funkcja odczytu z wyprzedzeniem jest włączona, ale nie może być zmieniana między wywołaniami MQGET[1]**	**Opcje MQGET, które nie są dozwolone, gdy funkcja odczytu z wyprzedzeniem jest włączona[2]**
Opcje MQGMO MQGET	• MQGMO_NO_WAIT • MQGMO_BROWSE_MESSAGE _UNDER_CURSOR • MQGMO_BROWSE_FIRST • MQGMO_BROWSE_NEXT • MQGMO_FAIL_IF_QUIESCING	• MQGMO_SYNCPOINT_IF _PERSISTENT • MQGMO_NO_SYNCPOINT • MQGMO_ACCEPT_OBCIĘTO _MSG • MQGMO_CONVERT	• MQGMO_SET_SIGNAL • MQGMO_SYNCPOINT, • MQGMO_MARK_SKIP _BACKOUT • MQGMO_MSG_UNDER _CURSOR[4] • Blokada MQGMO_LOCK • MQGMO_UNLOCK • MQGMO_LOGICAL_ORDER • MQGMO_COMPLETE_MSG • MQGMO_ALL_MSGS_AVAILABLE • MQGMO_ALL_SEGMENTS_ JEST DOSTĘPNE

Uwagi:

1. Jeśli te opcje zostaną zmienione między wywołaniami MQGET, zwracany jest kod przyczyny MQRC_OPTIONS_CHANGED.
2. Jeśli te opcje zostaną podane podczas pierwszego wywołania MQGET, odczyt z wyprzedzeniem zostanie wyłączony. Jeśli te opcje zostaną podane w kolejnym wywołaniu MQGET, zostanie zwrócony kod przyczyny MQRC_OPTIONS_ERROR.
3. Jeśli wartości parametrów MsgId i CorrelId aplikacji klienckiej między wywołaniami MQGET, komunikaty z poprzednimi wartościami mogły już zostać wysłane do klienta i pozostaną w buforze odczytu z wyprzedzeniem klienta do czasu zużytego (lub automatycznego wyczyszczenia).
4. Opcja MQGMO_MSG_UNDER_CURSOR nie jest dostępna, jeśli włączony jest odczyt z wyprzedzeniem. Funkcja odczytu z wyprzedzeniem jest wyłączona, gdy podczas otwierania kolejki określone są obie opcje MQOO_BROWSE i jedna z opcji MQOO_INPUT_SHARED lub MQOO_INPUT_EXCLUSIVE.
5. Jeśli opcja odczytu z wyprzedzeniem jest włączona, pierwsza komenda MQGET określa, czy komunikaty mają być przeglądane, czy też mają być wysyłane z kolejki. Jeśli aplikacja kliencka używa komendy MQGET ze zmienioną opcją, na przykład przy próbie wyszukania po początkowym dostawie lub próby pobrania za pomocą przeglądania początkowego, zwracany jest kod przyczyny MQRC_OPTIONS_CHANGED.

Jeśli klient zmienia kryteria wyboru między wywołaniami MQGET, komunikaty zapisywane w buforze odczytu z wyprzedzeniem, które są zgodne z początkowym kryterium wyboru, nie są wykorzystywane przez aplikację kliencką i pozostają bez zmian w buforze odczytu z wyprzedzeniem klienta. W sytuacjach, w których bufor odczytu z wyprzedzeniem klienta zawiera wiele komunikatów osieroconych, korzyści związane z odczytanie z wyprzedzeniem są tracone, a dla każdego zużytego komunikatu wymagane jest oddzielne żądanie do serwera. Aby określić, czy funkcja odczytu z wyprzedzeniem jest używana wydajnie, można użyć parametru statusu połączenia (READA).

Odczyt z wyprzedzeniem można zahamować, jeśli aplikacja jest żądana przez aplikację z powodu niezgodnych opcji określonych w pierwszym wywołaniu MQGET. W tej sytuacji status połączenia wskazuje odczyt z wyprzedzeniem jako zahamowany.

Jeśli z powodu tych ograniczeń dla wywołania MQGET użytkownik zdecyduje, że projekt aplikacji klienckiej nie nadaje się do odczytu z wyprzedzeniem, należy określić opcję MQOPEN MQOO_READ_AHEAD_NO. Alternatywnie ustaw domyślną wartość odczytu z wyprzedzeniem dla otwieranej kolejki, która została zmieniona na NO lub DISABLED.

Włączanie i wyłączanie odczytu z wyprzedzeniem

Domyślnie funkcja odczytu z wyprzedzeniem jest wyłączona. Istnieje możliwość włączenia odczytu z wyprzedzeniem na poziomie kolejki lub aplikacji.

O tym zadaniu

Po wywołaniu komendy MQOPEN z produktem MQOO_READ_AHEAD, klient WebSphere MQ umożliwia tylko odczyt z wyprzedzeniem, jeśli spełnione są określone warunki. Są one następujące:

- Zarówno klient, jak i menedżer kolejek zdalnych muszą być w wersji WebSphere MQ 7 lub nowszej.
- Aplikacja kliencka musi być skompilowana i powiązana z wątkami bibliotek klienta MQI produktu WebSphere MQ .
- Kanał klienta musi używać protokołu TCP/IP.
- Ustawienie SharingConversations (SHARECNV) kanału musi mieć wartość niezerową w definicji kanału zarówno klienta, jak i serwera.

Aby włączyć odczyt z wyprzedzeniem:

- Aby skonfigurować odczyt z wyprzedzeniem na poziomie kolejki, ustaw atrybut kolejki DEFREADA na YES.
- Aby skonfigurować odczyt z wyprzedzeniem na poziomie aplikacji:
  - w celu użycia odczytu z wyprzedzeniem wszędzie tam, gdzie to możliwe, należy użyć opcji MQOO_READ_AHEAD w wywołaniu funkcji MQOPEN. Nie jest możliwe, aby aplikacja kliencka używała odczytu z wyprzedzeniem, jeśli atrybut kolejki DEFREADA został ustawiony na wartość DISABLED.
  - w celu użycia odczytu z wyprzedzeniem tylko wtedy, gdy funkcja odczytu z wyprzedzeniem jest włączona w kolejce, należy użyć opcji MQOO_READ_AHEAD_AS_Q_DEF w wywołaniu funkcji MQOPEN.

Jeśli projekt aplikacji klienckiej nie nadaje się do odczytu z wyprzedzeniem, można go wyłączyć:

- na poziomie kolejki, ustawiając atrybut kolejki, DEFREADA na NO, jeśli nie chcesz, aby odczyt z wyprzedzeniem był używany, jeśli nie jest wymagany przez aplikację kliencką, lub WYŁĄCZONY, jeśli nie chcesz, aby odczyt z wyprzedzeniem był używany niezależnie od tego, czy odczyt z wyprzedzeniem jest wymagany przez aplikację kliencką.
- na poziomie aplikacji, za pomocą opcji MQOO_NO_READ_AHEAD w wywołaniu funkcji MQOPEN.

Dwie opcje MQCLOSE umożliwiają skonfigurowanie tego, co dzieje się z komunikatami, które są zapisywane w buforze odczytu z wyprzedzeniem, jeśli kolejka jest zamknięta.

- Użyj komendy MQCO_IMMEDIATE, aby usunąć komunikaty w buforze odczytu z wyprzedzeniem.
- Użyj komendy MQCO_QUIESCE, aby upewnić się, że komunikaty w buforze odczytu z wyprzedzeniem są konsumowane przez aplikację przed zamknięciem kolejki. Gdy zostanie wydana komenda MQCLOSE z opcją MQCO_QUIESCE, a w buforze odczytu z wyprzedzeniem znajdują się komunikaty, komenda MQRC_READ_AHEAD_MSGS powraca z parametrem MQCC_WARNING.

Strojenie wydajności dla nietrwałych komunikatów w systemie AIX

Jeśli używany jest produkt AIX V5.3 lub nowszy, należy rozważyć ustawienie parametru strojenia w celu użycia pełnej wydajności dla nietrwałych komunikatów.

Aby ustawić parametr strojenia w taki sposób, aby był on używany natychmiast, należy wprowadzić następującą komendę jako użytkownik root:

```
/usr/sbin/ioo -o j2_nPagesPerWriteBehindCluster=0
```

Aby ustawić parametr strojenia w taki sposób, aby był on uruchamiany natychmiast po restarcie, należy wprowadzić następującą komendę jako użytkownik root:

```
/usr/sbin/ioo -p -o j2_nPagesPerWriteBehindCluster=0
```

Zwykle komunikaty nietrwałe są przechowywane tylko w pamięci, ale istnieją okoliczności, w których program AIX może zaplanować zapisywanie nietrwałych komunikatów na dysku. Komunikaty zaplanowane do zapisania na dysku są niedostępne dla operacji MQGET, dopóki zapis na dysku nie zostanie zakończony. Sugerowana komenda strojenia zależy od tego progu. Zamiast zapisywania komunikatów w celu zapisania na dysku, gdy 16 kB danych znajduje się w kolejce, zapis na dysk występuje tylko wtedy, gdy rzeczywista pamięć masowa na komputerze staje się bliska zapełnieniowi. Jest to zmiana globalna i może mieć wpływ na inne komponenty oprogramowania.

W systemie AIX, gdy używane są aplikacje wielowątkowe, a w szczególności w przypadku uruchamiania na komputerach z wieloma procesorami, stanowczo zaleca się ustawienie wartości AIXTHREAD_SCOPE=S w identyfikatorze mqm `.profile` lub ustawienie wartości AIXTHREAD_SCOPE=S w środowisku przed uruchomieniem aplikacji, w celu zapewnienia lepszej wydajności i bardziej solidnego harmonogramu. Na przykład:

```
export AIXTHREAD_SCOPE=S
```

Ustawienie AIXTHREAD_SCOPE=S oznacza, że wątki użytkownika utworzone z atrybutami domyślnymi są umieszczane w zasięgu rywalizacji o zasięgu systemowym. Jeśli wątek użytkownika jest tworzony z zasięgiem rywalizacji o zasięgu systemowym, jest on powiązany z wątkiem jądra, który jest zaplanowany przez jądro. Bazowy wątek jądra nie jest współużytkowany z żadnym innym wątkiem użytkownika.

deskryptory plików.

W przypadku uruchamiania wielowątkowego procesu, takiego jak proces agenta, można osiągnąć miękki limit dla deskryptorów plików. Limit ten daje kod przyczyny IBM WebSphere MQ `MQRC_UNEXPECTED_ERROR (2195)` oraz, jeśli liczba deskryptorów plików jest wystarczająca, plik IBM WebSphere MQ FFST™ .

Aby uniknąć tego problemu, można zwiększyć limit procesu dla liczby deskryptorów plików. W tym celu należy zmienić atrybut `nofiles` w `/etc/security/limits` na 10 000 w przypadku identyfikatora użytkownika mqm lub w sekcji default.

Limity zasobów systemowych

Ustaw limit zasobów systemowych dla segmentu danych i segmentu stosu na nieograniczony, korzystając z następujących komend w wierszu komend:

```
ulimit -d unlimited
ulimit -s unlimited
```

Obsługa komunikatów o długości większej niż 4 MB

Komunikaty mogą być zbyt duże dla aplikacji, kolejki lub menedżera kolejek. W zależności od środowiska produkt WebSphere MQ udostępnia wiele sposobów postępowania z komunikatami, które są dłuższe niż 4 MB.

Można zwiększyć wartość atrybutu *MaxMsgLength* do 100 MB we wszystkich systemach WebSphere MQ w wersji V6 lub nowszej. Ustaw tę wartość, aby odzwierciedlić wielkość komunikatów korzystających z kolejki. W systemach WebSphere MQ innych niż WebSphere MQ for z/OSmożna również wykonać następujące czynności:

1. Użyj segmentowanych komunikatów. (Komunikaty mogą być segmentowane przez aplikację lub menedżer kolejek).
2. Użyj komunikatów odniesienia.

Każde z tych podejść zostało opisane w dalszej części niniejszej sekcji.

Zwiększanie maksymalnej długości komunikatu

Atrybut menedżera kolejek produktu *MaxMsgLength* definiuje maksymalną długość komunikatu, który może być obsługiwany przez menedżer kolejek. Podobnie, atrybut kolejki *MaxMsgLength* to maksymalna długość komunikatu, który może być obsługiwany przez kolejkę. Maksymalna obsługiwana długość komunikatu *domyślna* zależy od środowiska, w którym pracuje użytkownik.

W przypadku obsługi dużych komunikatów atrybuty te można zmieniać niezależnie. Wartość atrybutu menedżera kolejek można ustawić w zakresie od 32768 bajtów do 100 MB; wartość atrybutu kolejki można ustawić w zakresie od 0 do 100 MB.

Po zmianie jednego lub obu atrybutów programu *MaxMsgLength* zrestartuj aplikacje i kanały, aby zmiany zostały uwzględnione.

Po wprowadzeniu tych zmian długość komunikatu musi być mniejsza lub równa zarówno dla kolejki, jak i dla atrybutów *MaxMsgLength* menedżera kolejek. Jednak istniejące komunikaty mogą być dłuższe niż jeden z atrybutów.

Jeśli komunikat jest zbyt duży dla kolejki, zwracana jest wartość MQRC_MSG_TOO_BIG_FOR_Q. Podobnie, jeśli komunikat jest zbyt duży dla menedżera kolejek, zwracana jest wartość MQRC_MSG_TOO_BIG_FOR_Q_MGR.

Ta metoda obsługi dużych wiadomości jest łatwa i wygodna. Przed użyciem należy jednak rozważyć następujące czynniki:

- Równomierność wśród menedżerów kolejek jest zmniejszona. Maksymalna wielkość danych komunikatu jest określana przez *MaxMsgLength* dla każdej kolejki (włącznie z kolejkami transmisji), na której zostanie umieszczony komunikat. Ta wartość jest często ustawiana domyślnie na *MaxMsgLength*menedżera kolejek, zwłaszcza w przypadku kolejek transmisji. Utrudnia to określenie, czy komunikat jest zbyt duży, gdy ma być on podróżowany do zdalnego menedżera kolejek.
- Użycie zasobów systemowych zwiększa się. Na przykład aplikacje potrzebują większych buforów, a na niektórych platformach mogą być zwiększone użycie współużytkowanej pamięci masowej. Należy mieć wpływ na pamięć masową kolejki tylko wtedy, gdy jest to wymagane w przypadku większych komunikatów.
- Dotknięcie kanału jest zakłócające. Duży komunikat jest nadal liczony jako tylko jeden komunikat w stosunku do liczby partii, ale wymaga dłuższego przesyłania, zwiększając tym samym czasy odpowiedzi dla innych komunikatów.

Segmentacja komunikatów

Ta sekcja zawiera informacje na temat segmentacji komunikatów.

Uwaga: Nie jest obsługiwane w produkcie IBM WebSphere MQ for z/OS lub przez aplikacje korzystające z klas IBM WebSphere MQ dla usługi JMS.

Zwiększenie maksymalnej długości komunikatu, jak wyjaśniono w temacie “Zwiększanie maksymalnej długości komunikatu” na stronie 269 , ma pewne negatywne konsekwencje. Ponadto nadal może spowodować, że komunikat jest zbyt duży dla kolejki lub menedżera kolejek. W takich przypadkach można segmentować komunikat. Informacje na temat segmentów zawiera sekcja “Grupy komunikatów” na stronie 36.

W następnych sekcjach używane są wspólne zastosowania służące do segmentacji komunikatów. Zakłada się, że wywołania MQPUT lub MQGET *zawsze* działają w ramach jednostki pracy. Zawsze warto rozważyć zastosowanie tej techniki w celu zmniejszenia możliwości obecnych w sieci grup niekompletnych. Zakłada się, że zatwierdzanie jednofazowe przez menedżer kolejek jest poprawne, ale inne techniki koordynacji są równie ważne.

Ponadto w aplikacjach pobierających przyjmuje się, że jeśli wiele serwerów przetwarza tę samą kolejkę, każdy serwer wykonuje podobny kod, tak aby jeden serwer nigdy nie znalazł komunikatu lub segmentu, który oczekiwał na to, że tam będzie (ponieważ podano wcześniej wartość MQGMO_ALL_MSGS_AVAILABLE lub MQGMO_ALL_SEGMENTS_AVAILABLE).

Umieszczanie i zdobycie posegmentowanego przekazu, który obejmuje jednostki pracy

Można umieścić i uzyskać segmentowany komunikat, który obejmuje jednostkę pracy w podobny sposób, jak produkt “Umieszczanie i pobieranie grupy, która obejmuje jednostki pracy” na stronie 261.

Nie można jednak umieszczać lub umieszczać segmentowanych komunikatów w globalnej jednostce pracy.

Segmentacja i ponowne składanie przez menedżera kolejek
Jest to najprostszy scenariusz, w którym jedna aplikacja umieszcza komunikat, który ma zostać pobrany przez inną aplikację. Komunikat może być duży: nie jest zbyt duży, aby można było obsłużyć aplikację lub aplikację pobierający w pojedynczym buforze, ale jest zbyt duży dla menedżera kolejek lub kolejki, w której ma zostać umieszczony komunikat.

Jedynymi zmianami niezbędnymi dla tych aplikacji są wprowadzanie aplikacji w celu autoryzowania menedżera kolejek w celu przeprowadzenia segmentacji, jeśli jest to konieczne:

```
PMO.Options = (existing options)
MD.MsgFlags = MQMF_SEGMENTATION_ALLOWED
memcpy(MD.GroupId, MQGI_NONE, MQ_GROUP_ID_LENGTH)
MQPUT
```

i aplikacji pobierających, aby poprosić menedżera kolejek o ponowne złożenie komunikatu, jeśli został on podzielony na segmenty:

```
GMO.Options = MQGMO_COMPLETE_MSG | (existing options)
MQGET
```

W tym najprostszym scenariuszu aplikacja musi zresetować pole GroupId na wartość MQGI_NONE przed wywołaniem MQPUT, tak aby menedżer kolejek mógł wygenerować unikalny identyfikator grupy dla każdego komunikatu. Jeśli nie jest to zrobione, niepowiązane komunikaty mogą mieć ten sam identyfikator grupy, co może prowadzić do nieprawidłowego przetwarzania.

Bufor aplikacji musi być na tyle duży, aby zawierał ponownie zmontowany komunikat (chyba że zostanie podana opcja MQGMO_ACCEPT_TRUNCATED_MSG).

Jeśli atrybut MAXMSGLEN kolejki ma być modyfikowany w taki sposób, aby pomieścić segmentację komunikatów, należy rozważyć następujące kwestie:

- Minimalny segment komunikatu obsługiwany w kolejce lokalnej to 16 bajtów.
- W przypadku kolejki transmisji wartość MAXMSGLEN musi zawierać również miejsce wymagane dla nagłówków. Należy rozważyć użycie wartości o wielkości co najmniej 4000 bajtów większej od maksymalnej oczekiwanej długości danych użytkownika w dowolnym segmencie komunikatów, który można umieścić w kolejce transmisji.

Jeśli konieczna jest konwersja danych, może to być konieczne, aby aplikacja pobierała wartość MQGMO_CONVERT. Powinno to być proste, ponieważ wyjście konwersji danych jest przedstawiane wraz z kompletnym komunikatem. Nie próbuj konwertować danych w kanale nadawczym, jeśli wiadomość jest posegmentowana, a format danych jest taki, że wyjście konwersji danych nie może przeprowadzić konwersji na niekompletnych danych.

Segmentacja aplikacji
Segmentacja aplikacji jest używana, gdy segmentacja menedżera kolejek jest niewystarczająca lub gdy aplikacje wymagają konwersji danych z określonymi granicami segmentów.

Segmentacja aplikacji jest używana z dwóch głównych powodów:

1. Sama segmentacja menedżera kolejek jest niewystarczająca, ponieważ komunikat jest zbyt duży, aby można go było obsłużyć w pojedynczym buforze przez aplikacje.
2. Konwersja danych musi być wykonywana przez kanały nadawcze, a format jest taki, że aplikacja musi określać miejsce, w którym granice segmentu mają być w celu przekształcenia danego segmentu w możliwy sposób.

Jeśli jednak konwersja danych nie jest problemem lub jeśli aplikacja pobierający zawsze używa parametru MQGMO_COMPLETE_MSG, segmentacja menedżera kolejek może być również dozwolona przez określenie parametru MQMF_SEGMENTATION_ALLOWED. W naszym przykładzie aplikacja segmentuje komunikat w czterech segmentach:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT
```

```
MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_SEGMENT

MQCMIT
```

Jeśli parametr MQPMO_LOGICAL_ORDER nie zostanie użyty, aplikacja musi ustawić *Offset* i długość każdego segmentu. W takim przypadku stan logiczny nie jest obsługiwany automatycznie.

Aplikacja pobierające nie może zagwarantować, że bufor jest wystarczająco duży, aby pomieścić wszystkie ponownie złożone komunikaty. W związku z tym musi być ona przygotowana do indywidualnego przetwarzania segmentów.

W przypadku komunikatów posegmentowanych ta aplikacja nie chce rozpoczynać przetwarzania jednego segmentu, dopóki wszystkie segmenty, które nie stanowią komunikatu logicznego, są obecne. Wartość MQGMO_ALL_SEGMENTS_AVAILABLE jest zatem określona dla pierwszego segmentu. Jeśli określono parametr MQGMO_LOGICAL_ORDER, a istnieje bieżący komunikat logiczny, opcja MQGMO_ALL_SEGMENTS_AVAILABLE jest ignorowana.

Po pobraniu pierwszego segmentu komunikatu logicznego należy użyć komendy MQGMO_LOGICAL_ORDER, aby upewnić się, że pozostałe segmenty komunikatu logicznego są pobierane w kolejności.

Wiadomości nie są brane pod uwagę w różnych grupach. Jeśli takie komunikaty wystąpią, są one przetwarzane w kolejności, w jakiej pierwszy segment każdego komunikatu występuje w kolejce.

```
GMO.Options = MQGMO_SYNCPOINT | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_SEGMENTS_AVAILABLE | MQGMO_WAIT
do while ( SegmentStatus == MQSS_SEGMENT )
   MQGET
   /* Process each remaining segment of the logical message */
   ...

MQCMIT
```

Segmentacja aplikacji komunikatów logicznych
Komunikaty muszą być przechowywane w porządku logicznym w grupie, a niektóre lub wszystkie z nich mogą być tak duże, że wymagają segmentacji aplikacji.

W naszym przykładzie należy umieścić grupę czterech komunikatów logicznych. Wszystkie, ale trzeci komunikat są duże i wymagają segmentacji, która jest wykonywana przez umieszczanie aplikacji:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_LAST_SEGMENT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_LAST_SEGMENT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP

MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_LAST_SEGMENT

MQCMIT
```

W aplikacji pobierających wartość MQGMO_ALL_MSGS_AVAILABLE jest określona w pierwszej operacji MQGET. Oznacza to, że nie są pobierane żadne komunikaty ani segmenty grupy, dopóki cała grupa nie będzie dostępna. Po pobraniu pierwszej fizycznej wiadomości grupy MQGMO_LOGICAL_ORDER służy do zapewnienia, że segmenty i komunikaty grupy są pobierane w kolejności:

```
GMO.Options = MQGMO_SYNCPOINT | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_WAIT
```

```
do while ( (GroupStatus   != MQGS_LAST_MSG_IN_GROUP) ||
             (SegmentStatus != MQGS_LAST_SEGMENT) )
   MQGET
   /* Process a segment or complete logical message.  Use the GroupStatus
      and SegmentStatus information to see what has been returned */
   ...

MQCMIT
```

Uwaga: Jeśli określono parametr MQGMO_LOGICAL_ORDER i istnieje bieżąca grupa, komenda MQGMO_ALL_MSGS_AVAILABLE zostanie zignorowana.

Komunikaty odniesienia

Te informacje umożliwiają zapoznanie się z informacjami na temat komunikatów referencyjnych.

Uwaga: Nieobsługiwane w produkcie WebSphere MQ for z/OS.

Ta metoda umożliwia przeniesienie dużego obiektu z jednego węzła do drugiego bez zapisywania obiektu w kolejkach produktu WebSphere MQ w węźle źródłowym lub docelowym. Jest to szczególnie korzystne, gdy dane istnieją w innej formie, na przykład w przypadku aplikacji poczty elektronicznej.

W tym celu należy określić wyjście komunikatu na obu końcach kanału. Informacje na temat sposobu wykonania tej czynności zawiera sekcja “Programy obsługi wyjścia komunikatów kanału” na stronie 421.

Produkt WebSphere MQ definiuje format nagłówka komunikatu odwołania (MQRMH). Opis tego parametru zawiera sekcja MQRMH . Jest on rozpoznawany przy użyciu zdefiniowanej nazwy formatu i może po niej następować rzeczywiste dane.

Aby zainicjować przesyłanie dużego obiektu, aplikacja może umieścić komunikat składający się z nagłówka komunikatu odwołania bez następującego po nim danych. Ponieważ ten komunikat opuszcza węzeł, wyjście komunikatu pobiera obiekt w odpowiedni sposób i dodaje go do komunikatu odniesienia. Następnie zwraca komunikat (obecnie większy niż poprzednio) do wysyłającego agenta kanału komunikatów w celu przesłania do odbierającego agenta MCA.

Inne wyjście komunikatów jest skonfigurowane w odbierającym MCA. Gdy to wyjście komunikatu odbiera jeden z tych komunikatów, tworzy obiekt przy użyciu danych obiektu, które zostały dopisane i przekazuje komunikat *bez* tego komunikatu. Komunikat odniesienia może teraz zostać odebrany przez aplikację, a aplikacja wie, że obiekt (lub co najmniej jego część reprezentowana przez ten komunikat odniesienia) został utworzony w tym węźle.

Maksymalna ilość danych obiektu, które program zewnętrzny wysyłający komunikat może dopisać do komunikatu referencyjnego, jest ograniczona przez wynegocjowaną maksymalną długość komunikatu dla kanału. Wyjście może zwrócić tylko jeden komunikat do agenta MCA dla każdego komunikatu, który został przekazany, tak więc aplikacja umieszczanie może umieścić kilka komunikatów, aby spowodować przeniesienie jednego obiektu. Każdy komunikat musi identyfikować *logiczną* długość i przesunięcie obiektu, który ma być do niego dodawany. Jednak w przypadkach, gdy nie jest możliwe poznanie łącznej wielkości obiektu lub maksymalnej wielkości dozwolonej przez kanał, zaprojektuj wyjście komunikatu, tak aby aplikacja umieszczała tylko jeden komunikat, a samo wyjście umieszcza następny komunikat w kolejce transmisji, gdy dopisał on tyle danych, ile może do wiadomości, które zostało przekazane.

Przed skorzystaniem z tej metody postępowania z dużymi komunikatami należy rozważyć następujące kwestie:

- Agent MCA i wyjście komunikatów są uruchamiane w ramach identyfikatora użytkownika produktu WebSphere MQ . Wyjście komunikatu (a więc identyfikator użytkownika) wymaga dostępu do obiektu w celu pobrania go na końcu wysyłającego lub utworzenia go na końcu odbierającym. Może to być możliwe tylko w przypadkach, gdy obiekt jest powszechnie dostępny. To rodzi problem z bezpieczeństwem.
- Jeśli komunikat referencyjny z dołączonymi danymi masowymi musi przechodzić przez kilka menedżerów kolejek przed dotarciem do miejsca docelowego, to dane masowe *są* obecne w kolejkach WebSphere MQ w węzłach między nimi. W takich przypadkach nie trzeba jednak zapewnić specjalnego wsparcia lub wyjścia.

- Projektowanie wyjścia komunikatów jest utrudnione, jeśli dozwolone jest kolejkowanie przekierowywania lub kolejkowania niewysłanych wiadomości. W takich przypadkach fragmenty obiektu mogą być odbierane poza kolejką.
- Gdy komunikat odniesienia dociera do miejsca docelowego, wyjście komunikatu odbierającego powoduje utworzenie obiektu. Nie jest to jednak synchronizowane z jednostką pracy agenta MCA, więc jeśli partia jest wycofana, w późniejszym zadaniu wsadowym pojawi się inny komunikat referencyjny zawierający tę samą część obiektu, a wyjście komunikatu może próbować ponownie utworzyć tę samą część obiektu. Jeśli obiekt jest na przykład serią aktualizacji bazy danych, może to być niedopuszczalne. Jeśli tak, wyjście komunikatu musi zawierać dziennik, którego aktualizacje zostały zastosowane; może to wymagać użycia kolejki produktu WebSphere MQ .
- W zależności od charakterystyki typu obiektu, wyjścia komunikatów i aplikacje mogą wymagać współpracy przy obsłudze liczników użycia, aby można było usunąć obiekt, gdy nie jest on już potrzebny. Być może także wymagany jest identyfikator instancji; w nagłówku komunikatu referencyjnego jest dostępne pole (patrz MQRMH).
- Jeśli komunikat odniesienia jest umieszczany jako lista dystrybucyjna, obiekt musi być pobieralny dla każdej wynikowej listy dystrybucyjnej lub pojedynczego miejsca docelowego w tym węźle. Może być konieczne utrzymanie liczników użycia. Należy również rozważyć możliwość, że węzeł może być węzłem końcowym dla niektórych miejsc docelowych na liście, ale dla innych węzłów pośrednich.
- Dane masowe nie są zazwyczaj przekształcane. Jest to spowodowane tym, że konwersja ma miejsce *przed* wywołaniem wyjścia komunikatu. Z tego powodu konwersja nie może być żądana dla źródłowego kanału nadawczego. Jeśli komunikat odniesienia przechodzi przez węzeł pośredni, dane masowe są przekształcane po wysłaniu z węzła pośredniego, jeśli jest to wymagane.
- Nie można segmentować komunikatów referencyjnych.

Korzystanie z struktur MQRMH i MQMD

Patrz MQRMH i MQMD , aby uzyskać opis pól w nagłówku komunikatu odwołania i w deskryptorze komunikatu.

W strukturze MQMD ustaw pole *Format* na wartość MQFMT_REF_MSG_HEADER. Format MQHREF, o ile jest to wymagane dla komendy MQGET, jest automatycznie przekształcany przez produkt WebSphere MQ wraz z danymi masowymi, które są następujące.

Poniżej znajduje się przykład użycia pól *DataLogicalOffset* i *DataLogicalLength* w MQRMH:

Aplikacja umieszczana w aplikacji może umieścić komunikat referencyjny z:

- Brak danych fizycznych
- *DataLogicalLength* = 0 (ten komunikat reprezentuje cały obiekt)
- *DataLogicalOffset* = 0.

Zakładając, że obiekt ma długość 70 000 bajtów, wyjście komunikatu wysyłającego wysyła pierwsze 40 000 bajtów wzdłuż kanału w komunikacie referencyjnym zawierającym:

- 40 000 bajtów danych fizycznych po MQRMH
- *DataLogicalLength* = 40000
- *DataLogicalOffset* = 0 (od początku obiektu).

Następnie umieszcza inny komunikat w kolejce transmisji zawierającej:

- Brak danych fizycznych
- *DataLogicalLength* = 0 (do końca obiektu). Można tu podać wartość 30 000.
- *DataLogicalOffset* = 40000 (począwszy od tego punktu).

Gdy wyjście komunikatu jest widoczne przez wyjście komunikatu wysyłającego, pozostałe 30 000 bajtów danych jest dołączanych, a pola są ustawione na:

- 30 000 bajtów danych fizycznych po MQRMH
- *DataLogicalLength* = 30000

- *DataLogicalOffset* = 40000 (począwszy od tego punktu).

Ustawiona jest również flaga MQRMHF_LAST.

Opis przykładowych programów udostępnionych w celu użycia komunikatów referencyjnych znajduje się w sekcji “Przykładowe programy dla platform rozproszonych” na stronie 99.

Oczekiwanie na komunikaty

Jeśli program ma czekać do momentu nadejścia komunikatu do kolejki, należy podać opcję MQGMO_WAIT w polu *Options* struktury MQGMO.

Użyj pola *WaitInterval* w strukturze MQGMO, aby określićMaksymalny czas (w milisekundach), przez jaki wywołanie MQGET ma oczekiwać na przesłanie komunikatu do kolejki.

Jeśli komunikat nie zostanie wyświetlony w tym czasie, wywołanie MQGET zostanie zakończone z kodem przyczyny MQRC_NO_MSG_AVAILABLE.

W polu *WaitInterval* można określić nieograniczony przedział czasu oczekiwania, korzystając ze stałej MQWI_UNLIMITED. Jednak zdarzenia znajdujące się poza kontrolą mogą spowodować, że program będzie czekał przez długi czas, dlatego należy zachować ostrożność przy zachowaniu ostrożności. Aplikacje IMS nie mogą określać nieograniczonego przedziału czasu oczekiwania, ponieważ uniemożliwiłoby to zakończenie pracy systemu IMS . (Gdy system IMS kończy działanie, wymagane jest zakończenie wszystkich regionów zależnych). Zamiast tego aplikacje IMS mogą określać skończony okres oczekiwania, a następnie, jeśli wywołanie zakończy się bez pobierania komunikatu po upływie tego okresu, wywołaj inną wywołanie MQGET z opcją oczekiwania.

Uwaga: Jeśli więcej niż jeden program oczekuje w tej samej kolejce współużytkowanej na *usunięcie* komunikatu, tylko jeden program jest aktywowany przez przylot komunikatu. Jeśli jednak więcej niż jeden program oczekuje na przeglądanie wiadomości, wszystkie programy mogą być aktywowane. Więcej informacji na ten temat zawiera opis pola *Options* w strukturze MQGMO w produkcie MQGMO.

Jeśli stan kolejki lub menedżer kolejek ulegnie zmianie przed upływem odstępu czasu oczekiwania, wykonywane są następujące działania:

- Jeśli menedżer kolejek przejdzie w stan wygaszania, a użyto opcji MQGMO_FAIL_IF_QUIESCING, to oczekiwanie zostanie anulowane, a wywołanie MQGET zakończy się z kodem przyczyny MQRC_Q_MGR_QUIESCING. Bez tej opcji połączenie oczekuje na oczekiwanie.
- Jeśli menedżer kolejek jest zmuszony do zatrzymania lub został anulowany, wywołanie MQGET kończy się albo z kodem przyczyny MQRC_Q_MGR_ZATRZYMYWANIA, albo z kodem przyczyny MQRC_CONNECTION_BROKEN.
- Jeśli atrybuty kolejki (lub kolejki, do której tłumaczona jest nazwa kolejki) zostaną zmienione w taki sposób, że żądania pobierania są teraz wstrzymane, oczekiwanie zostanie anulowane, a wywołanie MQGET zakończy się z kodem przyczyny MQRC_GET_INHIBITED.
- Jeśli atrybuty kolejki (lub kolejki, do której tłumaczona jest nazwa kolejki), zostaną zmienione w taki sposób, że wymagana jest opcja FORCE, to oczekiwanie zostanie anulowane, a wywołanie MQGET zakończy się kodem przyczyny MQRC_OBJECT_CHANGED.

Aby uzyskać więcej informacji na temat okoliczności, w których te działania są wykonywane, patrz MQGMO.

Pomijanie wycofania

Aby program użytkowy nie mógł wejść do pętli *MQGET-error-backout* , można określić opcję **MQGMO_MARK_SKIP_BACKOUT** w wywołaniu MQGET.

Uwaga: Obsługiwane tylko w produkcie WebSphere MQ for z/OS.

W ramach jednostki pracy program użytkowy może wydać jedną lub więcej wywołań MQGET w celu pobrania komunikatów z kolejki. Jeśli program użytkowy wykryje błąd, może wycofać się z jednostki pracy. Spowoduje to odtworzenie wszystkich zasobów zaktualizowanych podczas tej jednostki pracy do stanu, w którym znajdowały się one przed uruchomieniem jednostki pracy, a następnie przywróci komunikaty pobrane przez wywołania MQGET.

Po przywróceniu te komunikaty są dostępne dla kolejnych wywołań MQGET wydawanych przez program użytkowy. W wielu przypadkach nie powoduje to problemu z programem użytkowym. Jednak w przypadku, gdy błąd prowadzący do wycofania nie może być obejście, komunikat przywrócony do kolejki może spowodować, że program użytkowy wprowadzi pętlę *MQGET-error-backout* .

Aby uniknąć tego problemu, należy określić opcję MQGMO_MARK_SKIP_BACKOUT w wywołaniu MQGET. Oznacza to, że żądanie MQGET nie jest używane w wycofaniu zainicjowanym przez aplikację. To znaczy, że nie może on być wycofany. Użycie tej opcji oznacza, że w razie wystąpienia wycofania aktualizacje innych zasobów są wycofane zgodnie z wymaganiami, ale oznaczony komunikat jest traktowany tak, jakby został pobrany w ramach nowej jednostki pracy.

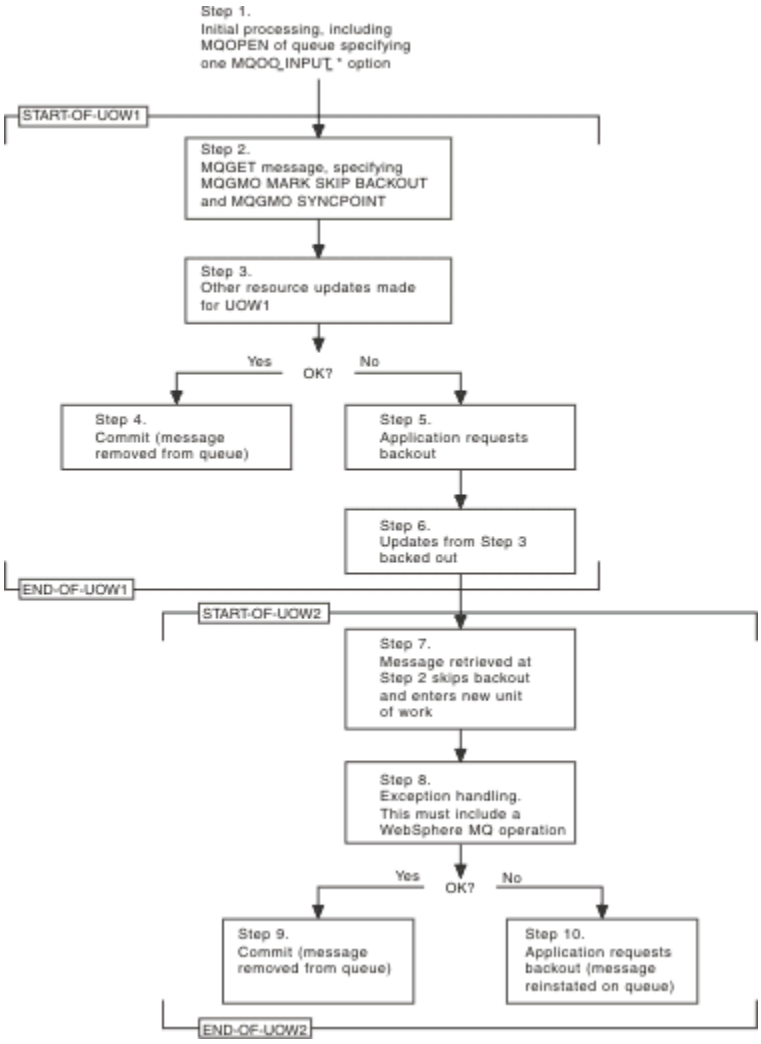
Program użytkowy musi wywołać wywołanie programu WebSphere MQ w celu zatwierdzenia nowej jednostki pracy lub utworzenia kopii zapasowej nowej jednostki pracy. Na przykład program może wykonać obsługę wyjątków, na przykład informując inicjatora, że komunikat został usunięty, i zatwierdzić jednostkę pracy, tak aby usunąć komunikat z kolejki. Jeśli nowa jednostka pracy zostanie wycofana (z jakiegokolwiek powodu), komunikat zostanie przywrócony do kolejki.

W ramach jednostki pracy może istnieć tylko jedno żądanie MQGET oznaczone jako pominięcie wycofania. Jednak może istnieć kilka innych komunikatów, które nie są oznaczone jako pomijanie wycofania. Gdy komunikat zostanie oznaczony jako pomijanie wycofania, wszystkie kolejne wywołania MQGET w ramach jednostki pracy, które określają wartość MQGMO_MARK_SKIP_BACKOUT, kończą się niepowodzeniem. Kod przyczyny: MQRC_SECOND_MARK_NOT_ALLOWED.

Uwaga:

1. Zaznaczony komunikat pomija wycofany komunikat tylko wtedy, gdy jednostka pracy zawierająca ją zostanie zakończona przez żądanie aplikacji w celu wycofania go. Jeśli jednostka pracy zostanie wycofana z innego powodu, zostanie ona wycofana do kolejki w taki sam sposób, w jaki będzie miała miejsce, gdyby nie została oznaczona do pominięcia wycofania.
2. Pominięcie wycofania nie jest obsługiwane w ramach procedur składowanych DB2 uczestniczących w jednostkach pracy kontrolowanych przez usługi RRS. Na przykład wywołanie MQGET z opcją MQGMO_MARK_SKIP_BACKOUT nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_OPTION_ENVIRONMENT_ERROR.

Rysunek 36 na stronie 277 przedstawia typową sekwencję kroków, które może zawierać program użytkowy, gdy żądanie MQGET jest wymagane do pominięcia wycofania.



Rysunek 36. Pomijanie wycofań za pomocą komendy MQGMO_MARK_SKIP_BACKOUT

Kroki w programie Rysunek 36 na stronie 277 są następujące:

Krok 1
Początkowe przetwarzanie odbywa się w ramach transakcji, w tym wywołanie MQOPEN w celu otwarcia kolejki (podanie jednej z opcji MQOO_INPUT_ * w celu pobrania komunikatów z kolejki w kroku 2).

Krok 2
Wywołano komendę MQGET z opcją MQGMO_SYNCPOINT i MQGMO_MARK_SKIP_BACKOUT. MQGMO_SYNCPOINT jest wymagany, ponieważ MQGET musi znajdować się w jednostce pracy dla komendy MQGMO_MARK_SKIP_BACKOUT, aby była ona efektywna. W produkcie Rysunek 36 na stronie 277 ta jednostka pracy jest określana jako UOW1.

Krok 3
Inne aktualizacje zasobów są wykonywane jako część UOW1. Mogą one obejmować kolejne wywołania MQGET (wydane bez komendy MQGMO_MARK_SKIP_BACKOUT).

Krok 4
Wszystkie aktualizacje z kroków 2 i 3 są kompletne zgodnie z wymaganiami. Program użytkowy zatwierdza aktualizacje, a UOW1 kończy działanie. Komunikat pobrany w kroku 2 jest usuwany z kolejki.

Krok 5
Niektóre aktualizacje z kroków 2 i 3 nie są kompletne zgodnie z wymaganiami. Program użytkowy żąda, aby aktualizacje dokonane podczas wykonywania tych kroków zostały wycofane.

Krok 6
Aktualizacje wprowadzone w kroku 3 są wycofane.

Krok 7
Żądanie MQGET wykonane w kroku 2 pomija wycofanie i staje się częścią nowej jednostki pracy UOW2.

Krok 8
UOW2 wykonuje obsługę wyjątków w odpowiedzi na wycofany element UOW1 . (Na przykład wywołanie MQPUT w innej kolejce, wskazując, że wystąpił problem, który spowodował, że UOW1 został wycofany).

Krok 9
Krok 8 zakończy się zgodnie z wymaganiami, program aplikacji zatwierdza działanie, a UOW2 kończy działanie. Ponieważ żądanie MQGET jest częścią obiektu UOW2 (patrz krok 7), to zatwierdzenie powoduje usunięcie komunikatu z kolejki.

Krok 10
Krok 8 nie jest kompletny, jeśli jest wymagany, a program użytkowy tworzy kopię zapasową UOW2. Ponieważ żądanie pobrania komunikatu jest częścią UOW2 (patrz krok 7), to jest on również wycofany i przywrócony do kolejki. Jest on teraz dostępny dla kolejnych wywołań MQGET wydanych przez ten lub inny program użytkowy (w taki sam sposób, jak każdy inny komunikat w kolejce).

Konwersja danych aplikacji

Jeśli to konieczne, MCAs przekształci deskryptor komunikatu i dane nagłówka w wymagany zestaw znaków i kodowanie. Konwersja może być zakończona albo na końcu łącza (czyli lokalnego agenta MCA lub zdalnego agenta MCA).

Gdy aplikacja umieszcza komunikaty w kolejce, lokalny menedżer kolejek dodaje informacje sterujące do deskryptorów komunikatów w celu ułatwienia sterowania komunikatami podczas ich przetwarzania przez menedżery kolejek i MCAs. W zależności od środowiska, pola danych nagłówka komunikatu są tworzone w zestawie znaków i kodowaniu systemu lokalnego.

W przypadku przenoszenia komunikatów między systemami czasami zachodzi konieczność przekształcenia danych aplikacji w zestaw znaków i kodowanie wymagane przez system odbierający. Można to zrobić albo z poziomu programów aplikacji w systemie odbierającym, albo przez MCAs w systemie wysyłającym. Jeśli konwersja danych jest obsługiwana w systemie odbierającym, należy użyć programów aplikacji w celu przekształcenia danych aplikacji, a nie w zależności od konwersji, która już wystąpiła w systemie wysyłającym.

Dane aplikacji są przekształcane w program użytkowy po określeniu opcji MQGMO_CONVERT w polu *Options* struktury MQGMO przekazanej do wywołania MQGET, a *wszystkie* są następujące:

- Pola *CodedCharSetId* lub *Encoding* ustawione w strukturze MQMD powiązanej z komunikatem w kolejce różnią się w zależności od pól *CodedCharSetId* lub *Encoding* ustawionych w strukturze MQMD określonej w wywołaniu MQGET.
- Pole *Format* w strukturze MQMD powiązanej z komunikatem nie ma wartości MQFMT_NONE.
- Wartość *BufferLength* podana w wywołaniu MQGET nie jest równa zero.
- Długość danych komunikatu nie jest równa zero.
- Menedżer kolejek obsługuje konwersję między polami *CodedCharSetId* i *Encoding* określonymi w strukturach MQMD powiązanych z komunikatem i wywołaniem MQGET. Szczegółowe informacje na temat obsługiwanych identyfikatorów kodowanego zestawu znaków i kodowania maszynowego można znaleźć w sekcji CodedCharSetId i Encoding .
- Menedżer kolejek obsługuje konwersję formatu komunikatu. Jeśli pole *Format* struktury MQMD powiązanej z komunikatem jest jednym z wbudowanych formatów, menedżer kolejek może przekształcić ten komunikat. Jeśli *Format* nie jest jednym z wbudowanych formatów, należy napisać wyjście konwersji danych, aby przekształcić ten komunikat.

Jeśli wysyłającym MCA jest przekształcanie danych, należy określić słowo kluczowe CONVERT (YES) w definicji każdego kanału nadawczego lub serwera, dla którego wymagana jest konwersja. Jeśli

konwersja danych nie powiedzie się, komunikat jest wysyłany do kolejki DLQ w wysyłającym menedżerze kolejek, a pole *Feedback* struktury MQDLH wskazuje przyczynę. Jeśli komunikat nie może zostać umieszczony w kolejce DLQ, kanał zostanie zamknięty, a nieprzekształcony komunikat pozostaje w kolejce transmisji. Konwersja danych w aplikacjach, a nie przy wysyłaniu MCAs pozwala uniknąć tej sytuacji.

Z reguły dane w komunikacie opisanym jako dane *znakowe* przez wbudowany format lub wyjście konwersji danych są konwertowane z kodowanego zestawu znaków używanego przez komunikat do tego żądania, a pola *liczbowe* są przekształcane na żądane kodowanie.

Więcej szczegółowych informacji na temat konwencji przetwarzania konwersji używanych podczas przekształcania wbudowanych formatów oraz informacje na temat pisania własnych wyjść konwersji danych zawiera sekcja “Pisanie wyjść konwersji danych” na stronie 426. Więcej informacji na temat tabel obsługi języków i obsługiwanych kodowań maszyn można znaleźć w sekcji Języki narodowe i Kodowanie maszyn .

Konwersja znaków nowego wiersza EBCDIC

Jeśli zachodzi potrzeba zapewnienia, że dane wysyłane z platformy EBCDIC do formatu ASCII są identyczne z danymi otrzymanego z powrotem, należy kontrolować konwersję znaków nowego wiersza EBCDIC.

Można to zrobić za pomocą przełącznika zależnego od platformy, który wymusza produkt WebSphere MQ do korzystania z niezmienionych tabel konwersji, ale musi być świadomy niespójnego zachowania, które może spowodować.

Problem pojawia się, ponieważ znak nowego wiersza EBCDIC nie jest konwertowany w sposób spójny przez platformy lub tabele konwersji. W wyniku tego, jeśli dane są wyświetlane na platformie ASCII, formatowanie może być niepoprawne. Może to utrudnić, na przykład, administrowanie systemem IBM i zdalnie z poziomu platformy ASCII za pomocą komendy RUNMQSC.

Więcej informacji na temat konwersji danych w formacie EBCDIC na format ASCII zawiera sekcja Konwersja danych .

Przeglądanie komunikatów w kolejce

W tej sekcji znajdują się informacje na temat przeglądania komunikatów w kolejce przy użyciu wywołania MQGET.

Aby skorzystać z wywołania MQGET w celu przeglądania komunikatów w kolejce:

1. Wywołaj komendę MQOPEN, aby otworzyć kolejkę do przeglądania, określając opcję MQOO_BROWSE.
2. Aby przejrzeć pierwszy komunikat w kolejce, wywołaj komendę MQGET z opcją MQGMO_BROWSE_FIRST. Aby znaleźć odpowiedni komunikat, wywołaj komendę MQGET wielokrotnie z opcją MQGMO_BROWSE_NEXT, aby przejść przez wiele komunikatów.

   Po każdym wywołaniu MQGET w celu wyświetlenia wszystkich komunikatów, *musi* ustawić pola *MsgId* i *CorrelId* struktury MQMD na wartość NULL.
3. Wywołaj komendę MQCLOSE, aby zamknąć kolejkę.

Kursor przeglądania

Po otwarciu (MQOPEN) w kolejce do przeglądania, wywołanie tworzy kursor przeglądania do użycia z wywołaniami MQGET, które korzystają z jednej z opcji przeglądania. Kursor przeglądania jest wyświetlany jako wskaźnik logiczny, który znajduje się przed pierwszym komunikatem w kolejce.

Istnieje możliwość użycia więcej niż jednego kursora przeglądania (z jednego programu) przez wydanie kilku żądań MQOPEN dla tej samej kolejki.

Podczas wywoływania komendy MQGET w celu przeglądania, należy użyć jednej z następujących opcji w strukturze MQGMO:

MQGMO_BROWSE_FIRST
Pobiera kopię pierwszego komunikatu spełniającą warunki określone w strukturze MQMD.

MQGMO_BROWSE_NEXT
Pobiera kopię następnego komunikatu spełniającą warunki określone w strukturze MQMD.

MQGMO_BROWSE_MSG_UNDER_CURSOR
Pobiera kopię komunikatu, który jest aktualnie wskazywał przez kursor, czyli ten, który został ostatnio pobrany przy użyciu opcji MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT.

We wszystkich przypadkach komunikat pozostaje w kolejce.

Po otwarciu kolejki kursor przeglądania jest pozycjonowany logicznie tuż przed pierwszym komunikatem w kolejce. Oznacza to, że jeśli wywołanie MQGET zostanie natychmiast po wywołaniu MQOPEN, można skorzystać z opcji MQGMO_BROWSE_NEXT w celu przeglądania pierwszego komunikatu. Nie trzeba używać opcji MQGMO_BROWSE_FIRST.

Kolejność, w jakiej komunikaty są kopiowane z kolejki, jest określana przez atrybut *MsgDeliverySequence* kolejki. (Więcej informacji na ten temat zawiera sekcja "Kolejność, w jakiej komunikaty są pobierane z kolejki" na stronie 251.)

- "Kolejki w sekwencji FIFO (pierwszy raz w pierwszej kolejności)" na stronie 280
- "Kolejki w kolejności priorytetów" na stronie 280
- "Niezatwierdzone komunikaty" na stronie 280
- "Zmiana kolejności kolejek" na stronie 281
- "Korzystanie z indeksu kolejki" na stronie 281

Kolejki w sekwencji FIFO (pierwszy raz w pierwszej kolejności)

Pierwszym komunikatem w kolejce w tej sekwencji jest komunikat, który był w kolejce najdłuższy.

Użyj komendy MQGMO_BROWSE_NEXT, aby odczytać komunikaty sekwencyjnie w kolejce. Podczas przeglądania wyświetlane są wszystkie komunikaty umieszczone w kolejce, ponieważ kolejka w tej sekwencji ma komunikaty umieszczone na końcu. Gdy kursor rozpozna, że dotarł do końca kolejki, kursor przeglądania pozostaje w miejscu, w którym jest i powraca z parametrem MQRC_NO_MSG_AVAILABLE. Następnie można je pozostawić w oczekiwaniu na dalsze komunikaty lub zresetować na początku kolejki przy użyciu wywołania MQGMO_BROWSE_FIRST.

Kolejki w kolejności priorytetów

Pierwszym komunikatem w kolejce w tej sekwencji jest komunikat, który był w kolejce najdłuższy i ma najwyższy priorytet w czasie, gdy wywołanie MQOPEN zostało wydane.

Użyj komendy MQGMO_BROWSE_NEXT, aby odczytać komunikaty w kolejce.

Kursor przeglądania wskazuje na następny komunikat, pracując z priorytetu pierwszego komunikatu, który ma zostać ukończy z komunikatem o najniższym priorytecie. Powoduje ona przeglądy wszystkich komunikatów umieszczonych w kolejce w tym czasie, o ile są one równe lub mniejsze od priorytetu komunikatu identyfikowanego przez bieżący kursor przeglądania.

Każdy komunikat umieszczony w kolejce o wyższym priorytecie może być przeglądany tylko przez:

- Otwarcie kolejki w celu ponownego przeglądania, w którym to momencie zostanie utworzony nowy kursor przeglądania
- Korzystanie z opcji MQGMO_BROWSE_FIRST

Niezatwierdzone komunikaty

Niezatwierdzony komunikat nigdy nie jest widoczny dla przeglądania; kursor przeglądania przeskoczy obok niego.

Komunikaty w jednostce pracy nie mogą być przeglądane do momentu zatwierdzenia jednostki pracy. Komunikaty nie zmieniają swojej pozycji w kolejce po zatwierdzeniu, tak więc pominięte, niezatwierdzone komunikaty nie będą widoczne, nawet jeśli *są* zatwierdzane, chyba że używana jest opcja MQGMO_BROWSE_FIRST i ponownie działają mimo kolejki.

Zmiana kolejności kolejek

Jeśli sekwencja dostarczania komunikatów została zmieniona z priorytetu na FIFO, a w kolejce znajdują się komunikaty, to kolejność komunikatów, które są już umieszczone w kolejce, nie jest zmieniana. Komunikaty dodane do kolejki później, przyjmują domyślny priorytet kolejki.

Korzystanie z indeksu kolejki

Podczas przeglądania indeksowanej kolejki, która zawiera tylko komunikaty o pojedynczym priorytecie (trwałe lub nietrwałe), menedżer kolejek używa indeksu do przeglądania, gdy używane są pewne formy przeglądania.

Uwaga: Obsługiwane tylko w produkcie WebSphere MQ for z/OS.

Jeśli indeksowana kolejka zawiera tylko komunikaty o pojedynczym priorytecie, używane są dowolne z poniższych form przeglądania:

1. Jeśli kolejka jest indeksowana przez MSGID, przeglądanie żądań, które przekaże identyfikator MSGID w strukturze MQMD, są przetwarzane przy użyciu indeksu w celu znalezienia komunikatu docelowego.
2. Jeśli kolejka jest indeksowana według identyfikatora CORRELID, przeglądanie żądań, które przekaże identyfikator CORRELID w strukturze MQMD, są przetwarzane przy użyciu indeksu w celu znalezienia komunikatu docelowego.
3. Jeśli kolejka jest indeksowana przez parametr GROUPID, przeglądanie żądań, które przekaże identyfikator GROUPID w strukturze MQMD, są przetwarzane przy użyciu indeksu w celu znalezienia komunikatu docelowego.

Jeśli żądanie przeglądania nie przekaże identyfikatora MSGID, CORRELID lub GROUPID w strukturze MQMD, kolejka jest indeksowana i zwracany jest komunikat, musi zostać znaleziony wpis indeksu dla komunikatu oraz informacje w niej używane do aktualizowania kursora przeglądania. Jeśli używany jest szeroki wybór wartości indeksu, nie spowoduje to dodatkowego przetwarzania dodatkowego żądania przeglądania.

Przeglądanie komunikatów, gdy długość komunikatu jest nieznana

Aby przejrzeć komunikat, jeśli nie znasz wielkości komunikatu i nie chcesz używać pól *MsgId*, *CorrelId*lub *GroupId* w celu znalezienia komunikatu, można użyć opcji MQGMO_BROWSE_MSG_UNDER_CURSOR:

1. Wydaj komendę MQGET z:
   - Albo opcja MQGMO_BROWSE_FIRST, albo MQGMO_BROWSE_NEXT
   - Opcja MQGMO_ACCEPT_TRUNCATED_MSG
   - Długość buforu zerowego

   Uwaga: Jeśli istnieje prawdopodobieństwo, że inny program będzie miał ten sam komunikat, należy rozważyć użycie opcji MQGMO_LOCK. Wartość MQRC_TRUNCATED_MSG_ACCEPTED powinna zostać zwrócona.
2. Użyj zwróconego *DataLength* , aby przydzielić potrzebną pamięć masową.
3. Wydaj komendę MQGET z opcją MQGMO_BROWSE_MSG_UNDER_CURSOR.

Komunikat wskazywał, że jest on ostatnim, który został pobrany. Kursor przeglądania nie zostanie przeniesiony. Można wybrać albo zablokować komunikat za pomocą opcji MQGMO_LOCK, albo odblokować zablokowany komunikat za pomocą opcji MQGMO_UNLOCK.

Wywołanie nie powiedzie się, jeśli żadna operacja MQGET z opcjami MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT nie została pomyślnie wykonana, ponieważ kolejka została otwarta.

Usuwanie wiadomości, które zostały przejrzane

Można usunąć z kolejki komunikat, który został już przejrzany, pod warunkiem, że otwarto kolejkę do usuwania komunikatów, jak również do przeglądania. (Należy określić jedną z opcji MQOO_INPUT_ *, a także opcję MQOO_BROWSE w wywołaniu MQOPEN).

Aby usunąć komunikat, wywołaj komendę MQGET ponownie, ale w polu *Options* struktury MQGMO podaj wartość MQGMO_MSG_UNDER_CURSOR.W tym przypadku wywołanie MQGET zignoruje pola *MsgId*, *CorrelId*i *GroupId* struktury MQMD.

W czasie między krokami przeglądania i usuwania, inny program mógł usunąć komunikaty z kolejki, w tym komunikat pod kursorem przeglądania. W tym przypadku wywołanie MQGET zwraca kod przyczyny, aby stwierdzić, że komunikat nie jest dostępny.

Przeglądanie komunikatów w porządku logicznym

"Uporządkowanie logiczne i fizyczne" na stronie 252 wyjaśnia różnicę między logicznym i fizycznym porządkiem komunikatów w kolejce. To rozróżnienie jest szczególnie ważne podczas przeglądania kolejki, ponieważ w ogólnym przypadku komunikaty nie są usuwane, a operacje przeglądania nie muszą zaczynać się od początku kolejki.

Jeśli aplikacja przegląda różne komunikaty jednej grupy (przy użyciu porządku logicznego), ważne jest, aby po zamówieniu logicznym osiągnąć początek następnej grupy, ponieważ ostatni komunikat jednej grupy może wystąpić fizycznie *po* pierwszym komunikacie następnej grupy. Opcja MQGMO_LOGICAL_ORDER zapewnia, że podczas skanowania kolejki następuje porządek logiczny.

Użyj funkcji MQGMO_ALL_MSGS_AVAILABLE (lub MQGMO_ALL_SEGMENTS_AVAILABLE), aby zachować ostrożność podczas przeglądania operacji. Należy rozważyć przypadek komunikatów logicznych za pomocą komendy MQGMO_ALL_MSGS_AVAILABLE. Wynika to z tego, że komunikat logiczny jest dostępny tylko wtedy, gdy wszystkie pozostałe komunikaty w grupie są również obecne. Jeśli tak nie jest, komunikat zostanie przekazany. Może to oznaczać, że gdy brakujące komunikaty nadejdą później, nie są one zauważane przez operację przeglądania.

Na przykład, jeśli obecne są następujące komunikaty logiczne,

```
Logical message 1 (not last) of group 123
Logical message 1 (not last) of group 456
Logical message 2 (last)     of group 456
```

i funkcja przeglądania jest wydawana za pomocą MQGMO_ALL_MSGS_AVAILABLE, zwracany jest pierwszy komunikat logiczny grupy 456, pozostawiając kursor przeglądania w tym komunikacie logicznym. Jeśli zostanie wyświetlony drugi (ostatni) komunikat grupy 123:

```
Logical message 1 (not last) of group 123
Logical message 2 (last)     of group 123
Logical message 1 (not last) of group 456  <=== browse cursor
Logical message 2 (last)     of group 456
```

i ta sama funkcja przeglądania-następna jest wydawana, nie jest zauważona, że grupa 123 jest teraz zakończona, ponieważ pierwszym komunikatem tej grupy jest *przed* kursorem przeglądania.

W niektórych przypadkach (na przykład, jeśli komunikaty są pobierane w sposób destruktywny, gdy grupa jest obecna w całości), można użyć komendy MQGMO_ALL_MSGS_AVAILABLE razem z MQGMO_BROWSE_FIRST. W przeciwnym razie należy powtórzyć skanowanie przeglądania w celu uwzględnienia nowo pojawiających się komunikatów, które zostały pominięte. Po prostu wydanie komendy MQGMO_WAIT razem z MQGMO_BROWSE_NEXT i MQGMO_ALL_MSGS_AVAILABLE nie uwzględnia ich. (To również dzieje się z komunikatami o wyższym priorytecie, które mogą nadejście po zakończeniu skanowania komunikatów).

W następnych sekcjach znajdują się przykłady przeglądania, które dotyczą nieposegmentowanych komunikatów, a segmentowane komunikaty są zgodne z podobnymi zasadami.

Przeglądanie komunikatów w grupach
W tym przykładzie aplikacja przegląda każdy komunikat w kolejce, w kolejności logicznej.

Komunikaty w kolejce mogą być zgrupowane. W przypadku pogrupowanych komunikatów aplikacja nie chce uruchamiać przetwarzania żadnej grupy, dopóki wszystkie komunikaty w jej obrębie nie zostaną odebrane. Wartość MQGMO_ALL_ALL_MSGS_AVAILABLE jest więc określona dla pierwszego komunikatu w grupie; dla kolejnych komunikatów w grupie opcja ta jest zbędna.

W tym przykładzie użyto komendy MQGMO_WAIT. Jednak mimo że oczekiwanie może być spełnione, jeśli pojawi się nowa grupa z przyczyn w programie “Przeglądanie komunikatów w porządku logicznym” na stronie 282, nie jest ona spełniona, jeśli kursor przeglądania przekazał już pierwszy komunikat logiczny w grupie, a pozostałe komunikaty są teraz przesyłane. Mimo to oczekiwanie na odpowiedni okres zapewnia, że aplikacja nie będzie stale zapętlała podczas oczekiwania na nowe komunikaty lub segmenty.

Parametr MQGMO_LOGICAL_ORDER jest używany przez cały czas w celu zapewnienia, że skanowanie jest w porządku logicznym. Kontrastuje to z destrukcyjnym przykładem MQGET, w którym ponieważ każda grupa jest usuwana, MQGMO_LOGICAL_ORDER nie jest używany w przypadku wyszukiwania pierwszego (lub tylko) komunikatu w grupie.

Zakłada się, że bufor aplikacji jest zawsze na tyle duży, aby pomieścić cały komunikat, niezależnie od tego, czy komunikat został podzielony na segmenty. W związku z tym w każdej operacji MQGET określono parametr MQGMO_COMPLETE_MSG.

Poniżej przedstawiono przykład przeglądania komunikatów logicznych w grupie:

```
/* Browse the first message in a group, or a message not in a group */
GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_WAIT
MQGET GMO.MatchOptions = MQMO_MATCH_MSG_SEQ_NUMBER, MD.MsgSeqNumber = 1
/* Examine first or only message */
 ...

GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Examine each remaining message in the group */
   ...
```

Grupa jest powtarzana, dopóki nie zostanie zwrócona wartość MQRC_NO_MSG_AVAILABLE.

Przeglądanie i pobieranie destrukcyjnie
W tym przykładzie aplikacja przegląda wszystkie komunikaty logiczne w grupie przed podjęciem decyzji o tym, czy ta grupa ma być odtwarzająca w sposób destruktywny.

Pierwsza część tego przykładu jest podobna do poprzedniej. Jednak w tym przypadku, po przeglądowaniu całej grupy, decydujemy się na cofanie i wydobycie jej destrukcyjnie.

Ponieważ każda grupa jest usuwana w tym przykładzie, parametr MQGMO_LOGICAL_ORDER nie jest używany w przypadku wyszukiwania pierwszego lub jedynego komunikatu w grupie.

Poniżej przedstawiono przykład przeglądania, a następnie odtwarzania destrukcyjnego:

```
GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MESSAGES_AVAILABE | MQGMO_WAIT
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Examine each remaining message in the group (or as many as
      necessary to decide whether to get it destructively) */
   ...

if ( we want to retrieve the group destructively )

   if ( GroupStatus == ' ' )
      /* We retrieved an ungrouped message */
      GMO.Options = MQGMO_MSG_UNDER_CURSOR | MQGMO_SYNCPOINT
      MQGET GMO.MatchOptions = 0
      /* Process the message */
      ...

   else
      /* We retrieved one or more messages in a group.  The browse cursor */
      /* will not normally be still on the first in the group, so we have */
      /* to match on the GroupId and MsgSeqNumber = 1.                    */
      /* Another way, which works for both grouped and ungrouped messages,*/
      /* would be to remember the MsgId of the first message when it was  */
      /* browsed, and match on that.                                      */
      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT
      MQGET GMO.MatchOptions = MQMO_MATCH_GROUP_ID
                             | MQMO_MATCH_MSG_SEQ_NUMBER,
```

```
          (MQMD.GroupId      = value already in the MD)
           MQMD.MsgSeqNumber = 1
    /* Process first or only message */
    ...

    GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT
                | MQGMO_LOGICAL_ORDER
    do while ( GroupStatus == MQGS_MSG_IN_GROUP )
       MQGET
       /* Process each remaining message in the group */
       ...
```

Unikanie powtarzającej się dostawy przejrzanych wiadomości

Korzystając z niektórych opcji otwierania i pobierania wiadomości, można oznaczyć wiadomości jako przeglądane, tak aby nie były one ponownie pobierane przez bieżące lub inne aplikacje współpracujące. Komunikaty mogą być nieoznaczone jawnie lub automatycznie, aby były ponownie dostępne do przeglądania.

Jeśli komunikaty są przeglądane w kolejce, można je pobrać w innej kolejności, w kolejności, w jakiej zostały pobrane, jeśli zostały one destrukcyjne. W szczególności można przeglądać ten sam komunikat wiele razy, co nie jest możliwe, jeśli jest on usuwany z kolejki. Aby tego uniknąć, można *oznaczyć* komunikaty w miarę ich przeglądania, a także unikać pobierania oznaczonych wiadomości. Czasami jest to określane jako *przeglądanie z zaznaczonym znakiem*. Aby oznaczyć przeglądane komunikaty, użyj opcji get message MQGMO_MARK_BROWSE_HANDLE, a w celu pobrania tylko komunikatów, które nie są oznaczone, użyj komendy MQGMO_UNMARKED_BROWSE_MSG. W przypadku użycia kombinacji opcji MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG i MQGMO_MARK_BROWSE_HANDLE i powtórzonych operacji MQGET, każdy komunikat w kolejce zostanie pobrany z kolei. Zapobiega to powtarzającej się dostarczania komunikatów, nawet jeśli używana jest komenda MQGMO_BROWSE_FIRST w celu zapewnienia, że komunikaty nie zostaną pominięte. Ta kombinacja opcji może być reprezentowana przez pojedynczą stałą MQGMO_BROWSE_HANDLE. Jeśli w kolejce nie ma komunikatów, które nie zostały przejrzane, zwracana jest wartość MQRC_NO_MSG_AVAILABLE.

Jeśli wiele aplikacji przegląda tę samą kolejkę, może otworzyć kolejkę z opcjami MQOO_CO_OP i MQOO_BROWSE. Uchwyt obiektu zwracany przez każdą operację MQOPEN uważa się za część współpracującej grupy. Dowolny komunikat zwrócony przez wywołanie MQGET, określający opcję MQGMO_MARK_BROWSE_CO_OP, jest uznawana za oznaczoną dla tego współpracującego zestawu uchwytów.

Jeśli komunikat został oznaczony przez pewien czas, może on być automatycznie nieoznaczony przez menedżer kolejek i udostępniony do ponownego przeglądania. Atrybut MsgMarkmenedżera kolejekBrowseInterval określa czas (w milisekundach), przez który komunikat ma pozostać oznaczony jako odpowiedni dla współpracującego zestawu uchwytów. Wartość MsgMarkBrowseInterval równa -1 oznacza, że komunikaty nigdy nie są automatycznie nieoznaczone.

Gdy pojedynczy proces lub zestaw kooperatywnych procesów zaznacza komunikaty zatrzymują się, wszystkie oznaczone komunikaty stają się nieoznaczone.

Przykłady przeglądania kooperatywnego

Istnieje możliwość uruchomienia wielu kopii aplikacji programu rozsyłającego w celu przeglądania komunikatów w kolejce i zainicjowania konsumenta na podstawie treści każdego komunikatu. W każdym programie rozsyłającym otwórz kolejkę za pomocą komendy MQOO_CO_OP. Oznacza to, że programy rozsyłające współpracują ze sobą i będą mieć świadomość, że są one oznaczone jako komunikaty. Następnie każdy program rozsyłający wykonuje powtarzające się wywołania MQGET, określając opcje MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG i MQGMO_MARK_BROWSE_CO_OP (można użyć pojedynczej stałej MQGMO_BROWSE_CO_OP, aby reprezentować tę kombinację opcji). Każda aplikacja programu rozsyłającego pobiera następnie tylko te komunikaty, które nie zostały jeszcze oznaczone przez inne współpracujące programy rozsyłające. Program rozsyłający inicjuje konsumenta i przekazuje element MsgToken zwracany przez komendę MQGET do konsumenta, co w sposób destruktywny pobiera komunikat z kolejki. Jeśli konsument wycofał operację MQGET komunikatu, komunikat jest dostępny dla jednej z przeglądarek w celu ponownego wysłania, ponieważ

nie jest już oznaczony. Jeśli konsument nie wykonuje operacji MQGET w komunikacie, po upływie wartości MsgMarkBrowseInterval , menedżer kolejek odznaczy komunikat dla współpracującego zestawu uchwytów i może zostać ponownie rozesłany.

Zamiast wielu kopii tej samej aplikacji programu rozsyłającego, użytkownik może mieć kilka różnych aplikacji programu rozsyłającego, które przeglądają kolejkę, a każda z nich jest odpowiednia do przetwarzania podzbioru komunikatów w kolejce. W każdym programie rozsyłającym otwórz kolejkę za pomocą komendy MQOO_CO_OP. Oznacza to, że programy rozsyłające współpracują ze sobą i będą mieć świadomość, że są one oznaczone jako komunikaty.

- Jeśli kolejność przetwarzania komunikatów dla pojedynczego przekaźnika jest ważna, każdy program rozsyłający wykonuje powtarzające się wywołania MQGET, określając opcje MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG i MQGMO_MARK_BROWSE_HANDLE (lub MQGMO_BROWSE_HANDLE). Jeśli przejrzany komunikat jest odpowiedni dla tego programu rozsyłającego do przetwarzania, to wywoła wywołanie MQGET z określeniem wartości MQMO_MATCH_MSG_TOKEN, MQGMO_MARK_BROWSE_CO_OP i MsgToken zwróconego przez poprzednie wywołanie MQGET. Jeśli wywołanie powiedzie się, program rozsyłający inicjuje konsumenta, przekazując do niego element MsgToken .
- Jeśli kolejność przetwarzania komunikatów nie jest istotna, a oczekuje się, że program rozsyłający będzie przetwarzać większość komunikatów, które napotyka, należy użyć opcji MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG i MQGMO_MARK_BROWSE_CO_OP (lub MQGMO_BROWSE_CO_OP). Jeśli program rozsyłający przegląda komunikat, którego nie można przetworzyć, usuń ten komunikat, wywołując komendę MQGET z opcją MQMO_MATCH_MSG_TOKEN, MQGMO_UNMARK_BROWSE_CO_OP i MsgToken zwróconej poprzednio.

Niektóre przypadki, w których wywołanie MQGET nie powiodło się

Jeśli niektóre atrybuty kolejki zostaną zmienione za pomocą opcji FORCE w komendzie między wywołaniem wywołania MQOPEN i wywołaniem MQGET, wywołanie MQGET nie powiedzie się i zwróci kod przyczyny MQRC_OBJECT_CHANGED.

Menedżer kolejek oznacza, że uchwyt obiektu nie jest już poprawny. Dzieje się tak również wtedy, gdy zmiany będą miały zastosowanie do każdej kolejki, do której nazwa kolejki jest tłumaczona. Atrybuty, które mają wpływ na uchwyt w ten sposób, są wymienione w opisie wywołania MQOPEN w produkcie MQOPEN. Jeśli wywołanie zwróci kod przyczyny MQRC_OBJECT_CHANGED, zamknij kolejkę, ponownie go otwórz, a następnie spróbuj ponownie uzyskać komunikat.

Jeśli operacje pobierania są zablokowane dla kolejki, z której podjęto próbę pobrania komunikatów (lub dowolnej kolejki, do której nazwa kolejki jest tłumaczona), wywołanie MQGET nie powiedzie się i zwróci kod przyczyny MQRC_GET_INHIBITED. Zdarza się to nawet w przypadku korzystania z wywołania MQGET w celu przeglądania. Może być możliwe pomyślne pobranie komunikatu w przypadku podjęcia próby wywołania MQGET w późniejszym czasie, jeśli projekt aplikacji jest taki, że inne programy regularnie zmieniają atrybuty kolejek.

Jeśli usunięto kolejkę dynamiczną (tymczasową lub trwałą), wywołania MQGET z użyciem wcześniej uzyskanego uchwytu obiektu nie powiodą się i zwróć kod przyczyny MQRC_Q_DELETED.

Zapisywanie aplikacji publikowania/subskrypcji

Rozpocznij pisanie aplikacji WebSphere MQ publikowania/subskrypcji.

Przegląd pojęć związanych z publikowaniem/subskrybowaniem zawiera sekcja Wprowadzenie do przesyłania komunikatów w trybie publikowania/subskrypcji produktu WebSphere MQ.

Informacje na temat pisania różnych typów aplikacji publikowania/subskrypcji można znaleźć w następujących tematach:

- “Zapisywanie aplikacji publikatorów” na stronie 286
- “Pisanie aplikacji subskrybentów” na stronie 293
- “Cykle życia publikowania/subskrybowania” na stronie 311
- “Właściwości komunikatu publikowania/subskrypcji” na stronie 316

- “Porządkowanie komunikatów” na stronie 318
- “Przechwytywanie publikacji” na stronie 318
- “Opcje publikowania” na stronie 326
- “Opcje subskrypcji” na stronie 326

Pojęcia pokrewne

“Pojęcia związane z projektowaniem aplikacji” na stronie 8
Do pisania aplikacji IBM WebSphere MQ można użyć wyboru języków proceduralnych lub obiektowych. Odsyłacze w tym temacie można znaleźć w informacjach dotyczących pojęć związanych z produktem IBM WebSphere MQ , które są przydatne dla programistów aplikacji.

“Wybór języka programowania do użycia” na stronie 80
W tej sekcji znajdują się informacje na temat języków programowania i środowisk obsługiwanych przez produkt IBM WebSphere MQ, a także niektóre zagadnienia dotyczące ich używania.

“Projektowanie aplikacji produktu IBM WebSphere MQ” na stronie 91
Po zdecydowaniu się, w jaki sposób aplikacje mogą korzystać z platform i środowisk, które są dostępne dla użytkownika, należy zdecydować, w jaki sposób korzystać z funkcji oferowanych przez produkt WebSphere MQ.

“Przykładowe programy produktu WebSphere MQ” na stronie 98
Ta kolekcja tematów zawiera informacje na temat przykładowych programów WebSphere MQ na różnych platformach.

“Pisanie aplikacji kolejkowania” na stronie 199
Ta sekcja zawiera informacje na temat pisania aplikacji kolejkowania, łączenia się i rozłączania z menedżerem kolejek, publikowania/subskrypcji oraz obiektów otwierających i zamykających.

“Pisanie aplikacji klienckich” na stronie 360
Co należy wiedzieć, aby pisać aplikacje klienckie w produkcie WebSphere MQ.

“Korzystanie z usług Web Service w produkcie WebSphere MQ” na stronie 972
Aplikacje produktu IBM WebSphere MQ dla usług Web Services można tworzyć przy użyciu transportu IBM WebSphere MQ dla protokołu SOAP lub mostu IBM WebSphere MQ dla protokołu HTTP.

“Budowanie aplikacji IBM WebSphere MQ” na stronie 439
Ta sekcja zawiera informacje na temat budowania aplikacji produktu IBM WebSphere MQ na różnych platformach.

“Obsługa błędów programu” na stronie 561
Te informacje wyjaśniają błędy związane z wywołaniami MQI aplikacji, gdy wywołuje połączenie, lub gdy jego komunikat jest dostarczany do miejsca docelowego.

Zapisywanie aplikacji publikatorów

Rozpoczęcie pracy z pisaniem aplikacji publikatorów poprzez zapoznanie się z dwoma przykładami. Pierwsza z nich jest możliwie najdokładniej modelowana w punkcie, w którym aplikacja wskazuje na umieszczanie komunikatów w kolejce, a druga demonstruje dynamicznie tworzenie tematów-bardziej powszechny wzorzec dla aplikacji publikatorów.

Pisanie prostej aplikacji publikatora produktu WebSphere MQ jest tak samo, jak napisanie punktu aplikacji produktu WebSphere MQ do punktu, w którym komunikaty są umieszczane w kolejce (Tabela 41 na stronie 286). Różnica polega na tym, że komunikaty MQPUT są wysyłane do tematu, a nie do kolejki.

Tabela 41. Wskaż wzorzec programu WebSphere MQ dla publikowania/subskrypcji.

Krok	**Punkt z punktem MQ Call**	**Publikuj wywołanie MQ**
Połącz z menedżerem kolejek	MQCONN	MQCONN
Otwarta kolejka	MQOPEN	
Otwórz wątek		MQOPEN

Tabela 41. Wskaż wzorzec programu WebSphere MQ dla publikowania/subskrypcji. (kontynuacja)		
Krok	**Punkt z punktem MQ Call**	**Publikuj wywołanie MQ**
Umieść komunikat (y)	MQPUT	MQPUT
Zamknij temat		MQCLOSE
Zamknij kolejkę	MQCLOSE	
Rozłącz z menedżerem kolejek	MQDISC	MQDISC

Aby zrobić ten konkretny, istnieją dwa przykłady zastosowań do publikowania cen akcji. W pierwszym przykładzie (“Przykład 1: publikator w stałym temacie” na stronie 287), który jest ściśle modelowany przy umieszczanie komunikatów w kolejce, administrator tworzy definicję tematu w podobny sposób, aby utworzyć kolejkę. Programmer kodów MQPUT służy do zapisywania komunikatów w temacie, a nie do zapisywania ich w kolejce. W drugim przykładzie (“Przykład 2: publikator w temacie o zmiennej” na stronie 290) wzorzec interakcji programu z produktem WebSphere MQ jest podobny. Różnica polega na tym, że programista udostępnia temat, do którego jest zapisywany komunikat, a nie przez administratora. W praktyce oznacza to zwykle, że łańcuch tematu jest zdefiniowany przez treść lub jest udostępniany przez inne źródło, takie jak dane wprowadzane przez użytkownika przez przeglądarkę.

Pojęcia pokrewne

“Pisanie aplikacji subskrybentów” na stronie 293
Rozpoczynanie pracy z pisaniem aplikacji subskrybentów przez zapoznanie się z trzema przykładami: aplikacja WebSphere MQ wykorzystująca komunikaty z kolejki, aplikacja tworująca subskrypcję i nie wymaga znajomości kolejkowania, a w końcu przykład, który używa zarówno kolejkowania, jak i subskrypcji.

Odsyłacze pokrewne

ZDEFINIUJ TEMAT
WYŚWIETL TEMAT
WYŚWIETL STATUS TPSTATUS

Przykład 1: publikator w stałym temacie

Program WebSphere MQ w celu zilustrować publikowanie w temacie zdefiniowanym administracyjnie.

Uwaga: Kompaktowy styl kodowania jest przeznaczony do czytelności, a nie do użytku produkcyjnego.

Patrz dane wyjściowe w programie Rysunek 38 na stronie 288 .

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
int main(int argc, char **argv)
{
    char     topicNameDefault[]   = "IBMSTOCKPRICE";
    char     publicationDefault[] = "129";
    MQCHAR48 qmName = "";

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle                    */
    MQHOBJ   Hobj = MQHO_NONE;            /* object handle sub queue              */
    MQLONG   CompCode = MQCC_OK;          /* completion code                      */
    MQLONG   Reason = MQRC_NONE;          /* reason code                          */
    MQOD     td = {MQOD_DEFAULT};         /* Object descriptor                    */
    MQMD     md = {MQMD_DEFAULT};         /* Message Descriptor                   */
    MQPMO   pmo = {MQPMO_DEFAULT};        /* put message options                  */
    MQCHAR   resTopicStr[151];            /* Returned vale of topic string        */
    char *   topicName   = topicNameDefault;
    char *   publication = publicationDefault;
    memset   (resTopicStr, 0 , sizeof(resTopicStr));

    switch(argc){                         /* replace defaults with args if provided */
        default:
            publication = argv[2];
        case(2):
            topicName = argv[1];
        case(1):
            printf("Optional parameters:  TopicObject Publication\n");
    }
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        td.ObjectType = MQOT_TOPIC;        /* Object is a topic                    */
        td.Version = MQOD_VERSION_4;       /* Descriptor needs to be V4            */
        strncpy(td.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        td.ResObjectString.VSPtr = resTopicStr;
        td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
        MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
        MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode,
&Reason);
        if (CompCode != MQCC_OK) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    if (CompCode == MQCC_OK)
        printf("Published \"%s\" using topic \"%s\" to topic string \"%s\"\n",
               publication, td.ObjectName, resTopicStr);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
```

Rysunek 37. Prosty publikator produktu WebSphere MQ do stałego tematu.

```
X:\Publish1\Debug>PublishStock
Optional parameters:  TopicObject Publication
Published "129" using topic "IBMSTOCKPRICE" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0

X:\Publish1\Debug>PublishStock IBMSTOCKPRICE 155
Optional parameters:  TopicObject Publication
Published "155" using topic "IBMSTOCKPRICE" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0
```

Rysunek 38. Przykładowe dane wyjściowe z pierwszego przykładu publikatora

Następujące wybrane wiersze kodu ilustrują aspekty pisania aplikacji publikatora dla produktu WebSphere MQ.

char topicNameDefault[] = "IBMSTOCKPRICE";
Domyślna nazwa tematu jest zdefiniowana w programie. Można go nadpisać, podając nazwę innego obiektu tematu jako pierwszy argument dla programu.

MQCHAR resTopicStr[151];
Produkt `resTopicStr` jest wskazyany przez produkt `td.ResObjectString.VSPtr` i jest używany przez produkt `MQOPEN` do zwracania przetłumaczanego łańcucha tematu. Ustaw długość `resTopicStr` o jedną większą niż długość, która została przekazana w programie `td.ResObjectString.VSBufSize` , aby zapewnić miejsce na zerowe zakończenie.

memset (resTopicStr, 0, sizeof(resTopicStr));
Zainicjuj parametr `resTopicStr` w celu usunięcia wartości NULL, aby upewnić się, że rozstrzygnięty łańcuch tematu zwrócony w polu `MQCHARV` ma wartość NULL.

td.ObjectType = MQOT_TOPIC
Istnieje nowy typ obiektu dla publikowania/subskrypcji: *obiekt tematu*.

td.Version = MQOD_VERSION_4;
Aby korzystać z nowego typu obiektu, należy użyć co najmniej *wersji 4* deskryptora obiektu.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
`topicName` jest nazwą obiektu tematu, czasami nazywanych obiektem tematu administracyjnego. W tym przykładzie należy wcześniej utworzyć obiekt tematu przy użyciu programu WebSphere MQ Explorer lub tej komendy MQSC,

```
DEFINE TOPIC(IBMSTOCKPRICE) TOPICSTR(NYSE/IBM/PRICE) REPLACE;
```

td.ResObjectString.VSPtr = resTopicStr;
Rozstrzygnięty łańcuch tematu jest odbity w finalnym `printf` w programie. Skonfiguruj strukturę produktu `MQCHARV ResObjectString` dla produktu WebSphere MQ , aby przywrócić rozstrzygnięty łańcuch z powrotem do programu.

MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
Otwórz temat dla danych wyjściowych, tak jak otwieranie kolejki dla danych wyjściowych.

pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
Użytkownik chce, aby nowi subskrybenci mieli możliwość otrzymywania publikacji, a także poprzez podanie wartości `MQPMO_RETAIN` w publikatorze po uruchomieniu subskrybenta, który otrzymuje najnowszą publikację, opublikowaną przed uruchomieniem subskrybenta, jako pierwszą zgodną z nią publikację. Alternatywą jest udostępnienie subskrybentom publikacji opublikowanych dopiero po uruchomieniu subskrybenta. Dodatkowo subskrybent ma możliwość odrzucenia zachowanej publikacji przez określenie wartości `MQSO_NEW_PUBLICATIONS_ONLY` w subskrypcji.

MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
Dodaj 1 do długości łańcucha przekazanego do programu `MQPUT` , aby przekazać znak kończący o wartości NULL do produktu WebSphere MQ jako część buforu komunikatów.

Co przedstawia pierwszy przykład? Przykład naśladuje możliwie najdokładniej wypróbowany i sprawdzony tradycyjny wzorzec do pisania punktu do programów WebSphere MQ . Istotną cechą wzorca programowania produktu WebSphere MQ jest fakt, że programista nie jest zaniepokojony, gdy wysyłane są komunikaty. Zadaniem programisty jest połączenie się z menedżerem kolejek i przekazanie mu komunikatów, które mają być dystrybuowane do odbiorców. W paradygmach punkt-punkt programista otwiera kolejkę (prawdopodobnie kolejkę aliasową), którą skonfigurował administrator. Kolejka aliasowa kieruje komunikaty do kolejki docelowej albo w lokalnym menedżerze kolejek, albo do menedżera kolejek zdalnych. Podczas gdy komunikaty oczekują na dostarczenie, są one przechowywane w kolejkach między źródłem a miejscem docelowym.

W przypadku wzorca publikowania/subskrypcji, zamiast otwierania kolejki programista otwiera temat. W naszym przykładzie temat jest powiązany z łańcuchem tematu przez administratora. Menedżer kolejek przekazuje publikację, korzystając z kolejek, do lokalnych lub zdalnych subskrybentów, którzy mają subskrypcje zgodne z łańcuchem tematu publikacji. Jeśli publikacje są zachowywane, menedżer kolejek przechowuje najnowszą kopię publikacji, nawet jeśli nie ma już żadnych subskrybentów. Zachowana

publikacja jest dostępna do przekazania do przyszłych subskrybentów. Aplikacja publikująca nie odgrywa żadnej części w wyborze lub kierowaniu publikacji do miejsca docelowego; jej zadaniem jest tworzenie i umieszczanie publikacji na tematy zdefiniowane przez administratora.

Ten przykład stałego tematu jest nietypowy dla wielu aplikacji publikowania/subskrypcji: jest on statyczny. Wymaga on od administratora zdefiniowania łańcuchów tematów i zmiany tematów, które są publikowane. Często aplikacje publikowania/subskrybowania muszą znać niektóre lub wszystkie drzewa tematów. Być może tematy często zmieniają się lub być może mimo że tematy nie zmieniają się zbyt wiele, liczba kombinacji tematów jest duża i zbyt uciążliwe dla administratora, aby zdefiniować węzeł tematu dla każdego łańcucha tematu, który może wymagać opublikowania. Być może łańcuchy tematów nie są znane przed publikacją. Aplikacja publikująca może używać informacji z treści publikacji w celu określenia łańcucha tematu lub może zawierać informacje o łańcuchach tematów do opublikowania w innym źródle, takim jak dane wprowadzane przez użytkownika z przeglądarki. Aby uzyskać bardziej dynamiczne style publikowania, w następnym przykładzie przedstawiono sposób dynamicznego tworzenia tematów w ramach aplikacji publikatora.

Wątki publikatorów i subskrybentów razem. Projektowanie reguł lub architektury na potrzeby nazewnictwa tematów oraz organizowanie ich w drzewach tematów jest ważnym krokiem w tworzeniu rozwiązania publikowania/subskrypcji. Należy uważnie przyjrzeć się, w jakim stopniu organizacja drzewa tematów łączy programy publikatorów i subskrybentów, a następnie wiąże je z treścią drzewa tematów. Zadaj sobie pytanie, czy zmiany w drzewie tematów wpływają na aplikacje publikatorów i subskrybentów oraz jak można zminimalizować efekt. Wbudowana architektura modelu publikowania/subskrypcji produktu WebSphere MQ jest pojęciem obiektu tematu administracyjnego, który udostępnia część główną lub poddrzewo główne tematu. Obiekt tematu umożliwia zdefiniowanie głównej części drzewa tematów w sposób administracyjny, który upraszcza programowanie aplikacji i operacje, a co za tym samym zwiększa łatwość konserwacji. Na przykład w przypadku wdrażania wielu aplikacji publikowania/subskrybowania, które mają izolowane drzewa tematów, a następnie administracyjnie definiując główną część drzewa tematów, można zagwarantować odseparowanie drzew tematów, nawet jeśli nie ma spójności konwencji nazewnictwa tematów przyjętych przez różne aplikacje.

W praktyce aplikacje wydawcy obejmują spektrum od wyłącznie korzystania z stałych tematów, jak w tym przykładzie, i tematów zmiennych, jak w następnym. Program “Przykład 2: publikator w temacie o zmiennej” na stronie 290 demonstruje również połączenie użycia tematów i łańcuchów tematów.

Pojęcia pokrewne

“Przykład 2: publikator w temacie o zmiennej” na stronie 290
Program produktu Websphere MQ w celu zilustronia publikowania w programowo zdefiniowanym temacie.

“Pisanie aplikacji subskrybentów” na stronie 293
Rozpoczynanie pracy z pisaniem aplikacji subskrybentów przez zapoznanie się z trzema przykładami: aplikacja WebSphere MQ wykorzystująca komunikaty z kolejki, aplikacja tworująca subskrypcję i nie wymaga znajomości kolejkowania, a w końcu przykład, który używa zarówno kolejkowania, jak i subskrypcji.

Przykład 2: publikator w temacie o zmiennej

Program produktu Websphere MQ w celu zilustronia publikowania w programowo zdefiniowanym temacie.

Uwaga: Kompaktowy styl kodowania jest przeznaczony do czytelności, a nie do użytku produkcyjnego.

Patrz dane wyjściowe w programie Rysunek 40 na stronie 292.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
int main(int argc, char **argv)
{
    char     topicNameDefault[]   = "STOCKS";
    char     topicStringDefault[] = "IBM/PRICE";
    char     publicationDefault[] = "130";
    MQCHAR48 qmName = "";

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle           */
    MQHOBJ   Hobj     = MQHO_NONE;   /* object handle sub queue          */
    MQLONG   CompCode = MQCC_OK;     /* completion code                  */
    MQLONG   Reason   = MQRC_NONE;   /* reason code                      */
    MQOD     td = {MQOD_DEFAULT};    /* Object descriptor                */
    MQMD     md = {MQMD_DEFAULT};    /* Message Descriptor               */
    MQPMO   pmo = {MQPMO_DEFAULT};   /* put message options              */
    MQCHAR   resTopicStr[151];       /* Returned value of topic string */
    char *   topicName   = topicNameDefault;
    char *   topicString = topicStringDefault;
    char *   publication = publicationDefault;
    memset   (resTopicStr, 0 , sizeof(resTopicStr));

    switch(argc){            /* Replace defaults with args if provided */
        default:
            publication = argv[3];
        case(3):
            topicString = argv[2];
        case(2):
            if (strcmp(argv[1],"/"))  /* "/" invalid = No topic object  */
                topicName = argv[1];
            else
                *topicName = '\0';
        case(1):
            printf("Provide parameters:  TopicObject TopicString Publication\n");
    }

    printf("Publish \"%s\" to topic \"%-.48s\" and topic string \"%s\"\n", publication,
topicName, topicString);
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        td.ObjectType = MQOT_TOPIC;    /* Object is a topic             */
        td.Version = MQOD_VERSION_4;   /* Descriptor needs to be V4     */
        strncpy(td.ObjectName, topicName,  MQ_TOPIC_NAME_LENGTH);
        td.ObjectString.VSPtr = topicString;
        td.ObjectString.VSLength = (MQLONG)strlen(topicString);
        td.ResObjectString.VSPtr = resTopicStr;
        td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
        MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
        MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode,
&Reason);
        if (CompCode != MQCC_OK) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    if (CompCode == MQCC_OK)
        printf("Published \"%s\" to topic string \"%s\"\n",  publication, resTopicStr);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
```

Rysunek 39. Prosty publikator produktu WebSphere MQ do tematu zmiennej.

```
X:\Publish2\Debug>PublishStock
Provide parameters:  TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0

X:\Publish2\Debug>PublishStock / NYSE/IBM/PRICE 131
Provide parameters:  TopicObject TopicString Publication
Publish "131" to topic "" and topic string "NYSE/IBM/PRICE"
Published "131" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0
```

Rysunek 40. Przykładowe dane wyjściowe z drugiego przykładu publikatora

Jest kilka punktów, które warto zwrócić uwagę na ten przykład.

char topicNameDefault[] = "STOCKS";
Domyślna nazwa tematu STOCKS definiuje część łańcucha tematu. Tę nazwę tematu można przesłonić, udostępniając go jako pierwszy argument w programie lub eliminując użycie nazwy tematu przez podanie / jako pierwszego parametru.

char topicString[101] = "IBM/PRICE";
IBM/PRICE jest domyślnym łańcuchem tematu. Ten łańcuch tematu można przesłonić, udostępniając go jako drugi argument programu.

Menedżer kolejek łączy łańcuch tematu udostępniany przez obiekt tematu STOCKS "NYSE"z łańcuchem tematu udostępnionym przez program "IBM/PRICE" i wstawia element "/" między dwoma łańcuchami tematów. Wynikiem jest rozstrzygnięty łańcuch tematu "NYSE/IBM/PRICE". Wynikowy łańcuch tematu jest taki sam, jak ten zdefiniowany w obiekcie tematu IBMSTOCKPRICE i ma on dokładnie taki sam efekt.

Obiekt tematu administracyjnego powiązany z rozstrzygniętym łańcuchem tematu nie musi być tym samym obiektem tematu, który został przekazany do produktu MQOPEN przez publikatora. Produkt WebSphere MQ korzysta z drzewa niejawnego w rozstrzygniętym łańcuchu tematu, aby określić, który administracyjny obiekt tematu definiuje atrybuty powiązane z publikacją.

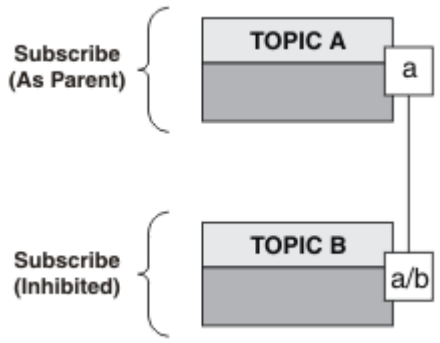
Załóżmy, że istnieją dwa obiekty tematów A i B, a A definiuje temat "a", a B definiuje temat "a/b" (Rysunek 41 na stronie 293). Jeśli program publikujący odwołuje się do obiektu tematu A i zawiera łańcuch tematu "b", rozstrzygając temat w łańcuchu tematu "a/b", opublikowanie dziedziczy jego właściwości z obiektu tematu B , ponieważ temat jest zgodny z łańcuchem tematu "a/b" zdefiniowanym dla produktu B.

if (strcmp(argv[1],"/"))
argv[1] jest opcjonalnie udostępnionym topicName. "/" jest niepoprawna jako nazwa tematu. Oznacza to, że nie ma żadnej nazwy tematu, a łańcuch tematu jest w całości udostępniany przez program. Dane wyjściowe w programie Rysunek 40 na stronie 292 przedstawiają cały łańcuch tematu, który jest dostępny dynamicznie przez program.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
W przypadku domyślnego elementu pracy należy wcześniej utworzyć opcjonalną topicName , korzystając z programu WebSphere MQ Explorer lub komendy MQSC:

```
DEFINE TOPIC(STOCKS) TOPICSTR(NYSE) REPLACE;
```

td.ObjectString.VSPtr = topicString;
Łańcuch tematu to pole MQCHARV w deskryptorze tematu.



Rysunek 41. Powiązania obiektów tematu

Co ilustruje drugi przykład? Chociaż kod jest bardzo podobny do pierwszego przykładu-skutecznie istnieją tylko dwie linie różnicy-wynik jest zdecydowanie odmiennym programem do pierwszego. Programista kontroluje miejsca docelowe, do których wysyłane są publikacje. W połączeniu z minimalnym wejściem administratora używanym do projektowania aplikacji subskrybentów, żadne tematy ani kolejki nie muszą być wstępnie zdefiniowane w celu kierowania publikacji z publikatorów do subskrybentów.

W paradygmach przesyłania komunikatów w trybie punkt z punktem należy zdefiniować kolejki, zanim komunikaty będą mogły przepływać. W przypadku publikowania/subskrypcji nie są one dostępne, chociaż produkt WebSphere MQ implementuje interfejs publikowania/subskrypcji przy użyciu bazowego systemu kolejkowania. Korzyści wynikające z gwarantowanego dostarczania, transakcyjności i luźnego połączenia powiązanego z przesyłaniem komunikatów i kolejkowaniem są dziedziczone przez aplikacje publikowania/subskrypcji.

Projektant musi zdecydować, czy publikator i subskrybent mają mieć świadomość bazowego drzewa tematów, czy też nie, a także czy programy subskrybentów są świadome kolejkowania, czy nie. Następnie należy zbadać przykładowe aplikacje subskrybenta. Są one przeznaczone do użycia z przykładami publikatora, zwykle publikowaniem i subskrybowaniem produktu `NYSE/IBM/PRICE`.

Pojęcia pokrewne

“Przykład 1: publikator w stałym temacie” na stronie 287
Program WebSphere MQ w celu zilustrować publikowanie w temacie zdefiniowanym administracyjnie.

“Pisanie aplikacji subskrybentów” na stronie 293
Rozpoczynanie pracy z pisaniem aplikacji subskrybentów przez zapoznanie się z trzema przykładami: aplikacja WebSphere MQ wykorzystująca komunikaty z kolejki, aplikacja tworująca subskrypcję i nie wymaga znajomości kolejkowania, a w końcu przykład, który używa zarówno kolejkowania, jak i subskrypcji.

Pisanie aplikacji subskrybentów

Rozpoczynanie pracy z pisaniem aplikacji subskrybentów przez zapoznanie się z trzema przykładami: aplikacja WebSphere MQ wykorzystująca komunikaty z kolejki, aplikacja tworująca subskrypcję i nie wymaga znajomości kolejkowania, a w końcu przykład, który używa zarówno kolejkowania, jak i subskrypcji.

W produkcie Tabela 42 na stronie 294 wyświetlane są trzy style konsumenta lub subskrybenta, wraz z sekwencjami wywołań funkcji produktu WebSphere MQ , które je charakteryzują.

1. Pierwszy styl, MQ Publication Consumer, jest taki sam, jak w przypadku programu MQ , który wykonuje tylko MQGET. Aplikacja nie ma wiedzy, że jest konsumowana w publikacjach-po prostu odczytuje wiadomości z kolejki. Subskrypcja, która powoduje, że publikacje są kierowane do kolejki, jest tworzona administracyjnie przy użyciu programu WebSphere MQ Explorer lub komendy.
2. Drugi styl jest preferowanym wzorcem dla większości aplikacji subskrybentów. Aplikacja subskrybenta tworzy subskrypcję, a następnie uzyskuje publikacje. Zarządzanie kolejkami jest wykonywane przez menedżer kolejek.

3. W trzecim stylu aplikacja subskrybenta zdecyduje się na otwarcie i zamknięcie kolejki bazowej, która jest używana do publikacji, a także do emisji subskrypcji w celu wypełnienia kolejki publikacjami.

Jednym ze sposobów zrozumienia tych stylów jest zapoznanie się z przykładowymi programami C wymienionymi w programie Tabela 42 na stronie 294 dla każdego ze stylów. Przykłady zostały zaprojektowane w taki sposób, aby były uruchamiane w połączeniu z przykładem publikatora znalezionym w produkcie “Zapisywanie aplikacji publikatorów” na stronie 286 .

Tabela 42. Wskaż wzorce programu WebSphere MQ w zależności od punktu.

Krok	Konsument komunikatów MQ	“Przykład 1: konsument publikacji MQ” na stronie 294	“Przykład 2: zarządzany subskrybent MQ” na stronie 297	“Przykład 3: Niezarządzany subskrybent MQ” na stronie 302
Połącz z menedżerem kolejek	MQCONN	MQCONN	MQCONN	MQCONN
Otwarta kolejka	MQOPEN	MQOPEN		MQOPEN
Subskrybowanie			MQSUB	MQSUB
Pobierz komunikat (y)	MQGET	MQGET	MQGET	MQGET
Zamknij kolejkę	MQCLOSE	MQCLOSE	(MQCLOSE)	MQCLOSE
Zamknij subskrypcję			MQCLOSE	MQCLOSE
Rozłącz z menedżerem kolejek	MQDISC	MQDISC	MQDISC	MQDISC

Użycie komendy MQCLOSE jest zawsze opcjonalne, aby zwolnić zasoby, przekazać opcje MQCLOSE lub tylko dla symetrii z opcją MQOPEN. Ponieważ nie jest konieczne określenie opcji MQCLOSE w sytuacji, gdy kolejka subskrypcji jest zamknięta w ramach subskrypcji zarządzanej MQ , a argument symetrii nie jest istotny, kolejka subskrypcji nie jest jawnie zamknięta w Przykład 2: Zarządzany subskrybent MQ .

Innym sposobem zrozumienia wzorców aplikacji publikowania/subskrypcji jest zbyt duże spojrzenie na interakcje między różnymi zaangażowanymi podmiotami. Diagramy linii Lifeline lub UML są dobrym sposobem na badanie interakcji. W sekcji “Cykle życia publikowania/subskrybowania” na stronie 311opisano trzy przykłady linii życia.

Przykład 1: konsument publikacji MQ

Konsument publikowania produktu MQ jest konsumentem komunikatów produktu IBM WebSphere MQ , który nie subskrybuje się do samych tematów.

Aby utworzyć kolejkę subskrypcji i publikacji dla tego przykładu, należy uruchomić następujące komendy lub zdefiniować obiekty za pomocą programu WebSphere MQ Explorer.

```
DEFINE QLOCAL(STOCKTICKER) REPLACE;
DEFINE SUB(IBMSTOCKPRICESUB) DEST(STOCKTICKER) TOPICOBJ(IBMSTOCKPRICE) REPLACE;
```

Subskrypcja programu `IBMSTOCKPRICESUB` odwołuje się do obiektu tematu `IBMSTOCK` utworzonego dla przykładu publikatora i kolejki lokalnej `STOCKTICKER`. Obiekt tematu `IBMSTOCK` definiuje łańcuch tematu, który jest używany w subskrypcji, `NYSE/IBM/PRICE`. Należy pamiętać, że przed utworzeniem subskrypcji konieczne jest zdefiniowanie obiektu tematu i kolejki używanej do odbierania publikacji.

Istnieje wiele cennych aspektów dla wzorca konsumenta publikacji produktu MQ :

1. Multiprocessing: dzielenie się z pracą lektur publikacji. Wszystkie publikacje znajdują się w pojedynczej kolejce powiązanej z tematem subskrypcji. Wiele konsumentów może otworzyć kolejkę za pomocą programu `MQOO_INPUT_SHARED`.

2. Centralnie zarządzane subskrypcje. Aplikacje nie konstruują własnych tematów subskrypcji ani subskrypcji. Administrator jest odpowiedzialny za to, gdzie są wysyłane publikacje.
3. Koncentracja subskrypcji: do jednej kolejki może zostać wysłana wiele różnych subskrypcji.
4. Trwałość subskrypcji: kolejka odbiera wszystkie publikacje, bez względu na to, czy konsumenci są aktywni.
5. Migracja i współistnienie: kod konsumenta działa równie dobrze w przypadku scenariusza "punkt z punktem" i "publikowania/subskrypcji".

Subskrypcja tworzy relację między łańcuchem tematu `NYSE/IBM/PRICE` a kolejką `STOCKTICKER`. Publikacje, w tym wszelkie aktualnie zachowane publikacje, są przekazywane do produktu `STOCKTICKER` od momentu utworzenia subskrypcji.

Można zarządzać lub zarządzać niezarządzaną subskrypcją administracyjną utworzoną przez administratora. Subskrypcja zarządzana staje się skuteczna od momentu jej utworzenia, podobnie jak subskrypcja niezarządzana. Nie wszystkie aspekty wzorca są dostępne dla subskrypcji zarządzanej. Więcej informacji znajduje się w sekcji “Przykład 3: Niezarządzany subskrybent MQ” na stronie 302

Uwaga: Kompaktowy styl kodowania jest przeznaczony do czytelności, a nie do użytku produkcyjnego.

Wyniki są wyświetlane w programie Rysunek 43 na stronie 296.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
int main(int argc, char **argv)
{
    MQCHAR      publicationBuffer[101];
    MQCHAR48 subscriptionQueueDefault = "STOCKTICKER";
    MQCHAR48 qmName = "";                       /* Use default queue manager    */

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN;       /* connection handle            */
    MQHOBJ   Hobj = MQHO_NONE;                  /* object handle sub queue      */
    MQLONG   CompCode = MQCC_OK;                /* completion code              */
    MQLONG   Reason = MQRC_NONE;                /* reason code                  */
    MQLONG   messlen = 0;
    MQOD     od = {MQOD_DEFAULT};               /* Unmanaged subscription queue */
    MQMD     md = {MQMD_DEFAULT};               /* Message Descriptor           */
    MQGMO    gmo = {MQGMO_DEFAULT};             /* Get message options          */
    char *   publication=publicationBuffer;
    char *   subscriptionQueue = subscriptionQueueDefault;

    switch(argc){                     /* Replace defaults with args if provided */
        default:
            subscriptionQueue = argv[1]
        case(1):
            printf("Optional parameter: subscriptionQueue\n");
    }

    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        strncpy(od.ObjectName, subscriptionQueue, MQ_Q_NAME_LENGTH);
        MQOPEN(Hconn, &od, MQOO_INPUT_AS_Q_DEF | MQOO_FAIL_IF_QUIESCING , &Hobj, &CompCode,
&Reason);
        if (CompCode != MQCC_OK) break;
        gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
        gmo.WaitInterval = 10000;
        printf("Waiting %d seconds for publications from %s\n", gmo.WaitInterval/1000,
subscriptionQueue);
        do {
            memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
            memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
            md.Encoding       = MQENC_NATIVE;
            md.CodedCharSetId = MQCCSI_Q_MGR;
            memset(publication, 0, sizeof(publicationBuffer));
            MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen,
                  &CompCode, &Reason);
            if (Reason == MQRC_NONE)
                printf("Received publication \"%s\"\n", publication);
        }
        while (CompCode == MQCC_OK);
        if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
```

Rysunek 42. Konsument publikowania produktu MQ .

```
X:\Subscribe1\Debug>Subscribe1
Optional parameter: subscriptionQueue
Waiting 10 seconds for publications from STOCKTICKER
Received publication "129"
Completion code 0 and Return code 0
```

Rysunek 43. Dane wyjściowe z konsumenta publikacji MQ

Istnieje kilka standardowych wskazówek do programowania w języku WebSphere MQ *C* , które mają być świadome:

memset(publication, 0, sizeof(publicationBuffer));
Upewnij się, że na końcu komunikatu ma wartość NULL, aby ułatwić formatowanie przy użyciu produktu `printf`. Przykład publikatora zawiera końcowe wartości null w buforze komunikatów przekazanego do programu MQPUT przez dodanie wartości 1 do `strlen(publication)`. Ustawienie w buforach MQCHAR buforów o wartości NULL jest dobrym stylem programowania dla programów IBM WebSphere MQ w języku C, które używają buforów do przechowywania łańcuchów, przy czym wartość NULL jest zgodna z tablicą znaków, która nie zapełnia całkowicie buforu.

MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen, &CompCode, &Reason);
Rezerwuj jedną wartość null na końcu buforu komunikatów, aby upewnić się, że zwrócony komunikat ma na końcu wartość NULL w przypadku, gdy "`if (messlen == strlen(publication));`" ma wartość true. Ta wskazówka stanowi uzupełnienie poprzedniego elementu i zapewnia, że w produkcie `publicationBuffer` jest co najmniej jedna wartość NULL, która nie jest nadpisywana przez zawartość produktu `publication`.

Pojęcia pokrewne

“Przykład 2: zarządzany subskrybent MQ” na stronie 297
Zarządzany subskrybent produktu MQ jest preferowanym wzorcem dla większości aplikacji subskrybentów. The example requires *nie* administrative definition of queues, topics or subscriptions.

“Przykład 3: Niezarządzany subskrybent MQ” na stronie 302
Niezarządzany subskrybent jest ważną klasą aplikacji subskrybenta. Dzięki temu użytkownik łączy korzyści płynące z publikowania/subskrybowania za pomocą *sterowania* kolejkowania i korzystania z publikacji. W przykładzie przedstawiono różne sposoby łączenia subskrypcji i kolejek.

“Zapisywanie aplikacji publikatorów” na stronie 286
Rozpoczęcie pracy z pisaniem aplikacji publikatorów poprzez zapoznanie się z dwoma przykładami. Pierwsza z nich jest możliwie najdokładniej modelowana w punkcie, w którym aplikacja wskazuje na umieszczanie komunikatów w kolejce, a druga demonstruje dynamicznie tworzenie tematów-bardziej powszechny wzorzec dla aplikacji publikatorów.

Przykład 2: zarządzany subskrybent MQ

Zarządzany subskrybent produktu MQ jest preferowanym wzorcem dla większości aplikacji subskrybentów. The example requires *nie* administrative definition of queues, topics or subscriptions.

This simplest kind of managed subscriber typically uses a *nietrwały* subscription. Ten przykład koncentruje się na nietrwałej subskrypcji. Subskrypcja trwa tylko tak długo, jak czas życia uchwytu subskrypcji z produktu MQSUB. Wszystkie publikacje , które są zgodne z łańcuchem tematu w czasie życia subskrypcji, są wysyłane do kolejki subskrypcji (a być może zachowana publikacja, jeśli opcja `MQSO_NEW_PUBLICATIONS_ONLY` nie jest ustawiona lub zostanie użyta jako domyślna, wcześniejsza publikacja zgodna z łańcuchem tematu została zachowana, a publikacja była trwała lub menedżer kolejek nie zakończył działania, ponieważ została utworzona).

Z tym wzorcem można również użyć *trwałej* subskrypcji. Zwykle , jeśli zarządzana trwała subskrypcja jest używana, jest wykonywana z powodów niezawodności, a nie w celu ustanowienia subskrypcji, która bez żadnych błędów może spowodować przeżycie subskrybentowi. For more information about different lifecycles associated with managed, unmanaged, durable and non-durable subscriptions see the related topics section.

Subskrypcje trwałe są często powiązane z publikacjami trwałymi i subskrypcjami nietrwałymi z publikacjami nietrwałymi, ale nie ma potrzeby relacji między trwałością subskrypcji a trwałością publikacji. Możliwe są wszystkie cztery kombinacje trwałości i trwałości.

W przypadku rozpatrywanych nietrwałych obserwacji zarządzanych menedżer kolejek tworzy kolejkę subskrypcji, która jest czyszczona i usuwana po zamknięciu kolejki. Publikacje są usuwane z kolejki, gdy subskrypcja nietrwała jest zamknięta.

Cennymi aspektami zarządzanego nietrwałego wzorca, które są przykładowo następujące:

1. W przypadku subskrypcji na żądanie produktu : łańcuch tematu subskrypcji jest dynamiczny. Jest ona udostępniana przez aplikację po jej uruchomieniu.

2. Kolejka samozarządzająca: kolejka subskrypcji jest samodefiniująca się i zarządzająca.
3. Cykl życia subskrypcji samozarządzającej: subskrypcje produktu *non-durable* istnieją tylko na czas trwania aplikacji subskrybenta.
   - Jeśli zostanie zdefiniowana *trwała* subskrypcja zarządzana, wówczas będzie ona przechowywana w trwałej kolejce subskrypcji, a publikacje będą przechowywane na niej bez aktywnych programów subskrybentów. Menedżer kolejek usuwa kolejkę (i usuwa z niej wszystkie niepobrane publikacje) tylko po tym, jak aplikacja lub administrator wybrała opcję usunięcia subskrypcji. Subskrypcję można usunąć za pomocą komendy administracyjnej lub zamykając subskrypcję za pomocą opcji `MQCO_REMOVE_SUB` .
   - Rozważ ustawienie `SubExpiry` dla trwałych subskrypcji, dzięki czemu publikacje przestają być wysyłane do kolejki, a subskrybent może korzystać z pozostałych publikacji przed usunięciem subskrypcji, co spowoduje, że menedżer kolejek usunie kolejkę i wszystkie pozostałe publikacje na jej temat.
4. Elastyczne wdrażanie łańcuchów tematów: zarządzanie tematem subskrypcji jest uproszczone przez zdefiniowanie głównej części subskrypcji przy użyciu tematu zdefiniowanego administracyjnie. Część główna drzewa tematów jest następnie ukryta w aplikacji. Przez ukrywanie części głównej aplikacji można wdrożyć bez potrzeby nieumyślnego tworzenia drzewa tematów, które nakłada się na inne drzewo tematów utworzone przez inną instancję lub inną aplikację.
5. Tematy administrowane: przez program przy użyciu łańcucha tematu, w którym pierwsza część jest zgodna z administrowanymi obiektami tematu, publikacje są zarządzane zgodnie z atrybutami obiektu tematu .
   - Jeśli na przykład pierwsza część łańcucha tematu jest zgodna z tematu powiązanym z klastrowym obiektem tematu, wówczas subskrypcja może odbierać publikacje od innych elementów klastra.
   - Selektywne dopasowanie zdefiniowanych administracyjnie obiektów tematu i programowo zdefiniowanych subskrypcji umożliwia połączenie korzyści obu z nich. The administrator provides attributes for topics, and the programmer dynamically defines "sub-topics" without being concerned about the management of topics.
   - to wynikowy łańcuch tematu, który jest używany w celu dopasowania do obiektu tematu, który udostępnia atrybuty powiązane z tematem, a niekoniecznie obiekt tematu o nazwie `sd.Objectname`, chociaż zwykle są one wyświetlane jako jeden i ten sam. Patrz “Przykład 2: publikator w temacie o zmiennej” na stronie 290.

Po zamknięciu subskrypcji przez program publikacje są nadal wysyłane do kolejki subskrypcji po zamknięciu subskrypcji przez subskrybenta za pomocą opcji `MQCO_KEEP_SUB`. Kolejka kontynuuje odbieranie publikacji, gdy subskrybent nie jest aktywny. To zachowanie można przesłonić, tworząc subskrypcję za pomocą opcji `MQSO_PUBLICATIONS_ON_REQUEST` i używając `MQSUBRQ` do żądania zachowanej publikacji.

Subskrypcję można wznowić w późniejszym czasie, otwierając subskrypcję za pomocą opcji `MQCO_RESUME` .

Za pomocą uchwytu kolejki `Hobj`zwróconego przez produkt `MQSUB` na wiele sposobów można użyć uchwytu kolejki. Uchwyt kolejki jest używany w przykładzie do uzyskiwania informacji o nazwie kolejki subskrypcji. Kolejki zarządzane są otwierane za pomocą domyślnych kolejek modelowych `SYSTEM.NDURABLE.MODEL.QUEUE` lub `SYSTEM.DURABLE.MODEL.QUEUE`. Użytkownik może przesłonić wartości domyślne, udostępniając własne trwałe i nietrwałe kolejki modelowe w temacie według właściwości jako właściwości obiektu tematu powiązanego z subskrypcją.

Niezależnie od atrybutów dziedziczonych z kolejek modelowych, nie można ponownie wykorzystać uchwytu kolejki zarządzanej w celu utworzenia dodatkowej subskrypcji. Nie można również uzyskać innego uchwytu dla zarządzanej kolejki, otwierając kolejkę zarządzaną po raz drugi przy użyciu zwróconej nazwy kolejki. Kolejka zachowuje się tak, jakby została otwarta dla wyłącznego wejścia.

Kolejki niezarządzane są bardziej elastyczne niż kolejki zarządzane. Możliwe jest, na przykład współużytkowanie niezarządzanych kolejek, lub zdefiniowanie wielu subskrypcji w jednej kolejce. Następny przykład “Przykład 3: Niezarządzany subskrybent MQ” na stronie 302demonstruje sposób łączenia subskrypcji z niezarządzaną kolejką subskrypcji.

Uwaga: Kompaktowy styl kodowania jest przeznaczony do czytelności, a nie do użytku produkcyjnego.

Wyniki są wyświetlane w programie Rysunek 46 na stronie 300.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>

void inquireQname(MQHCONN HConn, MQHOBJ Hobj, MQCHAR48 qName);

int main(int argc, char **argv)
{
    MQCHAR48 topicNameDefault     = "STOCKS";
    char     topicStringDefault[] = "IBM/PRICE";
    MQCHAR48 qmName = "";                 /* Use default queue manager     */
    MQCHAR48 qName = "";                  /* Allocate to query queue name  */
    char     publicationBuffer[101];      /* Allocate to receive messages  */
    char     resTopicStrBuffer[151];   /* Allocate to resolve topic string */

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle             */
    MQHOBJ   Hobj = MQHO_NONE;            /* publication queue handle      */
    MQHOBJ   Hsub = MQSO_NONE;            /* subscription handle           */
    MQLONG   CompCode = MQCC_OK;          /* completion code               */
    MQLONG   Reason = MQRC_NONE;          /* reason code                   */
    MQLONG   messlen = 0;
    MQSD     sd = {MQSD_DEFAULT};         /* Subscription Descriptor       */
    MQMD     md = {MQMD_DEFAULT};         /* Message Descriptor            */
    MQGMO   gmo = {MQGMO_DEFAULT};        /* get message options           */

    char *   topicName   = topicNameDefault;
    char *   topicString = topicStringDefault;
    char *   publication = publicationBuffer;
    char *   resTopicStr = resTopicStrBuffer;
    memset(resTopicStr, 0, sizeof(resTopicStrBuffer));

    switch(argc){               /* Replace defaults with args if provided */
        default:
          topicString = argv[2];
        case(2):
          if (strcmp(argv[1],"/"))       /* "/" invalid = No topic object  */
             topicName = argv[1];
          else
             *topicName = '\0';
        case(1):
             printf("Optional parameters: topicName, topicString\nValues \"%s\"  \"%s\"\n",
                   topicName, topicString);
    }
```

Rysunek 44. Subskrybent MQ -część 1: deklaracje i obsługa parametrów.

W tym przykładzie są dostępne dodatkowe komentarze dotyczące deklaracji.

MQHOBJ Hobj = MQHO_NONE;
Użytkownik nie może jawnie otworzyć nietrwałej zarządzanej kolejki subskrypcji w celu odbierania publikacji, ale konieczne jest przydzielenie pamięci dla uchwytu obiektu, który jest zwracany przez menedżer kolejek po otwarciu kolejki dla użytkownika. Ważne jest, aby zainicjować uchwyt do produktu MQHO_OBJECT. Ten wskazuje menedżera kolejek, że musi zwrócić uchwyt kolejki do kolejki subskrypcji.

MQSD sd = {MQSD_DEFAULT};
Nowy deskryptor subskrypcji używany w produkcie MQSUB.

MQCHAR48 qName;
Although the example doesn't require knowledge of the subscription queue, the example does inquire the name of the subscription queue - the MQINQ binding is a little awkward in the C language, so you might find this part of the example useful to study.

```
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        strncpy(sd.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        sd.ObjectString.VSPtr = topicString;
        sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
        sd.Options = MQSO_CREATE | MQSO_MANAGED | MQSO_NON_DURABLE | MQSO_FAIL_IF_QUIESCING ;
        sd.ResObjectString.VSPtr = resTopicStr;
        sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
        MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
        gmo.WaitInterval = 10000;
        inquireQname(Hconn, Hobj, qName);
        printf("Waiting %d seconds for publications matching \"%s\" from \"%-0.48s\"\n",
               gmo.WaitInterval/1000, resTopicStr, qName);
        do {
            memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
            memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
            md.Encoding       = MQENC_NATIVE;
            md.CodedCharSetId = MQCCSI_Q_MGR;
            memset(publicationBuffer, 0, sizeof(publicationBuffer));
            MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer-1),
                publication, &messlen, &CompCode, &Reason);
            if (Reason == MQRC_NONE)
                printf("Received publication \"%s\"\n", publication);
        }
        while (CompCode == MQCC_OK);
        if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
        MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
    return;
}
void inquireQname(MQHCONN Hconn, MQHOBJ Hobj, MQCHAR48 qName) {
#define _selectors 1
#define _intAttrs 1

    MQLONG select[_selectors] = {MQCA_Q_NAME}; /* Array of attribute selectors */
    MQLONG intAttrs[_intAttrs];                /* Array of integer attributes  */
    MQLONG CompCode, Reason;
    MQINQ(Hconn, Hobj, _selectors, select, _intAttrs, intAttrs, MQ_Q_NAME_LENGTH, qName,
          &CompCode, &Reason);
    if (CompCode != MQCC_OK) {
        printf("MQINQ failed with Condition code %d and Reason %d\n", CompCode, Reason);
        strcpy(qName, "unknown queue");
    }
    return;
}
```

Rysunek 45. Zarządzany subskrybent MQ -część 2: treść kodu.

```
W:\Subscribe2\Debug>solution2
Optional parameters: topicName, topicString
Values "STOCKS"  "IBM/PRICE"
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from
"SYSTEM.MANAGED.NDURABLE.48A0AC7403300020        "
Received publication "150"
Completion code 0 and Return code 0

W:\Subscribe2\Debug>solution2 / NYSE/IBM/PRICE
Optional parameters: topicName, topicString
Values ""  "NYSE/IBM/PRICE"
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from
"SYSTEM.MANAGED.NDURABLE.48A0AC7403310020        "
Received publication "150"
Completion code 0 and Return code 0
```

Rysunek 46. Dane wyjściowe z zarządzanego subskrybenta produktu MQ

W tym przykładzie są dostępne dodatkowe komentarze dotyczące kodu.

strncpy(sd.ObjectName, topicName, MQ_Q_NAME_LENGTH);
Jeśli parametr topicName ma wartość NULL lub jest pusta *(wartość domyślna)*, nazwa tematu nie jest używana do obliczenia przetłumaczanego łańcucha tematu.

sd.ObjectString.VSPtr = topicString;
Zamiast używać wyłącznie predefiniowanego obiektu tematu, w tym przykładzie programista udostępnia obiekt tematu i łańcuch tematu, które są łączone przez produkt MQSUB. Należy zauważyć, że łańcuch tematu jest strukturą MQCHARV .

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Alternatywa dla ustawienia długości pola MQCHARV .

sd.Options = MQSO_CREATE | MQSO_MANAGED | MQSO_NON_DURABLE | MQSO_FAIL_IF_QUIESCING;
Po zdefiniowaniu łańcucha tematu flagi sd.Options wymagają najbardziej uważnej uwagi. Istnieje wiele opcji, w przykładzie określa się tylko najczęściej używane; pozostałe są pozostawiane jako domyślne .

1. Ponieważ subskrypcja jest *nietrwała*, oznacza to, że ma ona czas życia otwartej subskrypcji w aplikacji, należy ustawić flagę MQSO_CREATE. Można również ustawić flagę *(wartość domyślna)* MQSO_NON_DURABLE w celu uzyskania czytelności.
2. Uzupełnieniem produktu MQSO_CREATE jest MQSO_RESUME. Both flags can be set together; the queue manager either creates a new subscription or resumes an existing subscription, whichever is appropriate. Jeśli jednak zostanie określona opcja MQSO_RESUME , należy również zainicjować strukturę MQCHARV dla produktu sd.SubName, nawet jeśli nie ma subskrypcji, która ma zostać wznowiona. Niepowodzenie inicjowania SubName powoduje zwrócenie kodu powrotu 2440: MQRC_SUB_NAME_ERROR z MQSUB.

   Uwaga: Produkt MQSO_RESUME jest zawsze ignorowany w przypadku nietrwałej subskrypcji zarządzanej: ale określanie jej bez inicjowania struktury MQCHARV dla sd.SubName powoduje błąd.
3. Dodatkowo istnieje trzecia flaga wpływający na sposób otwierania subskrypcji, MQSO_ALTER. Po nadaniu odpowiednich uprawnień właściwości wznawianej subskrypcji zostały zmienione tak, aby były zgodne z innymi atrybutami określonymi w MQSUB.

   Uwaga: Należy określić co najmniej jedną z opcji MQSO_CREATE, MQSO_RESUME i MQSO_ALTER . Patrz Opcje (MQLONG). Istnieją przykłady użycia wszystkich trzech flag w produkcie “Przykład 3: Niezarządzany subskrybent MQ” na stronie 302.
4. Ustaw opcję MQSO_MANAGED dla menedżera kolejek, aby automatycznie zarządzać subskrypcją.

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Opcjonalnie można pominąć ustawienie długości łańcucha MQCHARV dla łańcuchów zakończonych znakiem NULL i zamiast tego użyć flagi znaku końca wartości NULL.

sd.ResObjectString.VSPtr = resTopicStr;
Wynikowy łańcuch tematu jest umieszczany w programie printf w pierwszym miejscu programu. Skonfiguruj produkt MQCHARV ResObjectString dla produktu WebSphere MQ , aby przywrócić przetłumaczany łańcuch z powrotem do programu.

Uwaga: Produkt resTopicStringBuffer jest inicjowany z wartościami pustymi w produkcie memset(resTopicStr, 0, sizeof(resTopicStrBuffer)). Zwrócone łańcuchy tematów nie kończą się znakiem null kończącym.

sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
Ustaw wielkość buforu sd.ResObjectString na jeden mniejszy niż jego wielkość rzeczywista. This prevents overwriting the null terminator that is provided, in case the resolved topic string fills the entire buffer.

Uwaga: Jeśli łańcuch tematu jest dłuższy niż sizeof(resTopicStrBuffer)-1, nie jest zwracany żaden błąd. Nawet jeśli VSLength > VSBufSiz długość zwrócona w polu sd.ResObjectString.VSLength jest długością kompletnego łańcucha, a niekoniecznie długością zwróconego łańcucha. Przetestuj sd.ResObjectString.VSLength < sd.ResObjectString.VSBufSiz , aby potwierdzić, że łańcuch tematu został zakończony.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
Funkcja MQSUB tworzy subskrypcję. Jeśli nie jest to trwałe, prawdopodobnie nie interesuje Cię jego nazwa, ale można sprawdzić jego status w programie WebSphere MQ Explorer. You can provide the sd.SubName parameter as input, so you know what name to look for; you obviously have to avoid name clashes with other subscriptions.

MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
Zamknięcie zarówno subskrypcji, jak i kolejki subskrypcji jest opcjonalne. W tym przykładzie subskrypcja jest zamknięta, ale nie jest kolejką. Opcja MQCLOSE MQCO_REMOVE_SUB jest domyślna w tym przypadku, ponieważ subskrypcja nie jest trwała. Użycie produktu MQCO_KEEP_SUB jest błędem.

Uwaga: Subskrypcja *kolejka* nie jest zamknięta przez produkt MQSUB, a jej uchwyt Hobjpozostaje poprawny do momentu zamknięcia kolejki przez produkt MQCLOSE lub MQDISC. Jeśli aplikacja kończy się przedwcześnie, kolejka i subskrypcja są czyszczone przez menedżer kolejek w czasie po zakończeniu aplikacji.

Pojęcia pokrewne

“Przykład 1: konsument publikacji MQ” na stronie 294
Konsument publikowania produktu MQ jest konsumentem komunikatów produktu IBM WebSphere MQ , który nie subskrybuje się do samych tematów.

“Przykład 3: Niezarządzany subskrybent MQ” na stronie 302
Niezarządzany subskrybent jest ważną klasą aplikacji subskrybenta. Dzięki temu użytkownik łączy korzyści płynące z publikowania/subskrybowania za pomocą *sterowania* kolejkowania i korzystania z publikacji. W przykładzie przedstawiono różne sposoby łączenia subskrypcji i kolejek.

“Zapisywanie aplikacji publikatorów” na stronie 286
Rozpoczęcie pracy z pisaniem aplikacji publikatorów poprzez zapoznanie się z dwoma przykładami. Pierwsza z nich jest możliwie najdokładniej modelowana w punkcie, w którym aplikacja wskazuje na umieszczanie komunikatów w kolejce, a druga demonstruje dynamicznie tworzenie tematów-bardziej powszechny wzorzec dla aplikacji publikatorów.

Przykład 3: Niezarządzany subskrybent MQ

Niezarządzany subskrybent jest ważną klasą aplikacji subskrybenta. Dzięki temu użytkownik łączy korzyści płynące z publikowania/subskrybowania za pomocą *sterowania* kolejkowania i korzystania z publikacji. W przykładzie przedstawiono różne sposoby łączenia subskrypcji i kolejek.

Wzorzec niezarządzany jest zwykle powiązany z *trwałymi* subskrypcjami niż *nietrwały*. Zwykle cykl życia subskrypcji utworzonej przez niezarządzanego subskrybenta jest niezależny od cyklu życia samej aplikacji subskrybującej. Subskrypcja subskrypcji odbiera publikacje nawet wtedy, gdy żadna aplikacja subskrybująca nie jest aktywna.

Można utworzyć trwałe *zarządzane* subskrypcje w celu osiągnięcia tego samego wyniku, ale niektóre aplikacje wymagają większej elastyczności i kontroli nad kolejkami i komunikatami niż jest to możliwe w przypadku subskrypcji zarządzanej. W przypadku trwałej subskrypcji zarządzanej menedżer kolejek tworzy stałą kolejkę dla publikacji, które są zgodne z tematem subskrypcji. Powoduje ona usunięcie kolejki i powiązanych z nią publikacji w momencie usunięcia subskrypcji.

Zazwyczaj trwałe *zarządzane* subskrypcje są używane, jeśli cykl życia aplikacji i subskrypcja jest zasadniczo taka sama, ale trudno jest zagwarantować. Dzięki temu, że subskrypcja jest trwała, a wydawca tworzy trwałe publikacje, nie ma żadnych utraconych komunikatów, jeśli menedżer kolejek lub subskrybent zostaną przedwcześnie zakończone i muszą zostać odzyskane.

Menedżer kolejek niejawnie otwiera kolejkę subskrypcji trwałej dla subskrybenta w taki sposób, że współużytkowane przetwarzanie kolejki nie jest możliwe. Ponadto nie można utworzyć więcej niż jednej subskrypcji dla każdej zarządzanej kolejki, a kolejki są trudniejsze do zarządzania, ponieważ użytkownik ma mniejszą kontrolę nad nazwami kolejek. Z tych powodów należy rozważyć, czy *niezarządzany* subskrybent produktu MQ jest lepiej dopasowany do aplikacji wymagających trwałych subskrypcji niż subskrybent *zarządzany* MQ .

Kod w produkcie Rysunek 49 na stronie 308 demonstruje niezarządzany wzorzec trwałej subskrypcji. W przypadku ilustracji kod tworzy również subskrypcje niezarządzane, nietrwałe. W tym przykładzie przedstawiono następujące aspekty wzorca:

- Subskrypcje na żądanie: łańcuchy tematów subskrypcji są dynamiczne. Są one udostępniane przez aplikację po jej uruchomieniu.
- Uproszczone zarządzanie tematem subskrypcji: zarządzanie tematem subskrypcji jest uproszczone przez zdefiniowanie głównej części łańcucha tematu subskrypcji przy użyciu tematu zdefiniowanego administracyjnie. Powoduje to ukrycie głównej części drzewa tematów z aplikacji. Ukrywanie części głównej subskrybenta może być wdrażane w różnych drzewach tematów.
- Elastyczne zarządzanie subskrypcją: można zdefiniować subskrypcję administracyjnie lub utworzyć ją na żądanie w ramach programu subskrybenta. Nie ma różnicy między administracyjnym i programowo utworzonym subskrypcją, z wyjątkiem atrybutu, który pokazuje, w jaki sposób subskrypcja została utworzona. Istnieje trzeci typ subskrypcji, który jest tworzony automatycznie przez menedżer kolejek w celu dystrybucji subskrypcji. Wszystkie subskrypcje są wyświetlane w programie WebSphere MQ Explorer.
- Elastyczne powiązanie subskrypcji z kolejkami: predefiniowana kolejka lokalna jest powiązana z subskrypcją za pomocą funkcji `MQSUB` . Istnieją różne sposoby korzystania z programu MQSUB w celu powiązania subskrypcji z kolejkami:
  - Powiązanie subskrypcji z kolejką o *nie* istniejących subskrypcjach `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Powiąż *nową* subskrypcję z kolejką, która ma istniejące subskrypcje, `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Przenieś istniejącą subskrypcję do innej kolejki, `MQSO_ALTER + (Hobj from MQOPEN)`.
  - Wznów istniejącą subskrypcję powiązaną z istniejącą kolejką, `MQSO_RESUME + (Hobj = MQHO_NONE)` lub `MQSO_RESUME + (Hobj = from MQOPEN of queue with existing subscription)` .
  - Łącząc produkt `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` w różnych kombinacjach, można używać różnych stanów wejściowych subskrypcji i kolejki bez konieczności tworzenia kodu wielu wersji produktu `MQSUB` z różnymi wartościami `sd.Options` .
  - Alternatywnie, poprzez zakodowanie konkretnego wyboru `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` , menedżer kolejek zwraca błąd (Tabela 43 na stronie 305), jeśli stany subskrypcji i kolejki podane jako dane wejściowe dla `MQSUB` są niespójne z wartością `sd.Options`. Rysunek 55 na stronie 311 przedstawia wyniki wydania `MQSUB` dla subskrypcji X z różnymi ustawieniami poszczególnych opcji `sd.Options` i przekazywanie jej trzech różnych uchwytów obiektów.

  Zapoznaj się z różnymi danymi wejściowymi do przykładowego programu w programie Rysunek 48 na stronie 307 , aby zapoznać się z różnymi rodzajami błędów. Jeden wspólny błąd, `RC = 2440`, który nie jest zawarty w przypadkach wymienionych w tabeli, jest błędem nazwy subskrypcji. Jest to zwykle spowodowane przez przekazanie nazwy subskrypcji o wartości NULL lub niepoprawnej za pomocą produktu `MQSO_RESUME` lub `MQSO_ALTER` .
- Multiprocessing: Możesz podzielić się wśród wielu konsumentów pracą lektur publikacji. Wszystkie publikacje znajdują się w pojedynczej kolejce powiązanej z tematem subskrypcji. Konsumenci mają możliwość bezpośredniego otwierania kolejki przy użyciu produktu `MQOPEN` lub wznawiania subskrypcji przy użyciu produktu `MQSUB`.
- Koncentracja subskrypcji: w tej samej kolejce może zostać utworzonych wiele subskrypcji. Należy zachować ostrożność przy użyciu tej możliwości, ponieważ może ona prowadzić do "nakładania się" subskrypcji i wielokrotnego otrzymywania tej samej publikacji. Opcja `MQSO_GROUP_SUB` eliminuje zduplikowane publikacje spowodowane nakładającymi się subskrypcjami.
- Abonent i separacja konsumentów: Tak jak w przypadku trzech modeli konsumenckich przedstawionych na przykładach, inny model polega na oddzieleniu konsumenta od abonenta. Jest to odmiana niezarządzanego subskrybenta MQ , ale zamiast wydawania produktów `MQOPEN` i `MQSUB` w tym samym programie, jeden program subskrybuje publikacje, a inny program zużywa je. Subskrybent może na przykład należeć do klastra publikowania/subskrybowania, a konsument przyłączony do menedżera

kolejek poza klastrem menedżera kolejek. Konsument otrzymuje publikacje za pośrednictwem standardowego rozproszonego kolejkowania, definiując kolejkę subskrypcji jako definicję kolejki zdalnej.

Zrozumienie zachowania produktu MQSO_CREATE | MQSO_RESUME | MQSO_ALTER jest ważne, zwłaszcza jeśli planowane jest uproszczenie kodu za pomocą kombinacji tych opcji. Należy zbadać tabelę Tabela 43 na stronie 305 , która przedstawia wyniki przekazywania różnych uchwytów kolejek do tabeli MQSUB, a także wyniki działania przykładowego programu przedstawionego w sekcji Rysunek 50 na stronie 309 na serwerze Rysunek 55 na stronie 311.

Scenariusz użyty do skonstruowania tabeli ma jedną subskrypcję X i dwie kolejki, A i B . Parametr nazwy subskrypcji sd.SubName jest ustawiony na wartość X(nazwa subskrypcji przyłączonej do kolejki A). W kolejce B nie jest przyłączona żadna subskrypcja.

W produkcie Tabela 43 na stronie 305 MQSUB jest przekazywana subskrypcja X i uchwyt kolejki do kolejki A. Wyniki z opcji subskrypcji są następujące:

- MQSO_CREATE nie powiodło się, ponieważ uchwyt kolejki odpowiada kolejce A , która ma już subskrypcję X. To zachowanie jest przeciwstawne do pomyślnego wywołania. Wywołanie to powiodło się, ponieważ kolejka B nie ma subskrypcji do X dołączonej do niej.
- Pomyślnie MQSO_RESUME , ponieważ uchwyt kolejki odpowiada kolejce A , która ma już subskrypcję X. W przeciwieństwie do tego wywołanie nie powiedzie się, jeśli subskrypcja X nie istnieje w kolejce A.
- Produkt MQSO_ALTER działa w podobny sposób, jak produkt MQSO_RESUME w celu otwarcia subskrypcji i kolejki. Jeśli jednak atrybuty zawarte w deskryptorze subskrypcji przekazanej do produktu MQSUB różnią się od atrybutów subskrypcji, działanie produktu MQSO_RESUME nie powiedzie się, natomiast operacja MQSO_ALTER zakończy się tak długo, jak długo instancja programu ma uprawnienia do zmiany atrybutów. Należy pamiętać, że nigdy nie można zmienić łańcucha tematu w subskrypcji. Zamiast zwrócić błąd, program MQSUB ignoruje nazwę tematu i wartości łańcucha tematu w deskryptorze subskrypcji i używa wartości w istniejącej subskrypcji.

Następnie należy sprawdzić Tabela 43 na stronie 305 , gdzie zmaterializowana tabela zapytania (MQSUB) jest przekazywana do subskrypcji X, a uchwyt kolejki do kolejki B. Wyniki z opcji subskrypcji są następujące:

- MQSO_CREATE udaje się i tworzy subskrypcję X w kolejce B , ponieważ jest to nowa subskrypcja w kolejce B.
- MQSO_RESUME nie powiodło się. Produkt MQSUB wyszukuje subskrypcję X w kolejce B i nie znajduje jej, ale nie zwraca wartości *RC = 2428-subskrypcja X nie istnieje* , zwraca wartość *RC = 2019-Kolejka subskrypcji nie jest zgodna z uchwytem obiektu kolejki*. Zachowanie trzeciej opcji MQSO_ALTER sugeruje przyczynę tego nieoczekiwanego błędu. Produkt MQSUB oczekuje, że uchwyt kolejki będzie wskazywał kolejkę z subskrypcją. Sprawdza to najpierw przed sprawdzeniem, czy subskrypcja o nazwie sd.SubName istnieje.
- MQSO_ALTER zakończy się powodzeniem, a następnie przenosi subskrypcję z kolejki A do kolejki B.

Przypadek, który nie jest wyświetlany w tabeli, jest taki, że jeśli nazwa subskrypcji subskrypcji w kolejce A nie jest zgodna z nazwą subskrypcji w produkcie sd.SubName. Wywołanie to kończy się niepowodzeniem z kodem powrotu *RC = 2428-subskrypcja X nie istnieje w kolejce A* .

Tabela 43. Błędy z tabeli MQSUB z różnymi uchwytami kolejek i kombinacjami subskrypcji		
	Kolejka A **Subskrypcja X** **Kolejka B** **Brak subskrypcji**	**Kolejka A** **Brak subskrypcji** **Kolejka B** **Brak subskrypcji**
Hobj dla kolejki *Kolejka A* przekazanej do tabeli MQSUB	**MQSO_CREATE** RC = 2432-Subskrypcja X już istnieje w kolejce A **MQSO_RESUME** Wznawia subskrypcję X w kolejce A **MQSO_ALTER** Wznawia subskrypcję X w kolejce A i wprowadza dozwolone zmiany	**MQSO_CREATE** Tworzy subskrypcję X w kolejce A **MQSO_RESUME** RC = 2428-Subskrypcja X nie istnieje w kolejce A **MQSO_ALTER** RC = 2428-Subskrypcja X nie istnieje w kolejce A
Hobj dla kolejki *Kolejka B* przekazany do tabeli MQSUB	**MQSO_CREATE** Tworzy nową subskrypcję X w kolejce B **MQSO_RESUME** RC = 2019-Kolejka subskrypcji nie jest zgodna z uchwytem obiektu kolejki **MQSO_ALTER** Przenieś subskrypcję X z kolejki A do kolejki B	**MQSO_CREATE** Tworzy nową subskrypcję X w kolejce B **MQSO_RESUME** RC = 2428-subskrypcja X nie istnieje w kolejce B **MQSO_ALTER** RC = 2428-subskrypcja X nie istnieje w kolejce B
MQHO_NONE przekazano do tabeli MQSUB	**MQSO_CREATE** RC = 2019-Bad object handle: set MQSO_MANAGED flag to create a managed subscription and create a managed queue **MQSO_RESUME** Wznawia subskrypcję X w kolejce A i zwraca Hobj do kolejki A **MQSO_ALTER** Wznawia subskrypcję X w kolejce A, zwraca Hobj do kolejki A i wprowadza dozwolone zmiany	**MQSO_CREATE** RC = 2019-Bad object handle: set MQSO_MANAGED flag to create a managed subscription and create a managed queue **MQSO_RESUME** RC = 2428-Brak subskrypcji X **MQSO_ALTER** RC = 2019-Błędny uchwyt obiektu: brak kolejki A lub B

Uwaga: Kompaktowy styl kodowania jest przeznaczony do czytelności, a nie do użytku produkcyjnego.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>

void inquireQname(MQHCONN HConn, MQHOBJ Hobj, MQCHAR48 qName);

int main(int argc, char **argv)
{
    MQCHAR48 topicNameDefault            = "STOCKS";
    char     topicStringDefault[]        = "IBM/PRICE";
    char     subscriptionNameDefault[]   = "IBMSTOCKPRICESUB";
    char     subscriptionQueueDefault[]  = "STOCKTICKER";
    char     publicationBuffer[101];      /* Allocate to receive messages     */
    char     resTopicStrBuffer[151];      /* Allocate to resolve topic string */
    MQCHAR48 qmName = "";                 /* Default queue manager            */
    MQCHAR48 qName = "";                  /* Allocate storage for MQINQ       */

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle                */
    MQHOBJ   Hobj = MQHO_NONE;            /* subscription queue handle        */
    MQHOBJ   Hsub = MQSO_NONE;            /* subscription handle              */
    MQLONG   CompCode = MQCC_OK;          /* completion code                  */
    MQLONG   Reason = MQRC_NONE;          /* reason code                      */
    MQLONG   messlen = 0;
    MQOD     od = {MQOD_DEFAULT};         /* Unmanaged subscription queue     */
    MQSD     sd = {MQSD_DEFAULT};         /* Subscription Descriptor          */
    MQMD     md = {MQMD_DEFAULT};         /* Message Descriptor               */
    MQGMO   gmo = {MQGMO_DEFAULT};        /* get message options              */
    MQLONG   sdOptions = MQSO_CREATE | MQSO_RESUME | MQSO_DURABLE |
MQSO_FAIL_IF_QUIESCING;

    char *   topicName = topicNameDefault;
    char *   topicString = topicStringDefault;
    char *   subscriptionName =  subscriptionNameDefault;
    char *   subscriptionQueue = subscriptionQueueDefault;
    char *   publication = publicationBuffer;
    char *   resTopicStr = resTopicStrBuffer;
    memset(resTopicStrBuffer, 0, sizeof(resTopicStrBuffer));
```

Rysunek 47. Niezarządzany subskrybent MQ -część 1: deklaracje.

```
    switch(argc){                   /* Replace defaults with args if provided */
default:
        switch((argv[5][0])) {
case('A'):  sdOptions =  MQSO_ALTER  | MQSO_DURABLE | MQSO_FAIL_IF_QUIESCING;
            break;
case('C'):  sdOptions =  MQSO_CREATE | MQSO_DURABLE | MQSO_FAIL_IF_QUIESCING;
            break;
case('R'):  sdOptions =  MQSO_RESUME | MQSO_DURABLE | MQSO_FAIL_IF_QUIESCING;
            break;
default:    ;
        }
case(5):
    if (strcmp(argv[4],"/")) /* "/" invalid = No subscription  */
        subscriptionQueue = argv[4];
    else {
        *subscriptionQueue = '\0';
        if (argc > 5) {
            if (argv[5][0] == 'C') {
                sdOptions = sdOptions + MQSO_MANAGED;
            }
        }
        else
            sdOptions = sdOptions + MQSO_MANAGED;
    }

case(4):
    if (strcmp(argv[3],"/")) /* "/" invalid = No subscription  */
        subscriptionName = argv[3];
    else {
        *subscriptionName = '\0';
        sdOptions = sdOptions - MQSO_DURABLE;
    }
case(3):
    if (strcmp(argv[2],"/")) /* "/" invalid = No topic string  */
        topicString = argv[2];
    else
        *topicString = '\0';
case(2):
    if (strcmp(argv[1],"/")) /* "/" invalid = No topic object  */
        topicName = argv[1];
    else
        *topicName = '\0';
case(1):
    sd.Options = sdOptions;
    printf("Optional parameters: "
    printf("topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|
R(esume)\n");
    printf("Values \"%-.48s\"  \"%s\" \"%s\"  \"%-.48s\" sd.Options=%d\n",
           topicName, topicString, subscriptionName, subscriptionQueue, sd.Options);
    }
```

Rysunek 48. Niezarządzany subskrybent MQ -część 2: obsługa parametrów.

Dodatkowe komentarze dotyczące obsługi parametrów w tym przykładzie są następujące:

switch((argv[5][0]))
Istnieje możliwość wprowadzenia parametru Alter | C reate | Resume w parametrze 5, aby przetestować efekt przesłonięcie części ustawienia opcji MQSUB , która jest używana domyślnie w tym przykładzie. Domyślnym ustawieniem używanym w tym przykładzie jest MQSO_CREATE | MQSO_RESUME | MQSO_DURABLE.

Uwaga: Ustawienie MQSO_ALTER lub MQSO_RESUME bez ustawienia MQSO_DURABLE jest błędem, a sd.SubName musi być ustawione i odwoływać się do subskrypcji, która może zostać wznowiona lub zmieniona.

***subscriptionQueue = '\0';**
sdOptions = sdOptions + MQSO_MANAGED;
Jeśli domyślna kolejka subskrypcji STOCKTICKER jest zastępowana łańcuchem pustym, o ile ustawiona jest wartość MQSO_CREATE , w przykładzie ustawiana jest flaga MQSO_MANAGED i tworzona jest kolejka subskrypcji dynamicznej. Jeśli parametr Alter or Resume jest ustawiony w piątym parametrze, zachowanie tego przykładu będzie zależeć od wartości subscriptionName.

***subscriptionName = '\0';**
sdOptions = sdOptions - MQSO_DURABLE;

Jeśli domyślna subskrypcja, IBMSTOCKPRICESUB , zostanie zastąpiona łańcuchem pustym, na przykład zostanie usunięta flaga MQSO_DURABLE . Jeśli zostanie uruchomiony przykład udostępniający wartości domyślne dla innych parametrów, zostanie utworzona dodatkowa subskrypcja tymczasowa przeznaczona dla produktu STOCKTICKER , a następnie zostaną odebrane zduplikowane publikacje. Następnym razem, gdy uruchomisz przykład, bez żadnych parametrów, otrzymasz po prostu jedną publikację.

```
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        if (strlen(subscriptionQueue)) {
            strncpy(od.ObjectName, subscriptionQueue, MQ_Q_NAME_LENGTH);
            MQOPEN(Hconn, &od, MQOO_INPUT_AS_Q_DEF | MQOO_FAIL_IF_QUIESCING | MQOO_INQUIRE,
                  &Hobj, &CompCode, &Reason);
            if (CompCode != MQCC_OK) break;
        }
        strncpy(sd.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        sd.ObjectString.VSPtr = topicString;
        sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
        sd.SubName.VSPtr = subscriptionName;
        sd.SubName.VSLength = MQVS_NULL_TERMINATED;
        sd.ResObjectString.VSPtr = resTopicStr;
        sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
        MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
        gmo.WaitInterval = 10000;
        gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
        memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
        inquireQname(Hconn, Hobj, qName);
        printf("Waiting %d seconds for publications matching \"%s\" from %-0.48s\n",
                gmo.WaitInterval/1000, resTopicStr, qName);
        do {
            memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
            memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
            md.Encoding       = MQENC_NATIVE;
            md.CodedCharSetId = MQCCSI_Q_MGR;
            MQGET(Hconn, Hobj, &md, &gmo, sizeof(publication), publication, &messlen,
&CompCode, &Reason);
            if (Reason == MQRC_NONE)
                printf("Received publication \"%s\"\n", publication);
        }
        while (CompCode == MQCC_OK);
        if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
        MQCLOSE(Hconn, &Hsub, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
void inquireQname(MQHCONN Hconn, MQHOBJ Hobj, MQCHAR48 qName) {
#define _selectors 1
#define _intAttrs 1

    MQLONG select[_selectors] = {MQCA_Q_NAME}; /* Array of attribute selectors    */
    MQLONG intAttrs[_intAttrs];            /* Array of integer attributes           */
    MQLONG CompCode, Reason;
    MQINQ(Hconn, Hobj, _selectors, select, _intAttrs, intAttrs, MQ_Q_NAME_LENGTH, qName,
&CompCode, &Reason);
    if (CompCode != MQCC_OK) {
        printf("MQINQ failed with Condition code %d and Reason %d\n", CompCode, Reason);
        strncpy(qName, "unknown queue", MQ_Q_NAME_LENGTH);
    }
    return;
}
```

Rysunek 49. Niezarządzany subskrybent MQ -część 3: treść kodu.

Dodatkowe komentarze na temat kodu w tym przykładzie są następujące:

if (strlen(subscriptionQueue))
Jeśli nie istnieje nazwa kolejki subskrypcji, wówczas w przykładzie użyto wartości MQHO_NONE jako wartości Hobj.

MQOPEN(...);
Kolejka subskrypcji zostanie otwarta, a uchwyt kolejki został zapisany w produkcie Hobj.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
Subskrypcja jest otwierana za pomocą Hobj przekazanego z MQOPEN (lub MQHO_NONE , jeśli nie ma nazwy kolejki subskrypcji). Kolejka niezarządzana może zostać wznowiona bez jawnego otwierania go za pomocą MQOPEN.

MQCLOSE(Hconn, &Hsub, MQCO_NONE, &CompCode, &Reason);
Subskrypcja jest zamknięta za pomocą uchwytu subskrypcji. W zależności od tego, czy subskrypcja jest trwała, czy nie, subskrypcja jest zamykana z niejawnym MQCO_KEEP_SUB lub MQCO_REMOVE_SUB. Istnieje możliwość zamknięcia trwałej subskrypcji za pomocą produktu MQCO_REMOVE_SUB, ale *nie można* zamknąć nietrwałej subskrypcji za pomocą produktu MQCO_KEEP_SUB. Działanie produktu MQCO_REMOVE_SUB polega na usunięciu subskrypcji, w której zatrzymywane są dalsze publikacje wysyłane do kolejki subskrypcji.

MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
Nie jest podejmowane żadne specjalne działanie, jeśli subskrypcja jest niezarządzana. Jeśli kolejka jest zarządzana, a subskrypcja została zamknięta z jawnym lub niejawnym MQCO_REMOVE_SUB, to wszystkie publikacje są usuwane z kolejki i kolejki usunięte w tym momencie.

gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
Upewnij się, że otrzymane komunikaty są tymi, które są przeznaczone dla subskrypcji.

Wyniki przykładu ilustrują aspekty publikowania/subskrypcji:

W produkcie Rysunek 50 na stronie 309 przykład rozpoczyna się publikowaniem 130 w temacie NYSE/IBM/PRICE .

```
W:\Subscribe3\Debug>..\..\Publish2\Debug\publishstock
Provide parameters:  TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0
```

Rysunek 50. Opublikuj od 130 do NYSE/IBM/PRICE

W programie Rysunek 51 na stronie 309 wykonanie przykładu z użyciem parametrów domyślnych odbiera zachowaną publikację 130. Podany obiekt tematu i łańcuch tematu są ignorowane, jak pokazano na rysunku Rysunek 55 na stronie 311. Obiekt tematu i łańcuch tematu są zawsze pobierane z obiektu subskrypcji, gdy jest on udostępniany, a łańcuch tematu jest niezmienny. Rzeczywiste zachowanie przykładu zależy od wyboru lub kombinacji produktów MQSO_CREATE, MQSO_RESUMEi MQSO_ALTER . W tym przykładzie MQSO_RESUME jest wybraną opcją.

```
W:\Subscribe3\Debug>solution3
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8206
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0
```

Rysunek 51. Odbierz zachowaną publikację

W produkcie (Rysunek 52 na stronie 310) nie są odbierane żadne publikacje, ponieważ trwała subskrypcja otrzymała już zachowaną publikację. W tym przykładzie subskrypcja jest wznawiana przez podanie tylko nazwy subskrypcji bez nazwy kolejki. Jeśli podana nazwa kolejki została podana, kolejka zostanie otwarta jako pierwsza, a uchwyt przekazany do programu MQSUB.

Uwaga: The 2038 error from MQINQ is due to the implicit MQOPEN of STOCKTICKER by MQSUB not including the MQOO_INQUIRE option. Należy unikać kodu powrotu produktu 2038 z programu MQINQ , otwierając jawnie kolejkę.

```
W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE IBMSTOCKPRICESUB / Resume
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" "IBMSTOCKPRICESUB"  "" sd.Options=8204
MQINQ failed with Condition code 2 and Reason 2038
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from unknown queue
Completion code 0 and Return code 0
```

Rysunek 52. Wznów subskrypcję

W produkcie Rysunek 53 na stronie 310 przykład tworzy nietrwałą subskrypcję niezarządzaną za pomocą programu STOCKTICKER jako miejsca docelowego. Ponieważ jest to nowa subskrypcja, otrzymuje ona zachowaną publikację.

```
W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE / STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" ""  "STOCKTICKER" sd.Options=8194
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0
```

Rysunek 53. Otrzymuj zachowaną publikację z nową, niezarządzaną subskrypcją nietrwałą

W programie Rysunek 54 na stronie 310 , aby zademonstrować nakładające się subskrypcje, wysyłana jest inna publikacja, która zmienia zachowaną publikację. Następnie tworzona jest nowa, nietrwała, niezarządzana subskrypcja, która nie udostępnia nazwy subskrypcji. Zachowana publikacja jest otrzymywana dwa razy, raz dla nowej subskrypcji, a raz dla trwałej subskrypcji produktu IBMSTOCKPRICESUB , która jest nadal aktywna w kolejce produktu STOCKTICKER . Przykładem jest ilustracja, że jest to kolejka, która ma subskrypcje, a nie aplikacja. Mimo że nie nawiązano do subskrypcji produktu IBMSTOCKPRICESUB w tym wywołaniu aplikacji, aplikacja otrzymuje publikację dwa razy: jeden raz od trwałej subskrypcji, która została utworzona administracyjnie, a raz z nietrwałej subskrypcji utworzonej przez samą aplikację.

```
W:\Subscribe3\Debug>..\..\Publish2\Debug\publishstock
Provide parameters:  TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0

W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE / STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" ""  "STOCKTICKER" sd.Options=8194
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Received publication "130"
Completion code 0 and Return code 0
```

Rysunek 54. Nakładające się subskrypcje

W produkcie Rysunek 55 na stronie 311 przykład demonstruje, że udostępnienie nowego łańcucha tematu i istniejącej subskrypcji nie powoduje zmiany subskrypcji.

1. W pierwszym przypadku program Resume wznawia istniejącą subskrypcję, czego można się spodziewać, a także ignoruje zmieniony łańcuch tematu.
2. W drugim przypadku produkt Alter powoduje wystąpienie błędu RC = 2510, Topic not alterable.

3. W trzecim przykładzie program `Create` powoduje wystąpienie błędu `RC = 2432, Sub already exists`.

```
W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Resume
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values ""  "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8204
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0

W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Alter
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values ""  "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8201
Completion code 2 and Return code 2510

W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values ""  "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8202
Completion code 2 and Return code 2432
```

Rysunek 55. Tematy subskrypcji nie mogą być zmieniane

Pojęcia pokrewne

“Przykład 1: konsument publikacji MQ” na stronie 294
Konsument publikowania produktu MQ jest konsumentem komunikatów produktu IBM WebSphere MQ , który nie subskrybuje się do samych tematów.

“Przykład 2: zarządzany subskrybent MQ” na stronie 297
Zarządzany subskrybent produktu MQ jest preferowanym wzorcem dla większości aplikacji subskrybentów. The example requires *nie* administrative definition of queues, topics or subscriptions.

“Zapisywanie aplikacji publikatorów” na stronie 286
Rozpoczęcie pracy z pisaniem aplikacji publikatorów poprzez zapoznanie się z dwoma przykładami. Pierwsza z nich jest możliwie najdokładniej modelowana w punkcie, w którym aplikacja wskazuje na umieszczanie komunikatów w kolejce, a druga demonstruje dynamicznie tworzenie tematów-bardziej powszechny wzorzec dla aplikacji publikatorów.

Cykle życia publikowania/subskrybowania

W projektowaniu aplikacji publikowania/subskrypcji należy wziąć pod uwagę cykle życia tematów, subskrypcji, subskrybentów, publikacji, publikatorów i kolejek.

Cykl życia obiektu, taki jak subskrypcja, rozpoczyna się od jego utworzenia, a kończy wraz z jego usunięciem. Może również zawierać inne stany i zmiany, które przechodzi, takie jak tymczasowe zawieszenie, które mają tematy nadrzędne i podrzędne, są przedawane i usuwane.

Tradycyjnie obiekty WebSphere MQ , takie jak kolejki, są tworzone administracyjnie lub przez programy administracyjne korzystające z formatu komend programowalnych (PCF). Publikowanie/subskrypcja jest inne w przypadku udostępniania komend API `MQSUB` i `MQCLOSE` w celu tworzenia i usuwania subskrypcji, mając pojęcie zarządzanych subskrypcji, które nie tylko tworzą i usuwają kolejki, ale także usuwają niewykorzystane komunikaty i mają powiązania między obiektami tematu utworzonymi administracyjnie i programowo lub administracyjnie utworzonymi łańcuchami tematów.

To funkcjonalne gąsienice bogactwa funkcjonalnego dla szerokiego zakresu wymagań publikowania/subskrypcji, a także upraszcza projektowanie niektórych wspólnych wzorców aplikacji publikowania/subskrypcji. Zarządzane subskrypcje, na przykład, upraszczają programowanie i administrowanie subskrypcją, która ma trwać tylko tak długo, jak program, który go utworzył. Subskrypcje niezarządzane upraszczają programowanie w przypadku, gdy istnieje luźniejsze połączenie między publikacjami zasubskrybowanych i konsumowanych. Centralnie utworzone subskrypcje są przydatne w przypadku, gdy wzorzec jest jednym z ruchu publikacji routingu skierowanym do konsumentów na podstawie scentralizowanego modelu sterowania, na przykład wysyłania informacji o locie do zautomatyzowanych bram, podczas gdy programowo tworzone subskrypcje mogą być używane, jeśli personel bramy jest odpowiedzialny za subskrybowanie rekordów pasażerów dla tego lotu, poprzez wprowadzenie numeru lotu w bramce.

W tym ostatnim przykładzie może być odpowiednia zarządzana subskrypcja trwała: zarządzane, ponieważ subskrypcje są tworzone bardzo często i mają czytelny punkt końcowy, gdy brama jest zamykana, a subskrypcja może być programowo usunięta; trwała, aby uniknąć utraty rekordu pasażera ze względu na to, że subskrybent bramy jest wyłączony z jednego lub innego powodu.[2]. W celu zainicjowania publikacji zapisów pasażerów do bramy, ewentualny projekt będzie dla aplikacji bramowej obu subskrybować

rekordy pasażerów przy użyciu numeru bramy, i publikuje zdarzenie otwarcia bramy z wykorzystaniem numeru bramy. Wydawca odpowiada na zdarzenie otwarcia bram, publikując zapisy pasażerów-które mogą następnie trafić również do innych zainteresowanych, takich jak fakturowanie, do rejestrowania lotu, a także do obsługi klienta, do powiadomień tekstowych na telefony komórkowe pasażerskie o numerze bram.

Centralnie zarządzana subskrypcja może używać trwałego modelu niezarządzanego, kierujących listy pasażerów do bramy przy użyciu predefiniowanej kolejki dla każdej bramy.

Następujące trzy przykłady cykli życia publikowania/subskrypcji ilustrują sposób, w jaki zarządzane są nietrwałe, zarządzane trwałe i niezarządzane, trwałe subskrybenci współdziałają z subskrypcjami, tematami, kolejkami, publikatorami i menedżerem kolejek oraz w jaki sposób odpowiedzialność może być podzielona między administrację i programy subskrybentów.

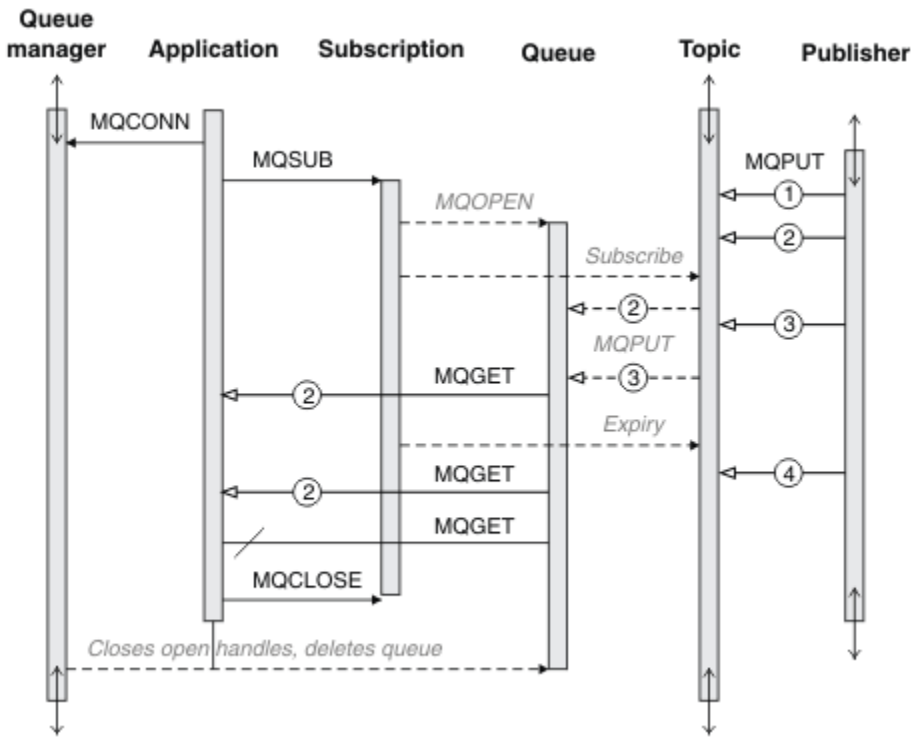
Zarządzany nietrwały subskrybent

Rysunek 56 na stronie 313 przedstawia aplikację tworzący zarządzane nietrwałe subskrypcje, otrzymując dwa komunikaty opublikowane w temacie określonym w subskrypcji i kończące się. Interakcje oznaczone kursywą szarą czcionką z strzałkami w kropce są niejawne.

Są pewne punkty, które warto zwrócić uwagę.

1. Aplikacja tworzy subskrypcję w temacie, który został już opublikowany dwukrotnie. Po otrzymaniu pierwszej publikacji subskrybent otrzymuje publikację *sekunda* , która jest aktualnie zachowaną publikacją.
2. Menedżer kolejek tworzy tymczasową kolejkę subskrypcji, a także tworzy subskrypcję dla tego tematu.
3. Subskrypcja ma limit czasu utraty ważności. Jeśli subskrypcja utraci ważność, nie będą wysyłane do tej subskrypcji żadne publikacje dotyczące tego tematu, ale subskrybent będzie nadal otrzymując komunikaty publikowane przed utratą ważności subskrypcji. Utrata ważności subskrypcji nie ma wpływu na wygaśnięcie subskrypcji.
4. Czwarta publikacja nie jest umieszczana w kolejce subskrypcji i w związku z tym ostatnia publikacja `MQGET` nie zwraca publikacji.
5. Mimo że subskrybent zamknie subskrypcję, nie zamyka połączenia z kolejką lub menedżerem kolejek.
6. Menedżer kolejek czyści wkrótce po zakończeniu działania aplikacji. Ponieważ subskrypcja jest zarządzana i nie jest trwała, kolejka subskrypcji jest usuwana.

[2] Wydawca musi wysłać rekordy pasażerów jako komunikaty trwałe, aby uniknąć innych możliwych awarii, oczywiście.



Rysunek 56. Zarządzane nietrwałe wąwozy subskrybentów

Zarządzany trwały subskrybent

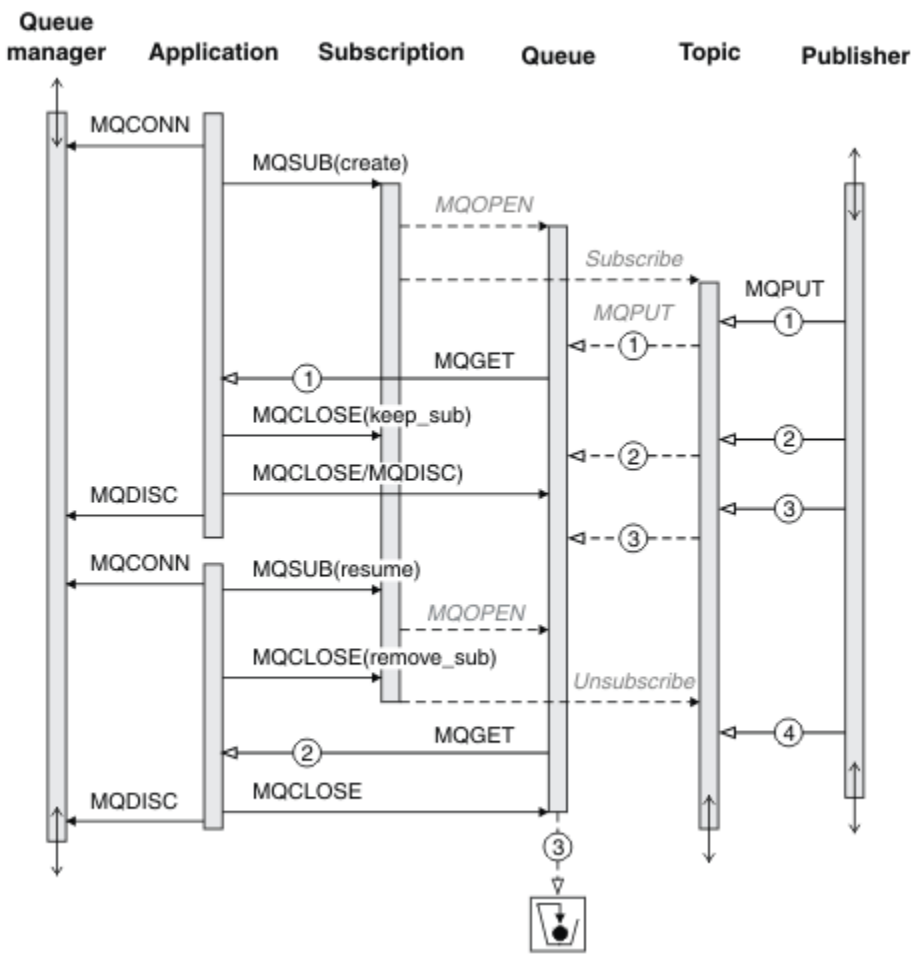
Zarządzany trwały subskrybent ma następny przykład krok dalej, a subskrypcja zarządzana zachowa zakończenie i restartowanie aplikacji subskrybującej.

Istnieją nowe punkty do odnoenia.

1. W tym przykładzie, w przeciwieństwie do ostatniego, temat publikowania nie istniał przed zdefiniowaniem go w subskrypcji.
2. Gdy subskrybent zostanie zamknięty po raz pierwszy, subskrypcja zostanie zamknięta z opcją MQCO_KEEP_SUB. To jest zachowanie domyślne w przypadku niejawnego zamykania zarządzanej subskrypcji trwałej.
3. Gdy subskrybent wznawia subskrypcję, kolejka subskrypcji jest ponownie otwierana.
4. Nowa publikacja 2, umieszczona w kolejce przed jej ponownym otwarciom, jest dostępna dla produktu MQGET, nawet po usunięciu subskrypcji.

   Mimo że subskrypcja jest trwała, subskrybent niezawodnie odbiera wszystkie wiadomości wysłane przez publikatora tylko wtedy, gdy *obie* subskrypcje są trwałe, a komunikaty trwałe. Trwałość komunikatu zależy od ustawienia pola Persistent w MQMD komunikatu wysłanego przez publikatora. Abonent nie ma nad tym żadnej kontroli.
5. Zamknięcie subskrypcji z flagą MQCO_REMOVE_SUB powoduje usunięcie subskrypcji, zatrzymanie kolejnych publikacji umieszczonych w kolejce subskrypcji. Gdy kolejka subskrypcji jest zamknięta, menedżer kolejek usuwa nieprzeczytaną publikację 3, a następnie usuwa kolejkę. Czynność jest równoważna administracyjnie usuwaniu subskrypcji.

   Uwaga: Nie należy usuwać kolejki ręcznie lub MQCLOSE z opcją MQCO_DELETElub MQCO_PURGE_DELETE. Widoczne szczegóły implementacji zarządzanej subskrypcji nie są częścią obsługiwanego interfejsu produktu WebSphere MQ . Menedżer kolejek nie może zarządzać subskrypcją niezawodnie, chyba że ma pełną kontrolę.



Rysunek 57. Zarządzane trwałe elementy życia subskrybentów

Niezarządzany trwały subskrybent

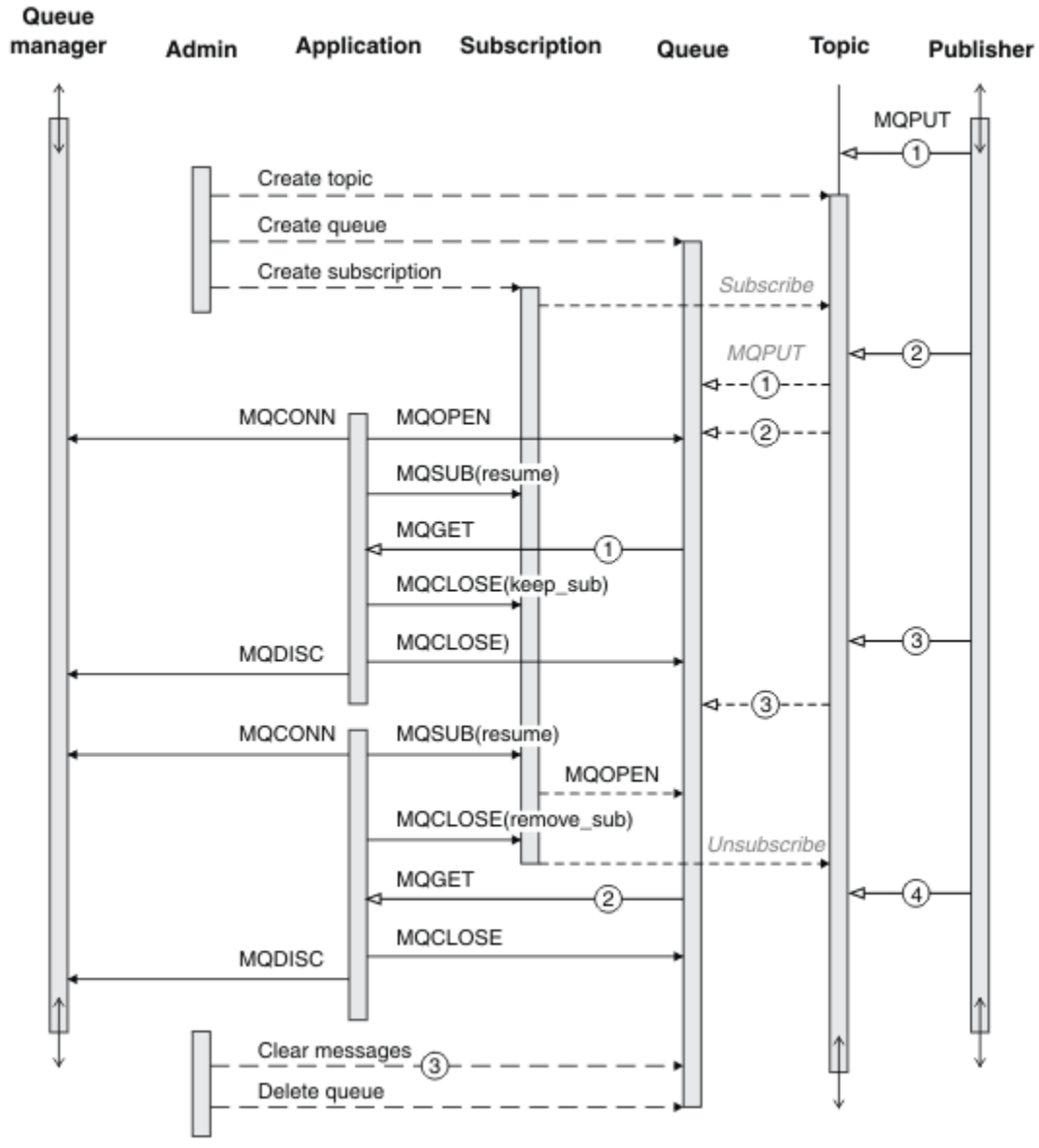
W trzecim przykładzie zostanie dodany administrator: niezarządzany trwały subskrybent. Dobrym przykładem jest pokazanie, w jaki sposób administrator może wchodzić w interakcje z aplikacją publikowania/subskrypcji.

Punkty do nota są wymienione.

1. Publikator umieszcza komunikat 1w temacie, który później staje się powiązany z obiektem tematu, który jest używany do subskrypcji. Obiekt tematu definiuje łańcuch tematu, który jest zgodny z tematem, który został opublikowany przy użyciu znaków wieloznacznych.
2. W temacie znajduje się zachowana publikacja.
3. Administrator tworzy obiekt tematu, kolejkę i subskrypcję. Obiekt tematu i kolejka muszą zostać zdefiniowane przed subskrypcją.
4. Aplikacja otwiera kolejkę powiązaną z subskrypcją i przekazuje MQSUB uchwyt kolejki. Może ona, alternatywnie, po prostu otworzyć subskrypcję, przekazując jej uchwyt kolejki MQHO_NONE. Konwersacja nie jest prawdziwa, nie można wznowić subskrypcji, przekazując jej tylko uchwyt kolejki bez nazwy subskrypcji-kolejka może mieć wiele subskrypcji.
5. Aplikacja otwiera subskrypcję przy użyciu opcji MQSO_RESUME , mimo że po raz pierwszy otwarto subskrypcję. Wznawiana jest administracyjna subskrypcja utworzona przez administratora.

6. Subskrybent otrzymuje zachowaną publikację, 1. Publikacja 2, mimo że została opublikowana przed otrzymaniem publikacji przez subskrybenta, została opublikowana po rozpoczęciu subskrypcji i jest drugą publikacją w kolejce subskrypcji.

   Uwaga: Jeśli zachowana publikacja nie zostanie opublikowana jako komunikat trwały, zostanie ona utracona po zrestartowaniu menedżera kolejek.
7. W tym przykładzie subskrypcja jest trwała. Program może utworzyć niezarządzaną, nietrwałą subskrypcję. To oczywiste, że nie jest to coś, co może zrobić administrator.
8. Efektem działania opcji MQCO_REMOVE_SUB przy zamykaniu subskrypcji jest usunięcie subskrypcji tak, jakby jej administrator usunował tę subskrypcję. Spowoduje to zatrzymanie kolejnych publikacji wysyłanych do kolejki, ale nie ma wpływu na publikacje, które są już w kolejce, nawet jeśli kolejka jest zamknięta, w przeciwieństwie do trwałej subskrypcji *zarządzanej* .
9. Później administrator usunie pozostały komunikat 3i usunie kolejkę.



Rysunek 58. Niezarządzane trwałe elementy życia subskrybentów

Normalny wzorzec dla subskrypcji niezarządzanej jest przeznaczony do przechowywania w kolejce i subskrypcji, który ma być wykonywany przez administratora. Zwykle nie jest podejmowana próba

emulowania zachowania zarządzanego subskrybenta oraz programowego programowania kolejek i subskrypcji w kodzie aplikacji. Jeśli użytkownik musi napisać logikę zarządzania, należy sprawdzić, czy możliwe jest osiągnięcie tych samych wyników przy użyciu wzorca zarządzanego. Nie jest łatwo pisać ściśle zsynchronizowany, całkowicie niezawodny kod zarządzania. Łatwiejsze jest późniejsze przechyłowanie, ręcznie lub za pomocą zautomatyzowanego programu do zarządzania, kiedy można mieć pewność, że wiadomości, subskrypcje i kolejki mogą być po prostu usunięte, niezależnie od ich stanu.

Właściwości komunikatu publikowania/subskrypcji

Wiele właściwości komunikatu jest związanych z przesyłaniem komunikatów w trybie publikowania/subskrypcji produktu WebSphere MQ .

Znacznik PubAccounting

Jest to wartość, która będzie znajdować się w polu AccountingToken deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. Element AccountingToken jest częścią kontekstu tożsamości komunikatu. Więcej informacji na temat kontekstu komunikatów zawiera sekcja “Kontekst komunikatu” na stronie 39. Więcej informacji na temat pola AccountingToken w strukturze MQMD można znaleźć w sekcji AccountingToken.

PubApplIdentityData

Jest to wartość, która będzie w polu ApplIdentityData deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. ApplIdentityDane są częścią kontekstu tożsamości komunikatu. Więcej informacji na temat kontekstu komunikatów zawiera sekcja “Kontekst komunikatu” na stronie 39. Więcej informacji na temat pola ApplIdentityData w strukturze MQMD zawiera sekcja ApplIdentityDane.

Jeśli opcja MQSO_SET_IDENTITY_CONTEXT nie zostanie podana, dane ApplIdentity, które zostaną ustawione w każdym komunikacie opublikowanym dla tej subskrypcji, są puste, jako domyślne informacje o kontekście.

Jeśli zostanie podana opcja MQSO_SET_IDENTITY_CONTEXT, użytkownik PubApplIdentityData jest generowany przez użytkownika, a pole to zawiera pole wejściowe zawierające dane ApplIdentity, które mają być ustawione w każdej publikacji dla tej subskrypcji.

PubPriority

Jest to wartość, która będzie należeć do pola Priorytet deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. Więcej informacji na temat pola Priorytet w strukturze MQMD zawiera sekcja Priorytet.

Wartość musi być większa lub równa zero; zero jest najniższym priorytetem. Można również użyć następujących wartości specjalnych:

- MQPRI_PRIORITY_AS_Q_DEF-W przypadku, gdy kolejka subskrypcji jest udostępniana w polu Hobj w wywołaniu MQSUB, a nie jest zarządzana, priorytet dla komunikatu jest przyjmowany z atrybutu DefProirity tej kolejki. Jeśli identyfikowana kolejka jest kolejką klastra lub istnieje więcej niż jedna definicja w ścieżce rozstrzygania nazw kolejek, priorytet jest określany, gdy komunikat publikacji jest umieszczany w kolejce zgodnie z opisem w polu Priorytet w strukturze MQMD. Jeśli wywołanie MQSUB korzysta z uchwytu zarządzanego, priorytet komunikatu jest przyjmowany z atrybutu DefPriority w kolejce modelowej powiązanej z subskrybowanym tematem.
- MQPRI_PRIORITY_AS_PUBLISHED-priorytet dla komunikatu jest priorytetem pierwotnej publikacji. Jest to początkowa wartość tego pola.

Identyfikator SubCorrel

Ostrzeżenie: Identyfikator korelacji może być przekazywany tylko między menedżerami kolejek w klastrze publikowania/subskrypcji, a nie w hierarchii.

Wszystkie publikacje wysłane w celu dopasowania do tej subskrypcji będą zawierać ten identyfikator korelacji w deskryptorze komunikatu. Jeśli wiele subskrypcji korzysta z tej samej kolejki w celu uzyskania ich publikacji, użycie identyfikatora MQGET według identyfikatora korelacji umożliwia uzyskanie tylko tych publikacji, które mają zostać uzyskane. Ten identyfikator korelacji może być wygenerowany przez menedżera kolejek lub przez użytkownika.

Jeśli opcja MQSO_SET_CORREL_ID nie jest określona, identyfikator korelacji jest generowany przez menedżer kolejek, a pole to jest polem wyjściowym zawierającym identyfikator korelacji, który zostanie ustawiony w każdym komunikacie opublikowanym dla tej subskrypcji.

Jeśli zostanie podana opcja MQSO_SET_CORREL_ID, identyfikator korelacji jest generowany przez użytkownika, a pole to jest polem wejściowym zawierającym identyfikator korelacji, który ma być ustawiony w każdej publikacji dla tej subskrypcji. W takim przypadku, jeśli pole zawiera wartość MQCI_NONE, to identyfikator korelacji, który zostanie ustawiony w każdym komunikacie opublikowanym dla tej subskrypcji, będzie identyfikatorem korelacji utworzonym przez oryginalną nazwę komunikatu.

Jeśli określona jest opcja MQSO_GROUP_SUB, a określony identyfikator korelacji jest taki sam, jak istniejąca grupowa subskrypcja za pomocą tej samej kolejki i nakładającego się łańcucha tematu, tylko najbardziej znacząca subskrypcja w grupie jest udostępniana z kopią publikacji.

Dane SubUser

To jest dane użytkownika subskrypcji. Dane podane w subskrypcji w tym polu zostaną dołączone jako właściwość komunikatu danych MQSubUserw każdej publikacji wysłanej do tej subskrypcji.

Właściwości publikacji

Tabela 44 na stronie 317 zawiera listę właściwości publikacji, które są udostępniane wraz z komunikatem do publikacji.

Dostęp do tych właściwości można uzyskać bezpośrednio z folderu **MQRFH2** lub pobrać za pomocą programu MQINQMP. Produkt MQINQMP akceptuje nazwę właściwości lub nazwę **MQRFH2** jako nazwę właściwości, która ma zostać zapytana.

Tabela 44. Właściwości publikacji

Nazwa właściwości	Nazwa MQRFH2	Typ	Opis
MQTopicString	mqps.Top	MQTYPE_STRING	Łańcuch tematu
MQSubUserData	mqps.Sud	MQTYPE_STRING	Dane użytkownika subskrybenta
MQIsRetained	mqps.Ret	MQTYPE_BOOLEAN	Zachowana publikacja
MQPubOptions	mqps.Pub	MQTYPE_INT32	Opcje publikacji
MQPubLevel	mqps.Pbl	MQTYPE_INT32	Poziom publikacji
MQPubTime	mqpse.Pts	MQTYPE_STRING	Czas publikacji
MQPubSeqNum	mqpse.Seq	MQTYPE_INT32	Numer kolejny publikacji
MQPubStrIntData	mqpse.Sid	MQTYPE_STRING	Łańcuch/liczba całkowita dodana przez publikator
MQPubFormat	mqpse.Pfmt	MQTYPE_INT32	Format komunikatu: MQRFH1 MQRFH2 PCF

Porządkowanie komunikatów

W przypadku konkretnego tematu komunikaty są publikowane przez menedżer kolejek w takiej samej kolejności, w jakiej są odbierane z aplikacji publikowania (z zastrzeżeniem zmiany kolejności w oparciu o priorytet komunikatu).

Porządkowanie komunikatów zwykle oznacza, że każdy subskrybent odbiera komunikaty z określonego menedżera kolejek, dotyczące określonego tematu, od konkretnego publikatora w kolejności, w jakiej są one publikowane przez tego publikatora.

Jednak, podobnie jak w przypadku wszystkich komunikatów produktu WebSphere MQ , komunikaty mogą być dostarczane w sposób sporadycznie poza kolejną kolejką. Sytuacja taka może wystąpić w następujących sytuacjach:

- Jeśli odsyłacz w sieci zostanie wyłączony, a kolejne komunikaty zostaną przekierowane na inny odsyłacz,
- Jeśli kolejka zostanie tymczasowo zapełniona lub zablokowana, tak aby komunikat został umieszczony w kolejce niedostarczonych komunikatów, a tym samym opóźniony, podczas gdy kolejne komunikaty są przekazywane prosto przez użytkownika.
- Jeśli administrator usunie menedżera kolejek, gdy publikatorzy i subskrybenci nadal działają, powodując umieszczenie w kolejce komunikatów, które mają zostać umieszczone w kolejce niedostarczonych komunikatów i subskrypcje, które mają zostać przerwane.

Jeśli te okoliczności nie mogą wystąpić, publikacje są zawsze dostarczane w porządku.

Uwaga: Nie można używać zgrupowanych ani segmentowanych komunikatów z publikowania/subskrybowania.

Przechwytywanie publikacji

Można przechwycić publikację, zmodyfikować ją, a następnie ponownie opublikować, zanim trafi do dowolnego innego subskrybenta.

Aby można było wykonać jedną z następujących czynności, warto przechwycić publikację, zanim zostanie on osiągnięty przez subskrybenta.

- Dołączanie dodatkowych informacji do wiadomości
- Zablokuj komunikat
- Transformowanie komunikatu

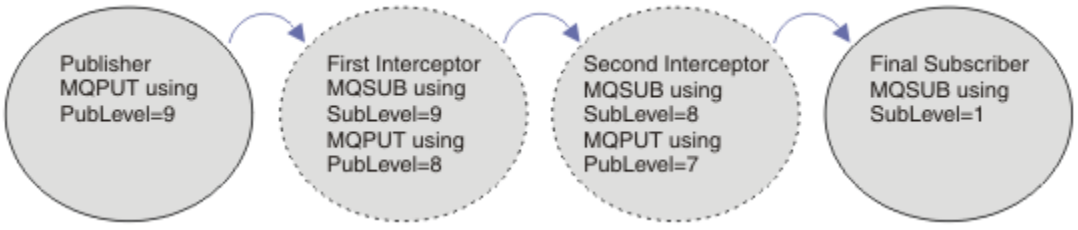
Tę samą operację można wykonać dla każdego komunikatu lub udostępnić operację w zależności od subskrypcji, komunikatu lub nagłówka komunikatu.

Odsyłacze pokrewne

MQ_PUBLISH_EXIT-wyjście publikowania

Poziomy subskrypcji

Ustaw poziom subskrypcji subskrypcji, aby przechwycić publikację, zanim zostanie osiągnięty jej finalny subskrybent. Przechwytywacz subskrybent subskrybuje wyższy poziom subskrypcji, a republiki na niższym poziomie publikacji. Budowanie łańcucha przechwytującego subskrybentów w celu przetwarzania komunikatów w publikacji przed dostarczyniem ich do końcowych subskrybentów.



Rysunek 59. Sekwencja przechwytujących subskrybentów

Aby przechwycić publikację, należy użyć atrybutu **MQSD** SubLevel . Po przechwyceniu komunikatu może on zostać przekształcony, a następnie ponownie opublikowany na niższym poziomie publikacji, zmieniając atrybut **MQPMO** PubLevel . Następnie komunikat trafia do końcowych subskrybentów lub zostanie ponownie przechwycony przez subskrybent pośredni na niższym poziomie subskrypcji.

Przechwytywacz subskrybent zwykle transformuje komunikat przed ponownym jego opublikowaniem. Sekwencja przechwytujących subskrybentów tworzy przepływ komunikatów. Alternatywnie można nie ponownie opublikować przechwyconej publikacji: Subskrybenty na niższych poziomach subskrypcji nie otrzymają komunikatu.

Upewnij się, że przechwytywacz odbiera publikacje przed innymi subskrybentami. Ustaw poziom subskrypcji przechwytywacza wyżej niż inni subskrybenci. Domyślnie subskrybenci mają SubLevel z 1. Najwyższa wartość to 9. Publikacja musi rozpoczynać się od PubLevel co najmniej tak samo wysoki, jak najwyższy SubLevel. Opublikuj początkowo z domyślnym PubLevel produktu 9.

- Jeśli w temacie został przechwycony subskrybent, ustaw wartość SubLevel na 9.
- W przypadku wielu przechwytywanych aplikacji w danym temacie należy ustawić niższą wartość SubLevel dla każdego kolejnego przechwytującego subskrybenta.
- Istnieje możliwość zaimplementowania maksymalnie 8 przechwytywanych aplikacji z poziomem subskrypcji od 9 w dół do 2 włącznie. Odbiorca końcowy komunikatu ma wartość SubLevel produktu 1.

Przechwytywacz o najwyższym poziomie subskrypcji, który jest równy lub niższy niż PubLevel publikacji, otrzymuje publikację w pierwszej kolejności. Skonfiguruj tylko jeden przechwytywacz subskrybenta dla tematu na określonym poziomie subskrypcji. Posiadanie wielu subskrybentów na określonym poziomie subskrypcji powoduje, że wiele kopii publikacji jest wysyłanych do końcowego zestawu aplikacji subskrybujących.

Subskrybent o wartości SubLevel produktu 0 jest używany jako catchall. Otrzymuje on publikację, jeśli żaden końcowy subskrybent nie otrzyma komunikatu. Subskrybent o wartości SubLevel produktu 0 może być używany do monitorowania publikacji, które nie zostały odebrane przez innych subskrybentów.

Programowanie przechwytującego subskrybenta

Użyj opcji subskrypcji opisanych w sekcji Tabela 45 na stronie 319.

Tabela 45. Opcje subskrypcji dla przechwytywaczy subskrybentów

Opcja subskrypcji	**Uwagi**
Opcje MQSO_SET_CORREL_ID i SubCorrelId ustawione na wartość MQCI_NONE	Zachowaj wartość CorrelId przechwyconej publikacji tak samo, jak oryginalna publikacja. **Uwaga:** Nie można przekazać identyfikatora korelacji publikacji w hierarchii. Pole jest używane przez menedżer kolejek.
Parametr PubPriority ustawiony na wartość MQPRI_PRIORITY_AS_PUBLISHED	Należy zachować priorytet przechwyconej publikacji, tak samo jak oryginalna publikacja.

Opcje w programie Tabela 45 na stronie 319 muszą być używane przez wszystkie przechwytujące subskrybenty. Wynika to z tego, że identyfikator korelacji i priorytet komunikatu nie są modyfikowane z ustawienia pierwotnego publikatora.

Po przetworzeniu publikacji przez subskrybent zostanie ponownie wysłany komunikat do tego samego tematu na poziomie PubLevel o wartości niższej niż SubLevel w ramach własnej subskrypcji. Jeśli przechwytywający subskrybent ustawił element SubLevel w produkcie 9, zostanie on ponownie opublikowany przez użytkownika o wartości PubLevel produktu 8.

Aby ponownie opublikować komunikat, wymagane jest kilka fragmentów informacji z oryginalnej publikacji. Ponownie wykorzystaj tę samą **MQMD** co w oryginalnym komunikacie i ustaw MQPMO_PASS_ALL_CONTEXT , aby zapewnić, że wszystkie informacje w **MQMD** zostaną przekazane do następnego subskrybenta. Skopiuj wartości z właściwości komunikatu, które są wyświetlane w Tabela

46 na stronie 320 , do odpowiednich pól ponownie opublikowanego komunikatu. Przechwytywacz subskrybenta może zmienić te wartości. Użyj operatora OR, aby dodać dodatkowe wartości do pola **MQPMO**.Opcje , aby połączyć opcje umieszczania komunikatu.

Należy jawnie otworzyć kolejkę publikacji, a nie używać zarządzanej kolejki publikacji. Nie można ustawić MQSO_SET_CORREL_ID dla kolejki zarządzanej. Nie można również ustawić MQOO_SAVE_ALL_CONTEXT w kolejce zarządzanej. Zapoznaj się z fragmentami kodu wymienionymi w sekcji “Przykłady” na stronie 320.

Tabela 46. Wartości MQPUT dla ponownie opublikowanych komunikatów

Ponownie publikuj komunikat przy użyciu komendy MQPUT	**Informacje w komunikacie o publikacji**
MQOD.ObjectString	Właściwość komunikatu MQTopicString
MQPMO.Options	Właściwość komunikatu MQPubOptions

Ostatni subskrybent ma możliwość wyboru ustawienia opcji subskrypcji w inny sposób. Na przykład może ona jawnie ustawić priorytet publikowania, a nie na MQPRI_PRIORITY_AS_PUBLISHED. Ustawienia subskrybenta końcowego mają wpływ tylko na publikację z poziomu końcowego przechwytującego subskrybenta w łańcuchu.

Zachowane publikacje

Zachowana publikacja musi zostać zachowana po przechwyceniu, kopiując oryginalne opcje put-message do ponownie opublikowanego komunikatu.

Opcja MQPMO_RETAIN jest ustawiana przez publikator. Każdy przechwytywacz subskrybenta musi przesłać MQPubOptions do opcji put-message z ponownie opublikowanego komunikatu, jak to pokazano na rysunku Tabela 46 na stronie 320. Kopiowanie opcji put-message powoduje zachowanie opcji ustawionych przez pierwotnego publikatora, w tym także informacje o tym, czy ma być zachowana publikacja.

Gdy publikacja kończy swój fragment łańcucha przechwytującego subskrybentów, i jest dostarczany do końcowych subskrybentów, to w końcu zostaje zachowana. Nowi subskrybenci, na poziomie SubLevel 1, żądający zachowanej publikacji, otrzymują ją bez żadnych dodatkowych przechwytów. Subskrybenty SubLevel większe niż 1 nie są wysyłane do zachowanej publikacji. W związku z tym zachowana publikacja nie jest modyfikowana przez łańcuch przechwytywania subskrybentów po raz drugi przez cały czas.

Przykłady

Przykłady to fragmenty kodu, które można połączyć w celu zbudowania przechwytywacza. Kod jest zapisywany jako krótki, a nie jako jakość produkcji.

Dyrektywy preprocesora w produkcie Rysunek 60 na stronie 321 definiują dwie właściwości, które mają zostać wyodrębnione z komunikatów publikacji, które są wymagane przez wywołanie MQI produktu MQINQMP .

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
#define      MQPUBOPTIONS        (MQPTR)(char*) "MQPubOptions",\
                                 0,\
                                 12,\
                                 MQVS_NULL_TERMINATED,\
                                 MQCCSI_APPL
#define      MQTOPICSTRING       (MQPTR)(char*) "MQTopicString",\
                                 0,\
                                 13,\
                                 MQVS_NULL_TERMINATED,\
                                 MQCCSI_APPL
```

Rysunek 60. Dyrektywy preprocesora

Rysunek 61 na stronie 321 zawiera listę deklaracji używanych w fragmentach kodu. Deklaracje są standardowe dla aplikacji produktu WebSphere MQ , z wyjątkiem podświetlonych terminów.

Podświetlone opcje umieszczania i pobierania są inicjowane w celu przekazania całego kontekstu. Podświetlone pola MQTOPICSTRING i MQPUBOPTIONS to inicjatory MQCHARV dla nazw właściwości, które są zdefiniowane w dyrektywach preprocesora. Nazwy są przekazywane do produktu MQINQMP.

```
int main(int argc, char **argv) {
    MQLONG   Reason            = MQRC_NONE;
    MQLONG   CompCode          = MQCC_OK;
    MQHCONN  Hcon              = MQHC_UNUSABLE_HCONN;
    MQCHAR   QMName[49]        = " ";
    MQCMHO   CrtMsgHOpts       = {MQCMHO_DEFAULT};
    MQHMSG   Hmsg              = MQHM_NONE;
    MQMD     md                = {MQMD_DEFAULT};
    MQHOBJ   gHobj             = MQHO_NONE;
    MQOD     getOD             = {MQOD_DEFAULT};
    MQGMO    gmo               = {MQGMO_DEFAULT};
    MQLONG   GO_Options        = MQOO_INPUT_AS_Q_DEF
                               | MQOO_FAIL_IF_QUIESCING
                               | MQOO_SAVE_ALL_CONTEXT;
    MQLONG   GC_Options        = MQCO_DELETE_PURGE;
    MQHOBJ   Hsub              = MQHO_NONE;
    MQSD     sd                = {MQSD_DEFAULT};
    MQLONG   SC_Options        = MQCO_NONE;
    MQHOBJ   pHobj             = MQHO_NONE;
    MQOD     putOD             = {MQOD_DEFAULT};
    MQLONG   PO_Options        = MQOO_OUTPUT
                               | MQOO_FAIL_IF_QUIESCING
                               | MQOO_PASS_ALL_CONTEXT;
    MQLONG   PC_Options        = MQCO_NONE;
    MQPMO    pmo               = {MQPMO_DEFAULT};
    MQIMPO   InqPropOpts       = {MQIMPO_DEFAULT};
    MQPD     PropDesc          = {MQPD_DEFAULT};
    MQLONG   Type              = MQTYPE_AS_SET;
    MQCHARV  TopStrProp        = {MQTOPICSTRING};
    MQCHARV  PubOptProp        = {MQPUBOPTIONS};
    MQLONG   DataLength        = 0;
    MQBYTE   buffer[256]       = "";
    MQLONG   buflen            = sizeof(buffer) - 1;
    MQLONG   messlen           = 0;
    char     TopStrBuf[256]    = "Initial value";
    int      i                 = 0;
```

Rysunek 61. Deklaracje

Inicjacje, które nie są łatwo wykonywane w deklaracjach, są wyświetlane w produkcie Rysunek 62 na stronie 322. Podświetlone wartości wymagają wyjaśnienia.

SYSTEM.NDURABLE.MODEL.QUEUE
W tym przykładzie zamiast korzystania z produktu MQSUB w celu otwarcia zarządzanej nietrwałej subskrypcji, kolejka modelowa SYSTEM.NDURABLE.MODEL.QUEUEjest używana do tworzenia tymczasowej kolejki dynamicznej. Jego uchwyt jest przekazywany do produktu MQSUB. Bezpośrednio otwierając kolejkę, można zapisać cały kontekst komunikatu i ustawić opcję subskrypcji, MQSO_SET_CORREL_ID.

MQGMO_CURRENT_VERSION
Ważne jest, aby używać bieżącej wersji większości struktur produktu WebSphere MQ . Pola, takie jak `gmo.MsgHandle` , są dostępne tylko w najnowszej wersji struktur sterujących.

MQGMO_PROPERTIES_IN_HANDLE
Łańcuch tematu i opcje umieszczania komunikatów ustawione w oryginalnej publikacji mają być pobierane przez przechwytywanego subskrybenta przy użyciu właściwości komunikatu. Alternatywnym rozwiązaniem jest bezpośrednie odczytanie struktury **MQRFH2** w komunikacie.

MQSO_SET_CORREL_ID
Użyj `MQSO_SET_CORREL_ID` w połączeniu z,

```
memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
```

Efektem tych opcji jest przekazanie identyfikatora korelacji. Identyfikator korelacji ustawiony przez pierwotny publikator jest umieszczany w polu identyfikatora korelacji publikacji odebranej przez przechwytywanego subskrybenta. Każdy przechwytywany subskrybent przechodzi na ten sam identyfikator korelacji. Następnie końcowy subskrybent ma opcję otrzymywania tego samego identyfikatora korelacji.

Uwaga: Jeśli publikacja jest przekazywana za pośrednictwem hierarchii publikowania/subskrypcji, identyfikator korelacji nigdy nie jest zachowywany.

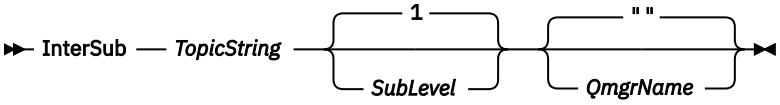
MQPRI_PRIORITY_AS_PUBLISHED
Publikacja jest umieszczana w kolejce publikacji z tym samym priorytetem komunikatów, w którym został opublikowany.

```
strncpy(getOD.ObjectName,    "SYSTEM.NDURABLE.MODEL.QUEUE",
        sizeof(getOD.ObjectName));
gmo.Version                = MQGMO_VERSION_4;
gmo.Options                = MQGMO_WAIT
                           | MQGMO_PROPERTIES_IN_HANDLE
                           | MQGMO_CONVERT;
gmo.WaitInterval           = 30000;
sd.Options                 = MQSO_CREATE
                           | MQSO_FAIL_IF_QUIESCING
                           | MQSO_SET_CORREL_ID;
sd.PubPriority             = MQPRI_PRIORITY_AS_PUBLISHED;
sd.Version                 = MQSD_VERSION_1;
memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
putOD.ObjectType             = MQOT_TOPIC;
putOD.ObjectString.VSPtr     = &TopStrBuf;
putOD.ObjectString.VSBufSize = sizeof(TopStrBuf);
putOD.ObjectString.VSLength  = MQVS_NULL_TERMINATED;
putOD.ObjectString.VSCCSID   = MQCCSI_APPL;
putOD.Version                = MQOD_VERSION_4;
pmo.Version                = MQPMO_VERSION_3;
```

Rysunek 62. Inicjalizacje

Rysunek 63 na stronie 323 przedstawia fragment kodu w celu odczytania parametrów wiersza komend, zakończenie inicjowania i utworzenie subskrypcji przechwytującej.

Uruchom program z komendą,



Aby obsługa błędów była niezauważalna, kod przyczyny z każdego wywołania MQI jest przechowywany w innym elemencie tablicy. Po każdym wywołaniu kod zakończenia jest testowany, a jeśli wartością jest MQCC_FAIL, element sterujący powoduje wyjście z bloku kodu do `{ } while(0)` .

Dwie warte uwagi linie kodu to:

pmo.PubLevel = sd.SubLevel - 1;
Ustawia poziom publikacji dla ponownie opublikowanego komunikatu na jeden mniejszy niż poziom subskrypcji przechwytującego subskrybenta.

gmo.MsgHandle = Hmsg;
Udostępnia uchwyt komunikatu dla produktu MQGET w celu zwrócenia właściwości komunikatu.

```
do {
   printf("Intercepting subscriber start\n");
   if (argc < 2) {
       printf("Required parameter missing - topic string\n");
       exit(99);
   } else {
       sd.ObjectString.VSPtr    = argv[1];
       sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
       printf("TopicString = %s\n", sd.ObjectString.VSPtr);
   }
   if (argc > 2) {
       sd.SubLevel  = atoi(argv[2]);
       pmo.PubLevel = sd.SubLevel - 1;
       printf("SubLevel is %d, PubLevel is %d\n", sd.SubLevel, pmo.PubLevel);
   }
   if (argc > 3)
       strncpy(QMName, argv[3], sizeof(QMName));
   MQCONN(QMName, &Hcon, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQOPEN(Hcon, &getOD, GO_Options, &gHobj, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQSUB(Hcon, &sd, &gHobj, &Hsub, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQCRTMH(Hcon, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   gmo.MsgHandle = Hmsg;
```

Rysunek 63. Przygotowanie do przechwycenia publikacji

Główny fragment kodu, Rysunek 64 na stronie 324, pobiera komunikaty z kolejki publikacji. Wysyła on zapytania do właściwości komunikatu i ponownie publikuje komunikaty przy użyciu łańcucha tematu oraz oryginalną właściwość **MQPMO**.opcja publikacji.

W tym przykładzie w publikacji nie jest wykonywana żadna transformacja. Łańcuch tematu ponownie opublikowanej publikacji jest zawsze zgodny z łańcuchem tematu przechwytującą subskrybent subskrybowany. Jeśli przechwytywacz subskrybenta jest odpowiedzialny za przechwytywanie wielu subskrypcji wysłanych do tej samej kolejki publikacji, może być konieczne wysłanie zapytania do łańcucha tematu w celu odróżnienia publikacji, które są zgodne z różnymi subskrypcjami.

Wywołania programu MQINQMP są podświetlone. Łańcuch tematu i publikacja właściwości opcji umieszczania komunikatów są zapisywane bezpośrednio w strukturach sterujących wyjścia. Jedynym powodem zmiany pola długości MQCHARV putOD.ObjectString z jawnej długości do łańcucha zakończonego znakiem NULL jest użycie printf do wyprowadzania łańcucha.

```
while (CompCode != MQCC_FAILED) {
   memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
   memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
   md.Encoding       = MQENC_NATIVE;
   md.CodedCharSetId = MQCCSI_Q_MGR;
   printf("MQGET : %d seconds wait time\n", gmo.WaitInterval/1000);
   MQGET(Hcon, gHobj, &md, &gmo, buflen, buffer, &messlen,
       &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   buffer[messlen] = '\0';
   MQINQMP(Hcon, Hmsg,    &InqPropOpts, &TopStrProp, &PropDesc, &Type,
       putOD.ObjectString.VSBufSize, putOD.ObjectString.VSPtr,
       &(putOD.ObjectString.VSLength), &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   memset((void *)((MQLONG)(putOD.ObjectString.VSPtr)
       + putOD.ObjectString.VSLength),'\0',1);
   putOD.ObjectString.VSLength = MQVS_NULL_TERMINATED;
   MQINQMP(Hcon, Hmsg, &InqPropOpts, &PubOptProp, &PropDesc, &Type,
       sizeof(pmo.Options), &(pmo.Options), &DataLength,
       &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQOPEN(Hcon, &putOD, PO_Options, &pHobj, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   printf("Republish message <%s> on topic <%s> with options %d\n",
       buffer, putOD.ObjectString.VSPtr, pmo.Options);
   MQPUT(Hcon, pHobj, &md, &pmo, messlen, buffer, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQCLOSE(Hcon, &pHobj, PC_Options, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
   break;
}
```

Rysunek 64. Przechwyć publikację i ponownie opublikuj

Końcowy fragment kodu jest wyświetlany w programie Rysunek 65 na stronie 324.

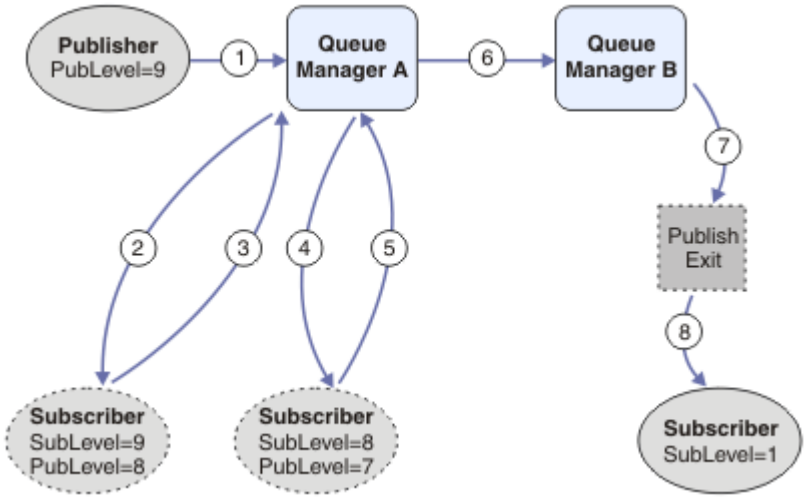
```
    } while (0);
    if (CompCode == MQCC_FAILED && Reason != MQRC_NO_MSG_AVAILABLE)
        printf("MQI Call failed with reason code %d\n", Reason);
    if (Hsub!= MQHO_NONE)
        MQCLOSE(Hcon, &Hsub, SC_Options, &CompCode, &Reason);
    if (Hcon!= MQHC_UNUSABLE_HCONN)
        MQDISC(&Hcon, &CompCode, &Reason);
}
```

Rysunek 65. Zakończenie

Przechwytywanie publikacji i rozproszonego publikowania/subskrypcji

Po wdrożeniu przechwytywanych subskrybentów lub publikowania wyjść do rozproszonej topologii publikowania/subskrypcji należy zastosować prosty wzorzec. Wdróż przechwytywanie subskrybentów w tych samych menedżerach kolejek, co publikatory, i opublikuj wyjścia w tych samych menedżerach kolejek co końcowe subskrybenci.

W produkcie Rysunek 66 na stronie 325 są wyświetlane dwa menedżery kolejek połączone w klastrze publikowaniusubskrypcji. Publikator tworzy publikację do tematu klastra na poziomie publikacji 9. Strzałki numerowane przedstawiają sekwencję kroków podjętych przez publikację w miarę przepływów do subskrybentów tematu klastra. Publikacja jest przechwytywana przez subskrybenta za pomocą elementu `Sublevel` 9 i ponownie publikowana za pomocą programu `Publevel` 8. Jest on ponownie przechwytywany przez subskrybenta na poziomie `Podpoziom` 8. Subskrybent zostanie ponownie opublikowany na poziomie `Publevel` 7. Subskrybent proxy udostępniony przez menedżer kolejek przekazuje publikację do menedżera kolejek B, w którym oprócz subskrybenta końcowego wdrożono wyjście publikowania. Publikacja jest przetwarzana przez wyjście publikowania, zanim zostanie ostatecznie odebrana przez końcowego subskrybenta na poziomie `Podpoziom` 1. Przechwytywani subskrybenci i wyjście publikowania są wyświetlane z niepoprawnymi konturami.



Rysunek 66. Wyjście przechwycenia i publikowania w klastrze

Celem prostego wzorca jest dla każdego abonenta, który otrzymuje publikację, aby otrzymać identyczną publikację. Publikacja przechodzi przez tę samą sekwencję transformacji niezależnie od miejsca, w którym subskrybent jest połączony. Prawdopodobnie chcesz uniknąć zmiany sekwencji transformacji, w zależności od tego, gdzie są połączone publikatory lub finałowe subskrybenci. Rozsądnym wyjątkiem byłoby dostosowanie publikacji w końcu dostarczonej do każdego pojedynczego abonenta. Użyj wyjścia publikowania, aby dostosować publikację w oparciu o kolejkę, do której ostatecznie została dostarczona publikacja.

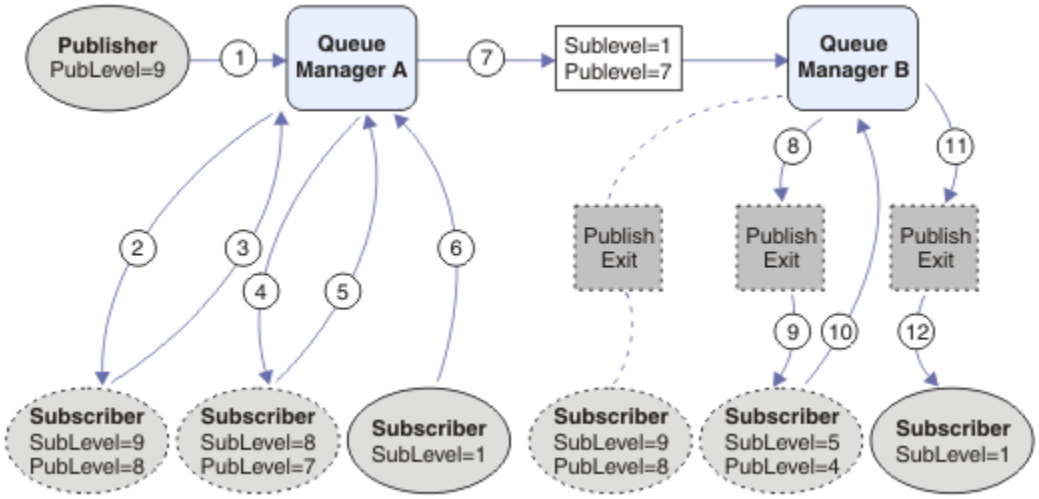
Należy dokładnie rozważyć miejsce wdrożenia przechwytywanych subskrybentów i wyjść publikowania w rozproszonej topologii publikowania/subskrypcji. Wzorzec prosty wdraża przechwytywanie subskrybentów do tego samego menedżera kolejek co publikatory, a publikowanie kończy się w tych samych menedżerach kolejek, co w przypadku końcowych subskrybentów.

Anty-wzorzec

Rysunek 67 na stronie 326 pokazuje, w jaki sposób można wykonywać awry, jeśli nie jest zgodny z prostym wzorcem. Aby skomplikować wdrożenie, do menedżera kolejek A dodawany jest ostatni subskrybent, a do menedżera kolejek B dodawane są dwa dodatkowe subskrybenty przechwytujące.

Publikacja jest przekazywana do menedżera kolejek B na poziomie `PubLevel` 7, gdzie jest przechwytywana przez subskrybenta na poziomie `SubLevel` 5 przed ich zużyciu przez końcowego subskrybenta na poziomie `SubLevel` 1. Wyjście publikowania przechwytuje publikację, zanim zostanie przekazana zarówno do przechwytującego konsumenta, jak i do konsumenta końcowego w menedżerze kolejek B. Publikacja dociera do końcowego subskrybenta menedżera kolejek A bez przetwarzania przez wyjście publikowania.

W topologii publikowania/subskrypcji subskrybenci proxy subskrybują element `SubLevel` 1i przekazują wartości `PubLevel` ustawione przez ostatni przechwytywacz subskrybenta. W produkcie Rysunek 67 na stronie 326wynik jest taki, że subskrybent nie przechwycił publikacji przy użyciu elementu `SubLevel` 9 w menedżerze kolejek B.



Rysunek 67. Złożone wdrożenie przechwytywanych subskrybentów

Opcje publikowania

Dostępnych jest kilka opcji, które sterują sposobem publikowania komunikatów.

Wstrzymać odpowiedź-do informacji od subskrybentów

Jeśli nie chcesz, aby subskrybenci mieli możliwość odpowiadania na otrzymane publikacje, możliwe jest wstrzymanie informacji w polach ReplyToQ i ReplyToQmgr w strukturze MQMD za pomocą opcji put-message w tabeli MQPMO_SUPPRESS_REPLYTO. Jeśli ta opcja jest używana, menedżer kolejek usuwa te informacje z deskryptora MQMD, gdy otrzymuje on publikację przed przekazaniem go do wszystkich subskrybentów.

Tej opcji nie można używać w połączeniu z opcją raportu, która wymaga kolejki ReplyTo, jeśli próba ta jest podejmowana z powodu niepowodzenia wywołania MQRC_MISSING_REPLY_TO_Q.

Poziom publikacji

Korzystanie z poziomów publikacji jest sposobem na kontrolowanie, którzy subskrybenci otrzymują publikację. Poziom publikacji oznacza poziom subskrypcji, do którego odnosi się publikacja. Publikacja otrzyma tylko subskrypcje o najwyższym poziomie subskrypcji, mniejsze lub równe poziomowi publikacji publikacji. Wartość ta musi być z zakresu od 0 do 9; zero oznacza najniższy poziom publikacji. Wartością początkową tego pola jest 9. Jednym z zastosowań poziomów publikacji i subskrypcji jest przechwytywanie publikacji.

Sprawdzanie, czy publikacja nie została dostarczona do żadnych subskrybentów

Aby sprawdzić, czy publikacja nie została dostarczona do żadnych subskrybentów, należy użyć opcji put-message komendy MQPMO_WARN_IF_NO_SUBS_MATCHED z wywołaniem MQPUT. Jeśli kod zakończenia komendy MQCC_WARNING i kod przyczyny MQRC_NO_SUBS_MATCHED zostaną zwrócone przez operację put, publikacja nie została dostarczona do żadnych subskrypcji. Jeśli w operacji put została określona opcja MQPMO_RETAIN, komunikat jest zachowywany i dostarczany do dowolnej późniejszej definicji zgodnej subskrypcji. W rozproszonym systemie publikowania/subskrybowania kod przyczyny MQRC_NO_SUBS_MATCHED jest zwracany tylko wtedy, gdy dla tematu w menedżerze kolejek nie zarejestrowano żadnych subskrypcji proxy.

Opcje subskrypcji

Dostępnych jest kilka opcji, które sterują sposobem obsługi subskrypcji komunikatów.

Trwałość komunikatu

Menedżery kolejek zachowują trwałość publikacji, które przesyłają do subskrybentów, zgodnie z ustawionym przez publikatora. Publikator ustawia trwałość, aby była jedną z następujących opcji:

0
Nietrwałe

1
Trwałe

2
Trwałość jako definicja kolejki/tematu

W przypadku publikowania/subskrypcji publikator rozstrzyża obiekt tematu i obiekt **topicString** na rozstrzygnięty obiekt tematu. Jeśli publikator określa trwałość jako definicję kolejki/tematu, to domyślna trwałość z rozstrzygniętego obiektu tematu jest ustawiana na potrzeby publikacji.

Zachowane publikacje

Aby kontrolować czas odbierania zachowanych publikacji, subskrybenci mogą korzystać z dwóch opcji subskrypcji:

Publikuj tylko na żądanie, MQSO_PUBLICATIONS_ON_REQUEST

Jeśli subskrybent ma mieć kontrolę nad publikacjami, można skorzystać z opcji subskrypcji MQSO_PUBLICATIONS_ON_REQUEST. Subskrybent może sterować tym, kiedy otrzymuje publikacje za pomocą wywołania MQSUBRQ (określając uchwyt Hsub, który został zwrócony z oryginalnego wywołania MQSUB), aby zażądać, aby został wysłany zachowaną publikację tematu. Subskrybenci korzystający z opcji subskrypcji MQSO_PUBLICATIONS_ON_REQUEST nie otrzymują żadnych niezachowanych publikacji.

Jeśli zostanie określona wartość MQSO_PUBLICATIONS_ON_REQUEST, należy użyć komendy MQSUBRQ w celu pobrania dowolnej publikacji. Jeśli nie zostanie użyta wartość MQSO_PUBLICATIONS_ON_REQUEST, zostaną wyświetlone komunikaty w postaci, w której są one publikowane.

Jeśli subskrybent korzysta z wywołania MQSUBRQ i korzysta ze znaków wieloznacznych w temacie subskrypcji, subskrypcja może być zgodna z wieloma tematami lub węzłami w drzewie tematów, a wszystkie z zachowanych komunikatów (o ile istnieją) zostaną wysłane do subskrybenta.

Ta opcja może być szczególnie przydatna w przypadku użycia z trwałymi subskrypcjami, ponieważ menedżer kolejek będzie nadal wysyłać publikacje do subskrybenta, jeśli subskrybent jest subskrybowany nawet wtedy, gdy ta aplikacja subskrybenta nie jest uruchomiona. Może to prowadzić do budowania komunikatów w kolejce subskrybenta. Tej kompilacji można uniknąć, jeśli subskrybent zarejestruje się za pomocą opcji MQSO_PUBLICATIONS_ON_REQUEST. Alternatywnie można użyć nietrwałych subskrypcji, jeśli jest to właściwe dla aplikacji, aby uniknąć tworzenia niechcianych komunikatów.

Jeśli subskrypcja jest trwała, a publikator korzysta z zachowanych publikacji, aplikacja subskrybenta może użyć wywołania MQSUBRQ w celu odświeżenia informacji o stanie po restarcie. Subskrybent musi następnie okresowo odświeżać swój stan przy użyciu wywołania MQSUBRQ.

Żadne publikacje nie będą wysyłane w wyniku wywołania MQSUB za pomocą tej opcji. Trwała subskrypcja, która została wznowiona po rozłączonym połączeniu, będzie używała opcji MQSO_PUBLICATIONS_ON_REQUEST, jeśli pierwotna subskrypcja została skonfigurowana w taki sposób, aby korzystała z tej opcji.

Tylko nowe publikacje, MQSO_NEW_PUBLICATIONS_ONLY

Jeśli zachowana publikacja istnieje w temacie, wszyscy subskrybenci, którzy dokonają subskrypcji po publikacji, otrzymają kopię tej publikacji. Subskrybent może korzystać z opcji subskrypcji MQSO_NEW_PUBLICATIONS_ONLY, jeśli subskrybent nie chce otrzymywać żadnych publikacji, które zostały wykonane wcześniej niż subskrypcja.

Grupowanie subskrypcji

Subskrypcje grupujące należy rozważyć, jeśli kolejka została ustawiona w celu odbierania publikacji i istnieje pewna liczba nakładających się na siebie subskrypcji, które publikują publikacje w tej samej kolejce. Ta sytuacja jest podobna do sytuacji w sekcji Overlapping subskrypcji.

Aby uniknąć otrzymywania zduplikowanych publikacji, należy ustawić opcję MQSO_GROUP_SUB w momencie subskrybowania tematu. W wyniku tego, gdy więcej niż jedna subskrypcja w grupie jest zgodna z tematem publikacji, tylko jedna subskrypcja jest odpowiedzialna za umieszczenie publikacji w kolejce. Pozostałe subskrypcje, które są zgodne z tematem publikacji, są ignorowane.

Subskrypcja odpowiedzialna za umieszczenie publikacji w kolejce jest wybierana na podstawie tego, że ma najdłuższy zgodny łańcuch tematu przed napotkaniem wszystkich znaków wieloznacznych. Może być on pomyślany jako najbliższa zgodna subskrypcja. Jego właściwości są propagowane do publikacji, w tym, bez względu na to, czy ma właściwość MQSO_NOT_OWN_PUBS . Jeśli tak, żadna publikacja nie zostanie dostarczona do kolejki, mimo że inne zgodne subskrypcje mogą nie mieć właściwości MQSO_NOT_OWN_PUBS .

Nie można umieścić wszystkich subskrypcji w jednej grupie w celu wyeliminowania zduplikowanych publikacji. Pogrupowane subskrypcje muszą spełniać następujące warunki:

1. Żadna z subskrypcji nie jest zarządzana.
2. Grupa subskrypcji dostarcza publikacje do tej samej kolejki.
3. Każda subskrypcja musi być na tym samym poziomie subskrypcji.
4. Komunikat publikacji dla każdej subskrypcji w grupie ma taki sam identyfikator korelacji.

   Aby upewnić się, że każda subskrypcja powoduje utworzenie komunikatu publikacji o tym samym identyfikatorze korelacji, należy ustawić wartość MQSO_SET_CORREL_ID , aby utworzyć własny identyfikator korelacji w publikacji, a następnie ustawić tę samą wartość w polu **SubCorrelId** w każdej subskrypcji. Nie należy ustawiać parametru **SubCorrelId** na wartość MQCI_NONE.

Więcej informacji na ten temat zawiera sekcja ../com.ibm.mq.ref.dev.doc/q100080_.dita#q100080_/mqso_group_sub .

Sprawdzanie i ustawianie atrybutów obiektu

Atrybuty to właściwości, które definiują parametry obiektu WebSphere MQ .

Wpływają one na sposób, w jaki menedżer kolejek przetwarza obiekt. Atrybuty każdego typu obiektu WebSphere MQ są opisane szczegółowo w sekcji Atrybuty obiektów.

Niektóre atrybuty są ustawiane podczas definiowania obiektu i mogą być zmieniane tylko za pomocą komend produktu WebSphere MQ . Przykładem takiego atrybutu jest domyślny priorytet komunikatów umieszczanych w kolejce. Działanie menedżera kolejek ma wpływ na inne atrybuty, które mogą zmieniać się w czasie. Przykładem może być bieżąca głębokość kolejki.

Za pomocą wywołania MQINQ można zapytać o bieżące wartości większości atrybutów. Interfejs MQI udostępnia także wywołanie MQSET, z którym można zmienić niektóre atrybuty kolejki. Nie można używać wywołań MQI w celu zmiany atrybutów dowolnego innego typu obiektu. Zamiast tego należy użyć następujących elementów:

Dla platformy WebSphere MQ dla platform Windows, UNIX i Linux

Narzędzie MQSC, opisane w sekcji MQSC reference(Skorowidz MQSC).

Uwaga: Nazwy atrybutów obiektów są wyświetlane w tej dokumentacji w postaci, w której są używane z wywołaniami MQINQ i MQSET. W przypadku korzystania z komend produktu WebSphere MQ w celu zdefiniowania, zmiany lub wyświetlenia atrybutów należy zidentyfikować atrybuty za pomocą słów kluczowych przedstawionych w opisach komend w odsyłaczkach do tematów.

Zarówno wywołania MQINQ, jak i MQSET używają tablic selektorów do identyfikowaniatych atrybutów, które mają zostać zapytane lub ustawione. Dla każdego atrybutu, z którym można pracować, istnieje selektor. Nazwa selektora ma przedrostek określony przez rodzaj atrybutu:

MQCA_	Te selektory odwołują się do atrybutów zawierających dane znakowe (na przykład nazwę kolejki).
MQIA_	Te selektory odwołują się do atrybutów, które zawierają wartości liczbowe (takie jak *CurrentQueueDepth*, liczba komunikatów w kolejce) lub wartość stała (na przykład *SyncPoint*, niezależnie od tego, czy menedżer kolejek obsługuje punkty synchronizacji).

Przed użyciem wywołania MQINQ lub MQSET należy połączyć aplikację z menedżerem kolejek i użyć wywołania MQOPEN, aby otworzyć obiekt do ustawienia lub zapytania o atrybuty. Operacje te są opisane w produktach "Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego" na stronie 211 i "Otwieranie i zamykanie obiektów" na stronie 220.

Użyj poniższych odsyłaczy, aby dowiedzieć się więcej na temat uzyskiwania informacji na temat atrybutów obiektu i ustawiania atrybutów obiektu:

- "Uzyskiwanie informacji o atrybutach obiektu" na stronie 329
- "Niektóre przypadki, w których wywołanie MQINQ nie powiodło się" na stronie 330
- "Ustawianie atrybutów kolejki" na stronie 331

Pojęcia pokrewne

"Interfejs kolejki komunikatów-przegląd" na stronie 199
Informacje na temat komponentów interfejsu kolejek komunikatów (Message Queue Interface-MQI).

"Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego" na stronie 211
Aby można było korzystać z usług programistycznych produktu WebSphere MQ , program musi mieć połączenie z menedżerem kolejek. Informacje zawarte w tej sekcji umożliwiają zapoznanie się z informacjami na temat nawiązywania połączenia z menedżerem kolejek i z jego rozłączeniem.

"Otwieranie i zamykanie obiektów" na stronie 220
Te informacje udostępniają wgląd w otwieranie i zamykanie obiektów WebSphere MQ .

"Umieszczanie komunikatów w kolejce" na stronie 231
Te informacje umożliwiają zapoznanie się z informacjami na temat umieszczania komunikatów w kolejce.

"Pobieranie komunikatów z kolejki" na stronie 246
Ta sekcja zawiera informacje na temat pobierania komunikatów z kolejki.

"Zatwierdzanie i wycofywanie jednostek pracy" na stronie 331
W tej sekcji opisano sposób zatwierdzania i tworzenia kopii zapasowych wszystkich możliwych do odtworzenia operacji get i put, które wystąpiły w jednostce pracy.

"Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy" na stronie 338
Informacje na temat wyzwalaczy i sposobów uruchamiania aplikacji IBM WebSphere MQ przy użyciu wyzwalaczy.

"Praca z interfejsem MQI i klastrami" na stronie 355
Istnieją specjalne opcje dotyczące wywołań i kodów powrotu, które odnoszą się do technologii klastrowej.

Uzyskiwanie informacji o atrybutach obiektu

Użyj wywołania MQINQ, aby dowiedzieć się więcej o atrybutach dowolnego typu produktu IBM WebSphere MQ.

Jako dane wejściowe dla tego wywołania należy podać:

- Uchwyt połączenia.
- Uchwyt obiektu.
- Liczba selektorów.
- Tablica selektorów atrybutów, każdy selektor ma postać MQCA_ * lub MQIA_ *. Każdy selektor reprezentuje atrybut o wartości, o którą użytkownik chce się dowiedzieć, a każdy selektor musi być poprawny dla typu obiektu reprezentowanego przez uchwyt obiektu. Selektory można określać w dowolnej kolejności.

- Liczba atrybutów całkowitoliczbowych, o których pytasz. Podaj wartość zero, jeśli nie jesteś pytany o atrybuty całkowitoliczbowe.
- Długość buforu atrybutów znakowych w programie *CharAttrLength*. Musi to być co najmniej suma długości wymaganych do przechowywania każdego łańcucha atrybutu znakowego. Wartość zero oznacza, że nie jest pytany o atrybuty znaków.

Dane wyjściowe komendy MQINQ są następujące:

- Zestaw wartości atrybutów całkowitoliczbowych skopiowanych do tablicy. Liczba wartości jest określana przez produkt *IntAttrCount*. Jeśli parametr *IntAttrCount* lub *SelectorCount* ma wartość zero, ten parametr nie jest używany.
- Bufor, w którym zwracane są atrybuty znakowe. Długość buforu jest nadawana przez parametr *CharAttrLength* . Jeśli parametr *CharAttrLength* lub *SelectorCount* ma wartość zero, ten parametr nie jest używany.
- Kod zakończenia. Jeśli kod zakończenia generuje ostrzeżenie, oznacza to, że wywołanie zostało zakończone tylko częściowo. W takim przypadku należy sprawdzić kod przyczyny.
- Kod przyczyny. Istnieją trzy sytuacje częściowego zakończenia:
  - Selektor nie ma zastosowania do typu kolejki
  - Brak wystarczającej ilości miejsca dla atrybutów całkowitych.
  - Brak wystarczającej ilości miejsca dla atrybutów znakowych

  Jeśli wystąpi więcej niż jedna z tych sytuacji, zwracana jest pierwsza z nich.

Jeśli otwarto kolejkę dla danych wyjściowych lub zapytania i zostanie ona rozstrzygana w nielokalnej kolejce klastra, można jedynie uzyskać informacje o nazwie kolejki, typie kolejki i wspólnych atrybutach. Jeśli użyto komendy MQOO_BIND_ON_OPEN, wartości wspólnych atrybutów są wartościami z wybranej kolejki. Wartości te należą do dowolnej z możliwych kolejek klastra, jeśli użyto wartości MQOO_BIND_NOT_FIXED lub MQOO_BIND_ON_GROUP, albo użyto komendy MQOO_BIND_AS_Q_DEF, a atrybut kolejki produktu *DefBind* miał wartość MQBND_BIND_NOT_FIXED. Więcej informacji na ten temat można znaleźć w sekcji “MQOPEN i klastry” na stronie 356 i MQOPEN .

Uwaga: Wartości zwracane przez wywołanie są obrazem stanu wybranych atrybutów. Atrybuty mogą się zmieniać, zanim program będzie działać na zwróconych wartościach.

W tabeli MQINQznajduje się opis wywołania MQINQ.

Niektóre przypadki, w których wywołanie MQINQ nie powiodło się

Jeśli alias zostanie otwarty w celu sprawdzenia jego atrybutów, zwracane są atrybuty kolejki aliasowej (obiekt WebSphere MQ używany do uzyskania dostępu do innej kolejki), a nie atrybuty kolejki podstawowej.

Jednak definicja kolejki podstawowej, do której jest rozstrzygany alias, jest również otwierana przez menedżer kolejek, a jeśli inny program zmieni użycie kolejki podstawowej w przedziale czasu między wywołaniami MQOPEN i MQINQ, wywołanie MQINQ nie powiedzie się i zwróci kod przyczyny MQRC_OBJECT_CHANGED. Wywołanie nie powiedzie się również wtedy, gdy atrybuty obiektu kolejki aliasowej zostaną zmienione.

Podobnie w przypadku otwarcia kolejki zdalnej w celu sprawdzenia jej atrybutów zwracane są tylko atrybuty lokalnej definicji kolejki zdalnej.

W przypadku określenia jednego lub większej liczby selektorów, które nie są poprawne dla typu atrybutów kolejki, dla których jest wykonywane zapytanie, wywołanie MQINQ kończy się z ostrzeżeniem i ustawia dane wyjściowe w następujący sposób:

- W przypadku atrybutów będących liczbami całkowitymi, odpowiadające im elementy *IntAttrs* są ustawione na wartość MQIAV_NOT_APPLICABLE.
- W przypadku atrybutów znakowych odpowiednie części łańcucha *CharAttrs* są ustawiane na gwiazdki.

Jeśli zostanie określony jeden lub większa liczba selektorów, które nie są poprawne dla typu atrybutów obiektu, dla których zostanie zapytany, wywołanie MQINQ nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_SELECTOR_ERROR.

Nie można wywołać funkcji MQINQ w celu wyszukania kolejki modelowej. W tym celu należy użyć narzędzia MQSC lub komend dostępnych na platformie.

Ustawianie atrybutów kolejki

Informacje zawarte w tej sekcji umożliwiają poznanie sposobu ustawiania atrybutów kolejki przy użyciu wywołania MQSET.

Za pomocą wywołania MQSET można ustawić tylko następujące atrybuty kolejki:

- *InhibitGet* (ale nie dla kolejek zdalnych)
- *DistList* (nie w systemie z/OS)
- *InhibitPut*
- *TriggerControl*
- *TriggerType*
- *TriggerDepth*
- *TriggerMsgPriority*
- *TriggerData*

Wywołanie MQSET ma takie same parametry jak wywołanie MQINQ. Jednak dla tabeli MQSET wszystkie parametry z wyjątkiem kodu zakończenia i kodu przyczyny są parametrami wejściowymi. Brak sytuacji częściowych.

Uwaga: Nie można użyć interfejsu MQI w celu ustawienia atrybutów obiektów WebSphere MQ innych niż kolejki zdefiniowane lokalnie.

Więcej informacji na temat wywołania MQSET zawiera sekcja MQSET.

Zatwierdzanie i wycofywanie jednostek pracy

W tej sekcji opisano sposób zatwierdzania i tworzenia kopii zapasowych wszystkich możliwych do odtworzenia operacji get i put, które wystąpiły w jednostce pracy.

W tym temacie są używane następujące terminy:

- Zatwierdzanie
- Wycofanie
- Koordynacja punktu synchronizacji
- Punkt synchronizacji
- Jednostka pracy
- zatwierdzanie jednofazowe
- zatwierdzania dwufazowe.

Użytkownik zaznajomiony z tymi terminami przetwarzania transakcji może przejść do programu “Uwagi dotyczące punktu synchronizacji w aplikacjach produktu IBM WebSphere MQ” na stronie 333.

Zatwierdzanie i wycofanie

Gdy program umieszcza komunikat w kolejce w obrębie jednostki pracy, ten komunikat jest widoczny dla innych programów tylko wtedy, gdy program zatwierdza jednostkę pracy. Aby zatwierdzić jednostkę pracy, wszystkie aktualizacje muszą być pomyślne, aby zachować integralność danych. Jeśli program wykryje błąd i zdecyduje, że operacja put nie jest trwała, może wycofać się z jednostki pracy. Gdy program wykonuje wycofany program, program IBM WebSphere MQ odtwarza kolejkę, usuwając komunikaty umieszczone w kolejce przez tę jednostkę pracy. Sposób, w jaki program wykonuje operacje zatwierdzania i tworzenia kopii zapasowych, zależy od środowiska, w którym działa program.

Podobnie, gdy program pobiera komunikat z kolejki w obrębie jednostki pracy, komunikat ten pozostaje w kolejce do momentu zatwierdzenia przez program jednostki pracy, ale komunikat nie jest dostępny do pobrania przez inne programy. Komunikat zostaje trwale usunięty z kolejki, gdy program zatwierdza jednostkę pracy. Jeśli program tworzy kopię zapasową jednostki pracy, program IBM WebSphere MQ odtwarza kolejkę, udostępniając komunikaty do pobrania przez inne programy.

Koordynacja punktu synchronizacji, punkt synchronizacji, jednostka pracy
Koordynacja punktu synchronizacji to proces, za pomocą którego jednostki pracy są zatwierdzane lub wycofane z integralności danych.

Decyzja o zatwierdzeniu lub wycofaniu zmian jest podejmowana, w najprostszym przypadku, na końcu transakcji. Może być jednak bardziej przydatne dla aplikacji w celu zsynchronizowania zmian danych w innych punktach logicznych w ramach transakcji. Te punkty logiczne są nazywane *punktami synchronizacji* (lub *punktami synchronizacji*), a okres przetwarzania zestawu aktualizacji między dwoma punktami synchronizacji jest nazywany *jednostką pracy*. Kilka wywołań MQGET i wywołań MQPUT może być częścią pojedynczej jednostki pracy. Maksymalną liczbą komunikatów w jednostce pracy można sterować za pomocą atrybutu MAXUMSGS komendy ALTER QMGR na innych platformach, z wyjątkiem systemu z/OS. Szczegółowe informacje na temat tych komend można znaleźć w Informacje dodatkowe dotyczące komend MQSC *Skorowidz Komend Skryptowych WebSphere MQ Script (MQSC)* .

zatwierdzanie jednofazowe
Proces *zatwierdzania jednofazowego* to proces, w którym program może zatwierdzać aktualizacje w kolejce bez koordynacji jej zmian z innymi menedżerami zasobów.

zatwierdzania dwufazowe.
Proces *zatwierdzania dwufazowego* to proces, w którym aktualizacje, które program udostępnił do kolejek produktu IBM WebSphere MQ , mogą być koordynowane z aktualizacjami innych zasobów (na przykład baz danych pod kontrolą DB2). W ramach takiego procesu aktualizacje wszystkich zasobów są zatwierdzane lub wycofane razem.

Aby ułatwić obsługę jednostek pracy, produkt IBM WebSphere MQ udostępnia atrybut *BackoutCount* . Wartość ta jest zwiększana za każdym razem, gdy zostanie utworzona kopia zapasowa komunikatu w ramach jednostki pracy. Jeśli komunikat wielokrotnie powoduje nieprawidłowe zakończenie działania jednostki pracy, wartość *BackoutCount* w końcu przekracza wartość *BackoutThreshold*. Ta wartość jest ustawiana, gdy kolejka jest zdefiniowana. W takiej sytuacji aplikacja może usunąć komunikat z jednostki pracy i umieścić ją w innej kolejce, zgodnie z definicją w sekcji *BackoutRequeueQName*. Gdy komunikat jest przenoszony, jednostka pracy może zatwierdzić.

Aby dowiedzieć się więcej na temat zatwierdzania i tworzenia kopii zapasowych jednostek pracy, należy użyć następujących odsyłaczy:

- “Uwagi dotyczące punktu synchronizacji w aplikacjach produktu IBM WebSphere MQ” na stronie 333
- “Punkty synchronizacji w produkcie IBM WebSphere MQ w systemach UNIX, Linux, and Windows” na stronie 334

Pojęcia pokrewne
“Interfejs kolejki komunikatów-przegląd” na stronie 199
Informacje na temat komponentów interfejsu kolejek komunikatów (Message Queue Interface-MQI).

“Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego” na stronie 211
Aby można było korzystać z usług programistycznych produktu WebSphere MQ , program musi mieć połączenie z menedżerem kolejek. Informacje zawarte w tej sekcji umożliwiają zapoznanie się z informacjami na temat nawiązywania połączenia z menedżerem kolejek i z jego rozłączeniem.

“Otwieranie i zamykanie obiektów” na stronie 220
Te informacje udostępniają wgląd w otwieranie i zamykanie obiektów WebSphere MQ .

“Umieszczanie komunikatów w kolejce” na stronie 231
Te informacje umożliwiają zapoznanie się z informacjami na temat umieszczania komunikatów w kolejce.

“Pobieranie komunikatów z kolejki” na stronie 246
Ta sekcja zawiera informacje na temat pobierania komunikatów z kolejki.

“Sprawdzanie i ustawianie atrybutów obiektu” na stronie 328
Atrybuty to właściwości, które definiują parametry obiektu WebSphere MQ .

“Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy” na stronie 338
Informacje na temat wyzwalaczy i sposobów uruchamiania aplikacji IBM WebSphere MQ przy użyciu wyzwalaczy.

“Praca z interfejsem MQI i klastrami” na stronie 355
Istnieją specjalne opcje dotyczące wywołań i kodów powrotu, które odnoszą się do technologii klastrowej.

Uwagi dotyczące punktu synchronizacji w aplikacjach produktu IBM WebSphere MQ

Ta sekcja zawiera informacje na temat używania punktów synchronizacji w aplikacjach produktu IBM WebSphere MQ .

Zatwierdzanie dwufazowe jest obsługiwane w następujących warunkach:

- WebSphere MQ for AIX
- WebSphere MQ for HP-UX
- WebSphere MQ for Linux
- WebSphere MQ dla systemu Solaris
- WebSphere MQ dla systemu Windows
- CICS for MVS/ESA 4.1
- CICS Transaction Server for z/OS
- TXSeries
- IMS/ESA
- Inni koordynatorzy zewnętrzni korzystający z interfejsu XA X/Open

Zatwierdzanie jednofazowe jest obsługiwane w następujących warunkach:

- Produkt WebSphere MQ w systemach UNIX
- WebSphere MQ dla systemu Windows

Uwaga: Szczegółowe informacje na temat interfejsów zewnętrznych zawiera sekcja “Interfejsy do zewnętrznych menedżerów punktów synchronizacji” na stronie 336, a dokumentacja XA *CAE Specification Distributed Transaction Processing: The XA Specification*(Specyfikacja rozproszonego przetwarzania transakcji CAE: Specyfikacja XA) opublikowana przez grupę Open Group. Menedżery transakcji (takie jak CICS, IMS, Encinai Tuxedo) mogą uczestniczyć w zatwierdzaniu dwufazowym, które są koordynowane z innymi zasobami odtwarzalnymi. Oznacza to, że funkcje kolejkowania udostępniane przez produkt WebSphere MQ mogą być wprowadzane w zasięgu jednostki pracy, zarządzanej przez menedżera transakcji.

Przykłady dostarczane z produktem WebSphere MQ pokazują produkt WebSphere MQ koordynujący bazy danych zgodne z protokołem XA. Więcej informacji na temat tych przykładów można znaleźć w sekcji “Przykładowe programy dla platform rozproszonych” na stronie 99.

W aplikacji WebSphere MQ można określić każdy element wstawiany i uzyskać informacje o tym, czy wywołanie ma być pod kontrolą punktu synchronizacji. Aby operacja put była wykonywana w ramach elementu sterującego syncpoint, należy użyć wartości MQPMO_SYNCPOINT w polu *Options* struktury MQPMO podczas wywoływania komendy MQPUT. W przypadku operacji pobierania należy użyć wartości MQGMO_SYNCPOINT w polu *Options* struktury MQGMO. Jeśli użytkownik nie wybierze jawnie opcji, działanie domyślne zależy od platformy. Domyślny element sterujący syncpoint nie jest taki sam.

Gdy wywołanie MQPUT1 jest wysyłane z MQPMO_SYNCPOINT, domyślne zachowanie zmienia się tak, że operacja put jest wykonywana asynchronicznie. Może to spowodować zmianę w zachowaniu niektórych aplikacji, które polegają na zwróconych określonych polach w strukturach MQOD i MQMD, ale które teraz zawierają niezdefiniowane wartości. Aplikacja może określić MQPMO_SYNC_RESPONSE, aby upewnić się, że operacja put jest wykonywana synchronicznie i że wszystkie odpowiednie wartości pól są zakończone.

Jeśli aplikacja otrzymuje kod przyczyny MQRC_BACKED_OUT w odpowiedzi na operację MQPUT lub MQGET w punkcie synchronizacji, aplikacja powinna normalnie wycofać bieżącą transakcję za pomocą MQBACK, a następnie, w razie potrzeby, ponowić próbę wykonania całej transakcji. Jeśli aplikacja odbierze wywołanie MQRC_BACKED_OUT w odpowiedzi na wywołanie MQCMIT lub MQDISC, nie ma potrzeby wywoływania komendy MQBACK.

Za każdym razem, gdy tworzona jest wycofana kopia zapasowa wywołania MQGET, pole *BackoutCount* struktury MQMD komunikatu, którego dotyczy to komunikat, jest zwiększane. Wysoka wartość *BackoutCount* wskazuje komunikat, który został wielokrotnie wycofany. Może to wskazywać na problem z tym komunikatem, który powinien zostać zbadany. Szczegółowe informacje na temat *BackoutCount*zawiera sekcja BackoutCount .

Jeśli program wysyła wywołanie MQDISC w czasie, gdy istnieją niezatwierdzone żądania, niejawny punkt synchronizacji ma miejsce. Jeśli program zakończy działanie w sposób nieprawidłowy, wystąpi niejawne wyjście.

Zmiany atrybutów kolejki (przez wywołanie MQSET lub przez komendy) nie są naruszane przez zatwierdzanie lub wycofywanie jednostek pracy.

Punkty synchronizacji w produkcie IBM WebSphere MQ w systemach UNIX, Linux, and Windows

Obsługa punktu synchronizacji jest wykonywana na dwóch typach jednostek pracy: lokalnym i globalnym.

lokalna jednostka pracy to jedna, w której jedynymi aktualizowanymi zasobami są te, które są aktualizowane przez menedżera kolejek produktu WebSphere MQ . W tym przypadku koordynacja punktu synchronizacji jest udostępniana przez sam menedżer kolejek przy użyciu procedury zatwierdzania jednofazowego.

Globalne jednostki pracy to jedna z nich, w której aktualizowane są również zasoby należące do innych menedżerów zasobów, takich jak bazy danych. Produkt WebSphere MQ może koordynować takie jednostki pracy. Mogą być one również koordynowane przez zewnętrzny kontroler transakcji, taki jak inny menedżer transakcji lub kontroler transakcji IBM i .

W celu zapewnienia pełnej integralności należy użyć procedury zatwierdzania dwufazowego. Two-phase commit can be provided by XA-compliant transaction managers and databases such as IBM's TXSeries and UDB. Produkty WebSphere MQ (z wyjątkiem produktów WebSphere MQ for IBM i i WebSphere MQ for z/OS) mogą koordynować globalne jednostki pracy przy użyciu procesu zatwierdzania dwufazowego.

Lokalne jednostki pracy

Jednostki pracy, które dotyczą tylko menedżera kolejek, są nazywane *lokalnymi* jednostkami pracy. Koordynowanie punktu synchronizacji jest udostępniane przez sam menedżer kolejek (koordynacja wewnętrzna) przy użyciu procesu zatwierdzania jednofazowego.

Aby uruchomić lokalną jednostkę pracy, aplikacja generuje żądania MQGET, MQPUT lub MQPUT1 , określając odpowiednią opcję punktu synchronizacji. Jednostka pracy jest zatwierdzana za pomocą komendy MQCMIT lub wycofana przy użyciu komendy MQBACK. Jednak jednostka pracy kończy się również wtedy, gdy połączenie między aplikacją a menedżerem kolejek jest zerwane, umyślnie lub niezamierzone.

Jeśli aplikacja rozłącza się (MQDISC) z menedżera kolejek, podczas gdy globalna jednostka pracy koordynowana przez produkt WebSphere MQ jest nadal aktywna, podejmowana jest próba zatwierdzenia jednostki pracy. Jeśli jednak aplikacja kończy działanie bez rozłączania, jednostka pracy zostanie wycofana, ponieważ aplikacja zostanie uznana za zakończonej nieprawidłowo.

Globalne jednostki pracy

Globalne jednostki pracy należy używać wtedy, gdy konieczne jest uwzględnienie aktualizacji zasobów należących do innych menedżerów zasobów.

W tym przypadku koordynacja może być wewnętrzna lub zewnętrzna względem menedżera kolejek:

Wewnętrzna koordynacja punktu synchronizacji

Koordynacja menedżera kolejek globalnych jednostek pracy nie jest obsługiwana przez produkt WebSphere MQ for IBM i lub WebSphere MQ for z/OS. Nie jest on obsługiwany w środowisku klienckim MQI produktu WebSphere MQ.

W tym miejscu WebSphere MQ wykonuje koordynację. Aby uruchomić globalną jednostkę pracy, aplikacja wysyła wywołanie MQBEGIN.

Jako dane wejściowe dla wywołania MQBEGIN należy podać uchwyt połączenia (*Hconn*), który jest zwracany przez wywołanie MQCONN lub MQCONNX. Ten uchwyt reprezentuje połączenie z menedżerem kolejek produktu WebSphere MQ .

Aplikacja generuje żądania MQGET, MQPUT lub MQPUT1 , określając odpowiednią opcję punktu synchronizacji. Oznacza to, że można użyć komendy MQBEGIN w celu zainicjowania globalnej jednostki pracy, która aktualizuje zasoby lokalne, zasoby należące do innych menedżerów zasobów lub obie te wartości. Aktualizacje zasobów należących do innych menedżerów zasobów są dokonywane przy użyciu interfejsu API tego menedżera zasobów. Nie można jednak używać interfejsu MQI do aktualizowania kolejek należących do innych menedżerów kolejek. Wydaj komendę MQCMIT lub MQBACK przed uruchomieniem kolejnych jednostek pracy (lokalne lub globalne).

Globalna jednostka pracy jest zatwierdzana za pomocą komendy MQCMIT; inicjuje to dwufazowe zatwierdzanie wszystkich menedżerów zasobów zaangażowanych w jednostkę pracy. Proces zatwierdzania dwufazowego jest używany, w którym menedżerowie zasobów (na przykład menedżery baz danych zgodne z interfejsem XA, takie jak DB2, Oraclei Sybase), są najpierw poproszeni o przygotowanie do zatwierdzenia. Tylko, jeśli wszyscy są przygotowani są proszeni o zatwierdzenie. Jeśli dowolny menedżer zasobów sygnalizuje, że nie może zatwierdzić, to każdy z nich jest proszony o wyjście. Alternatywnie można użyć komendy MQBACK, aby wycofać zmiany wszystkich menedżerów zasobów.

Jeśli aplikacja rozłącza się (MQDISC), podczas gdy globalna jednostka pracy jest nadal aktywna, jednostka pracy jest zatwierdzana. Jeśli jednak aplikacja kończy działanie bez rozłączania, jednostka pracy zostanie wycofana, ponieważ aplikacja zostanie uznana za zakończonej nieprawidłowo.

Dane wyjściowe komendy MQBEGIN to kod zakończenia i kod przyczyny.

Jeśli do uruchomienia globalnej jednostki pracy używana jest opcja MQBEGIN, uwzględniane są wszystkie zewnętrzne menedżery zasobów, które zostały skonfigurowane z menedżerem kolejek. Wywołanie uruchamia jednak jednostkę pracy, ale kończy się ostrzeżeniem, jeśli:

- Brak uczestniczących menedżerów zasobów (oznacza to, że żaden menedżer zasobów nie został skonfigurowany z menedżerem kolejek)

lub wersji

- Jeden lub większa liczba menedżerów zasobów nie jest dostępna.

W takich przypadkach jednostka pracy musi zawierać aktualizacje tylko tych menedżerów zasobów, które były dostępne w momencie uruchomienia jednostki pracy.

Jeśli jeden z menedżerów zasobów nie może zatwierdzić swoich aktualizacji, wszystkie menedżery zasobów są instruowane, aby wycofać ich aktualizacje, a program MQCMIT kończy działanie z ostrzeżeniem. W nietypowych okolicznościach (zwykle interwencja operatora) wywołanie MQCMIT może się nie powieść, jeśli niektórzy menedżerowie zasobów zatwierdzają swoje aktualizacje, ale inne je wycofają; praca jest uważana za zakończonej wynikiem *mieszanym* . Takie wystąpienia są diagnozowane w dzienniku błędów menedżera kolejek, dzięki czemu możliwe jest podjęcie działań zaradczych.

Operacja MQCMIT globalnej jednostki pracy powiedzie się, jeśli wszystkie osoby zarządzające zasobami zaangażują się w ich aktualizacje.

Opis wywołania MQBEGIN można znaleźć w sekcji MQBEGIN.

Zewnętrzna koordynacja punktu synchronizacji

Dzieje się tak wtedy, gdy wybrano koordynator punktu synchronizacji inny niż WebSphere MQ , na przykład CICS, Encinalub Tuxedo.

W takiej sytuacji produkt WebSphere MQ w systemach UNIX and Linux oraz produkt WebSphere MQ for Windows rejestrują swoje zainteresowania w wyniku jednostki pracy z koordynatorem punktu synchronizacji, dzięki czemu mogą one zatwierdzać lub wycofać wszystkie niezatwierdzone operacje get lub put w zależności od potrzeb. Zewnętrzny koordynator punktu synchronizacji określa, czy są dostępne jedno-lub dwufazowe protokoły zobowiązań.

W przypadku korzystania z zewnętrznego koordynatora, MQCMIT, MQBACK i MQBEGIN nie mogą zostać wydane. Wywołania tych funkcji nie powiodą się z powodu kodu przyczyny MQRC_ENVIRONMENT_ERROR.

Sposób uruchamiania zewnętrznie koordynowanej jednostki pracy zależy od interfejsu programistycznego udostępnianego przez koordynatora punktu synchronizacji. Może być wymagane jawne wywołanie. Jeśli wymagane jest wywołanie jawne, a zostanie wydana wywołanie MQPUT z opcją MQPMO_SYNCPOINT, gdy jednostka pracy nie jest uruchomiona, zwracany jest kod zakończenia MQRC_SYNCPOINT_NOT_AVAILABLE.

Zakres jednostki pracy jest określany przez koordynatora punktu synchronizacji. Stan połączenia między aplikacją a menedżerem kolejek wpływa na powodzenie lub niepowodzenie wywołań MQI, które dotyczą aplikacji, a nie stanu jednostki pracy. Aplikacja może na przykład rozłączyć się i ponownie połączyć się z menedżerem kolejek podczas aktywnej jednostki pracy i wykonać dalsze operacje MQGET i MQPUT w tej samej jednostce pracy. Jest to znane jako oczekiwanie na rozłączenie.

W programach CICS można używać wywołań funkcji API produktu WebSphere MQ , niezależnie od tego, czy mają być używane możliwości interfejsu CICSXA. Jeśli interfejs XA nie jest używany, operacje umieszczania i pobierania komunikatów z kolejkami i z kolejek nie będą zarządzane w jednostkach roboczych CICS . Jedną z przyczyn wyboru tej metody jest to, że ogólna spójność jednostki pracy nie jest dla Ciebie istotna.

Jeśli integralność jednostek pracy jest dla Ciebie ważna, należy użyć interfejsu XA. Jeśli używany jest interfejs XA, program CICS korzysta z protokołu zatwierdzania dwufazowego w celu zapewnienia, że wszystkie zasoby w obrębie jednostki pracy są aktualizowane razem.

Więcej informacji na temat konfigurowania obsługi transakcyjnej zawiera sekcja “Scenariusze obsługi transakcyjnej” na stronie 42, a także dokumentacja TXSeries CICS , na przykład *TXSeries for Multiplatforms CICS Administration Guide for Open Systems*.

Interfejsy do zewnętrznych menedżerów punktów synchronizacji

WebSphere MQ on UNIX and Linux systems, and WebSphere MQ for Windows support coordination of transactions by external syncpoint managers that use the X/Open XA interface.

Niektóre menedżery transakcji XA (TXSeries) wymagają, aby każdy menedżer zasobów XA udostępniał swoją nazwę. Jest to łańcuch o nazwie name w strukturze przełącznika XA. Menedżer zasobów dla produktu WebSphere MQ w systemach UNIX, Linux i Windows nosi nazwę MQSeries_XA_RMI. Szczegółowe informacje na temat interfejsów XA można znaleźć w dokumentacji interfejsu XA *CAE Specification Distributed Transaction Processing: The XA Specification*(Specyfikacja rozproszonego przetwarzania transakcyjnego CAE: Specyfikacja XA) opublikowanej przez grupę Open Group.

W konfiguracji interfejsu XA produkt WebSphere MQ w systemach UNIX, Linuxi Windows spełnia rolę Resource Managerinterfejsu XA. Koordynator punktu synchronizacji XA może zarządzać zestawem menedżerów zasobów XA i synchronizować zatwierdzanie lub wycofanie transakcji w obu menedżerach zasobów. W ten sposób działa on dla statycznie zarejestrowanego menedżera zasobów:

1. Aplikacja powiadamia koordynatora punktu synchronizacji o tym, że chce uruchomić transakcję.
2. Koordynator punktu synchronizacji wysyła wywołanie do wszystkich menedżerów zasobów, o których wie, w celu powiadomienia ich o bieżącej transakcji.
3. Problemy aplikacji wywołują aktualizację zasobów zarządzanych przez menedżery zasobów powiązane z bieżącą transakcją.
4. Aplikacja żąda, aby koordynator punktu synchronizacji zatwierdził transakcję lub wycofał transakcję.

5. Koordynator punktu synchronizacji wywołuje każdy menedżer zasobów przy użyciu protokołów zatwierdzania dwufazowego w celu zakończenia transakcji zgodnie z żądaniem.

Specyfikacja interfejsu XA wymaga, aby każdy Resource Manager udostępniał strukturę o nazwie *Przełącznik XA*. Ta struktura deklaruje możliwości programu Resource Manageroraz funkcje, które mają być wywoływane przez koordynatora punktu synchronizacji.

Istnieją dwie wersje tej struktury:

MQRMIXASwitch	Zarządzanie statycznymi zasobami XA
MQRMIXASwitchDynamic	Dynamiczne zarządzanie zasobami XA

Listę bibliotek zawierających tę strukturę można znaleźć w sekcji "Struktura przełącznika IBM WebSphere MQ XA" na stronie 71.

Metoda, która musi być używana do łączenia ich z koordynatorem punktu synchronizacji XA, jest zdefiniowana przez koordynatora. Należy zapoznać się z dokumentacją udostępnionej przez tego koordynatora w celu określenia, w jaki sposób można włączyć WebSphere MQ do współpracy z koordynatorem punktu synchronizacji XA.

Struktura *xa_info* , która jest przekazywana do dowolnego wywołania *xa_open* przez koordynatora punktu synchronizacji, może być nazwą menedżera kolejek, który ma być administrowany. Ten sam formularz ma taką samą postać, jak nazwa menedżera kolejek przekazanego do tabeli MQCONN lub MQCONNX. Jeśli domyślny menedżer kolejek ma być używany, może on być pusty. Można jednak użyć dwóch dodatkowych parametrów TPM i AXLIB.

Program TPM umożliwia określenie nazwy menedżera transakcji w produkcie WebSphere MQ , na przykład CICS. AXLIB pozwala na określenie rzeczywistej nazwy biblioteki w menedżerze transakcji, w którym znajdują się punkty wejścia XA AX.

Jeśli używany jest dowolny z tych parametrów lub inny niż domyślny menedżer kolejek, należy określić nazwę menedżera kolejek przy użyciu parametru QMNAME. Więcej informacji na ten temat zawiera sekcja Parametry CHANNEL, TRPTYPE, CONNAME i QMNAME w łańcuchu xa_open.

Ograniczenia

1. Globalne jednostki pracy nie są dozwolone ze współużytkowanym serwerem Hconn (zgodnie z opisem w sekcji "Połączenia współużytkowane (niezależne od wątku) z produktem MQCONNX" na stronie 217).
2. W systemach Windows wszystkie funkcje zadeklarowane w przełączniku XA są deklarowane jako funkcje _cdecl.
3. Zewnętrzny koordynator punktu synchronizacji może w danym momencie administrować tylko jednym menedżerem kolejek. Jest to spowodowane tym, że koordynator ma efektywne połączenie z każdym menedżerem kolejek i w związku z tym podlega regułce, że tylko jedno połączenie jest dozwolone w danym momencie.

   Uwaga: Uwaga: Aplikacja kliencka JMS (aplikacja CLIENT JEE) działająca na serwerze JEE nie ma tego ograniczenia, tak więc pojedyncza transakcja zarządzana przez serwer JEE może koordynować wiele menedżerów kolejek w tej samej transakcji. Jednak aplikacja serwera JMS działająca w trybie powiązań nadal podlega regułce, z której dozwolony jest tylko jedno połączenie w danym momencie.
4. Wszystkie aplikacje, które są uruchamiane za pomocą koordynatora punktu synchronizacji, mogą łączyć się tylko z menedżerem kolejek, który jest administrowany przez koordynatora, ponieważ są one już efektywnie połączone z tym menedżerem kolejek. Muszą one wydać komendę MQCONN lub MQCONNX w celu uzyskania uchwytu połączenia i musi wydać komendę MQDISC przed wyjściem z systemu. Alternatywnie mogą one używać wyjścia UE014015 w przypadku programu TXSeries CICS.

Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy

Informacje na temat wyzwalaczy i sposobów uruchamiania aplikacji IBM WebSphere MQ przy użyciu wyzwalaczy.

Niektóre aplikacje produktu WebSphere MQ , które obsługują kolejki, działają w sposób ciągły, dlatego są zawsze dostępne do pobierania komunikatów, które docierają do kolejek. Być może jednak nie będzie to pożądane, gdy liczba komunikatów przychodzących do kolejek jest nieprzewidywalna. W takim przypadku aplikacje mogą pochłaniać zasoby systemowe nawet wtedy, gdy nie ma żadnych komunikatów do pobrania.

Produkt WebSphere MQ udostępnia narzędzie, które umożliwia automatyczne uruchomienie aplikacji, gdy dostępne są komunikaty do pobrania. Ten obiekt jest znany jako *wyzwalanie*.

Więcej informacji na temat kanałów wyzwalających zawiera sekcja Wyzwalanie kanałów .

Co to jest wyzwalanie?

Menedżer kolejek definiuje pewne warunki jako stanowiące *zdarzenia wyzwalające*.

Jeśli wyzwalanie jest włączone dla kolejki i wystąpi zdarzenie wyzwalające, menedżer kolejek wysyła *komunikat wyzwalacza* do kolejki o nazwie *kolejka inicjujowa*. Obecność komunikatu wyzwalacza w kolejce inicjuj. wskazuje, że wystąpiło zdarzenie wyzwalające.

Komunikaty wyzwalacza wygenerowane przez menedżer kolejek nie są trwałe. Zmniejsza to rejestrowanie (skutkuje poprawą wydajności) i minimalizuje duplikaty podczas restartu, a więc poprawia czas restartu.

Program, który przetwarza kolejkę inicjujący, jest nazywany *aplikacją monitora wyzwalacza*, a jego funkcją jest odczytywanie komunikatu wyzwalacza i podjęcie odpowiednich działań w oparciu o informacje zawarte w komunikacie wyzwalacza. Zazwyczaj to działanie polega na uruchomieniu innej aplikacji w celu przetworzenia kolejki, która wygenerował komunikat wyzwalacza. Z punktu widzenia menedżera kolejek nie ma nic specjalnego w aplikacji wyzwalacza-monitor; jest to po prostu inna aplikacja, która odczytuje komunikaty z kolejki (kolejka inicjujący).

Jeśli wyzwalanie jest włączone dla kolejki, można utworzyć *obiekt definicji procesu* powiązany z tą kolejką. Ten obiekt zawiera informacje na temat aplikacji, która przetwarza komunikat, który spowodował zdarzenie wyzwalające. Jeśli tworzony jest obiekt definicji procesu, menedżer kolejek wyodrębnia te informacje i umieszcza je w komunikacie wyzwalaczem w celu użycia przez aplikację monitorującego wyzwalacza. Nazwa definicji procesu powiązana z kolejką jest nadawana przez atrybut lokalnej kolejki produktu *ProcessName* . Każda kolejka określa inną definicję procesu lub kilka kolejek może współużytkować definicję procesu.

Jeśli uruchomienie kanału ma być wyzwalane, nie trzeba definiować obiektu definicji procesu. Zamiast niej używana jest definicja kolejki transmisji.

Wyzwalanie jest obsługiwane przez klienty WebSphere MQ działające w następujących środowiskach:

- Systemy UNIX and Linux
- Systemy Windows

Aplikacja działająca w środowisku klienta jest taka sama, jak działająca w pełnym środowisku WebSphere MQ , z tą różnicą, że połączono ją z bibliotekami klienta. Jednak monitor wyzwalacza i aplikacja, która ma zostać uruchomiona, muszą znajdować się w tym samym środowisku.

Wyzwalanie obejmuje:

Kolejka aplikacji
Kolejka aplikacji jest kolejką lokalną, która w momencie wyzwalania i gdy spełnione są warunki, wymaga, aby komunikaty wyzwalacza były zapisywane.

Definicja procesu
Z kolejką aplikacji może być powiązany *obiekt definicji procesu* zawierający szczegółowe informacje na temat aplikacji, które będą dostawać komunikaty z kolejki aplikacji. (Lista atrybutów znajduje się w sekcji Atrybuty definicji procesów).

Należy pamiętać, że jeśli wyzwalacz ma uruchamiać kanał, nie ma potrzeby definiowania obiektu definicji procesu.

Kolejka transmisji
Jeśli wyzwalacz ma być uruchamiany w celu uruchomienia kanału, potrzebna jest kolejka transmisji.

W przypadku kolejki transmisji w systemach AIX, HP-UX, IBM i, Solaris, z/OSlub Windows atrybut *TriggerData* kolejki transmisji może określać nazwę kanału, który ma być uruchomiony. Może to zastąpić definicję procesu wyzwalającą kanały, ale jest ona używana tylko wtedy, gdy definicja procesu nie została utworzona.

zdarzenie wyzwalające
Zdarzenie wyzwalające to zdarzenie, które powoduje wygenerowanie komunikatu wyzwalacza przez menedżer kolejek. Zwykle jest to komunikat, który przylatuje do kolejki aplikacji, ale może również wystąpić w innych sytuacjach (patrz sekcja “Warunki dla zdarzenia wyzwalającego” na stronie 344). Produkt WebSphere MQ oferuje szereg opcji umożliwiających sterowanie warunkami, które powodują zdarzenie wyzwalające (patrz “Sterowanie zdarzeniami wyzwalającymi” na stronie 348).

komunikat wyzwalacza
Menedżer kolejek tworzy *komunikat wyzwalacza* , gdy rozpoznaje zdarzenie wyzwalające (patrz “Warunki dla zdarzenia wyzwalającego” na stronie 344). Kopiuje on do komunikatu wyzwalacza informacje o aplikacji, która ma zostać uruchomiona. Informacje te pochodzą z kolejki aplikacji oraz z obiektu definicji procesu powiązanego z kolejką aplikacji. Komunikaty wyzwalacza mają stały format (patrz sekcja “Format komunikatów wyzwalacza” na stronie 354).

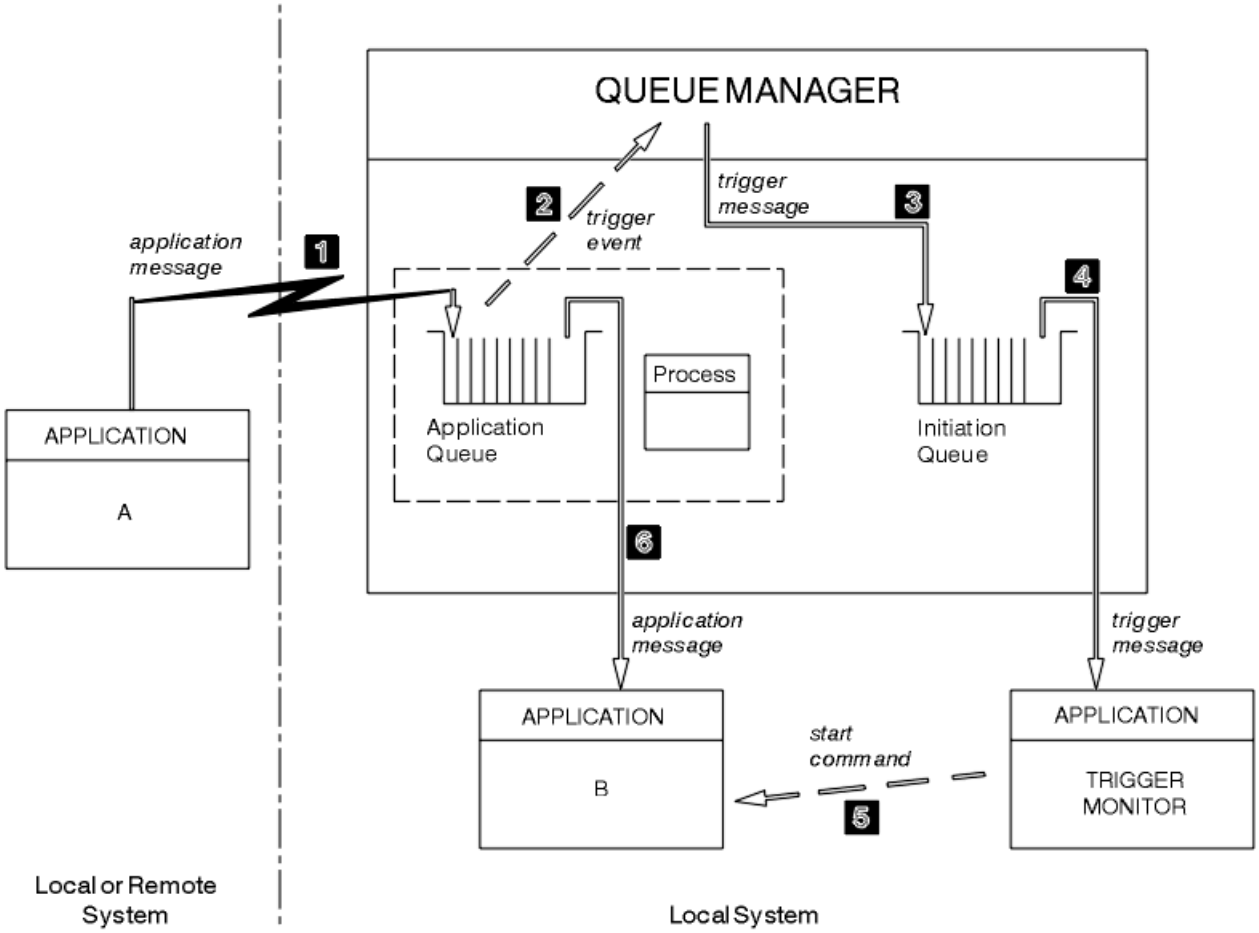
Kolejka inicjująca
Kolejka inicjuj. jest kolejką lokalną, w której menedżer kolejek umieszcza komunikaty wyzwalacza. Należy pamiętać, że kolejka inicjujący nie może być kolejką aliasową ani kolejką modelową. Menedżer kolejek może być właścicielem więcej niż jednej kolejki inicjujowej, a każda z nich jest powiązana z jedną lub większą liczbę kolejek aplikacji. Kolejka współużytkowana, kolejka lokalna dostępna dla menedżerów kolejek w grupie współużytkowania kolejek, może być kolejką inicjują w produkcie WebSphere MQ dla systemu z/OS.

monitor wyzwalacza
Monitor wyzwalacza to działający w sposób ciągły program, który służy do wykonywania jednej lub większej liczby kolejek inicjuj. Gdy komunikat wyzwalacza przybywa do kolejki inicjującej, monitor wyzwalacza wczytuje ten komunikat. Monitor wyzwalacza używa informacji znajdujących się w komunikacie wyzwalacza. Wysyła ona komendę uruchamiające aplikację, która ma pobierać komunikaty przychodzące do kolejki aplikacji, przekazując informacje zawarte w nagłówku komunikatu wyzwalacza, który zawiera nazwę kolejki aplikacji.

Na wszystkich platformach specjalny monitor wyzwalacza znany jako inicjator kanału jest odpowiedzialny za uruchamianie kanałów. W systemie z/OSinicjator kanału jest zwykle uruchamiany ręcznie lub może być uruchamiany automatycznie, gdy menedżer kolejek rozpoczyna się od zmiany CSQINP2 w startowym JCL menedżera kolejek. Na innych platformach jest ona uruchamiana automatycznie po uruchomieniu menedżera kolejek lub ręcznie przy użyciu komendy runmqchi.

(Więcej informacji na ten temat zawiera sekcja “Przetwarzanie kolejki inicjuj przez monitory wyzwalacza” na stronie 351).

Aby zrozumieć, w jaki sposób działa wyzwalanie, należy rozważyć Rysunek 68 na stronie 340, który jest przykładem typu wyzwalacza FIRST (MQTT_FIRST).



Rysunek 68. Przepływ komunikatów aplikacji i wyzwalaczy

W programie Rysunek 68 na stronie 340sekwencja zdarzeń jest następująca:

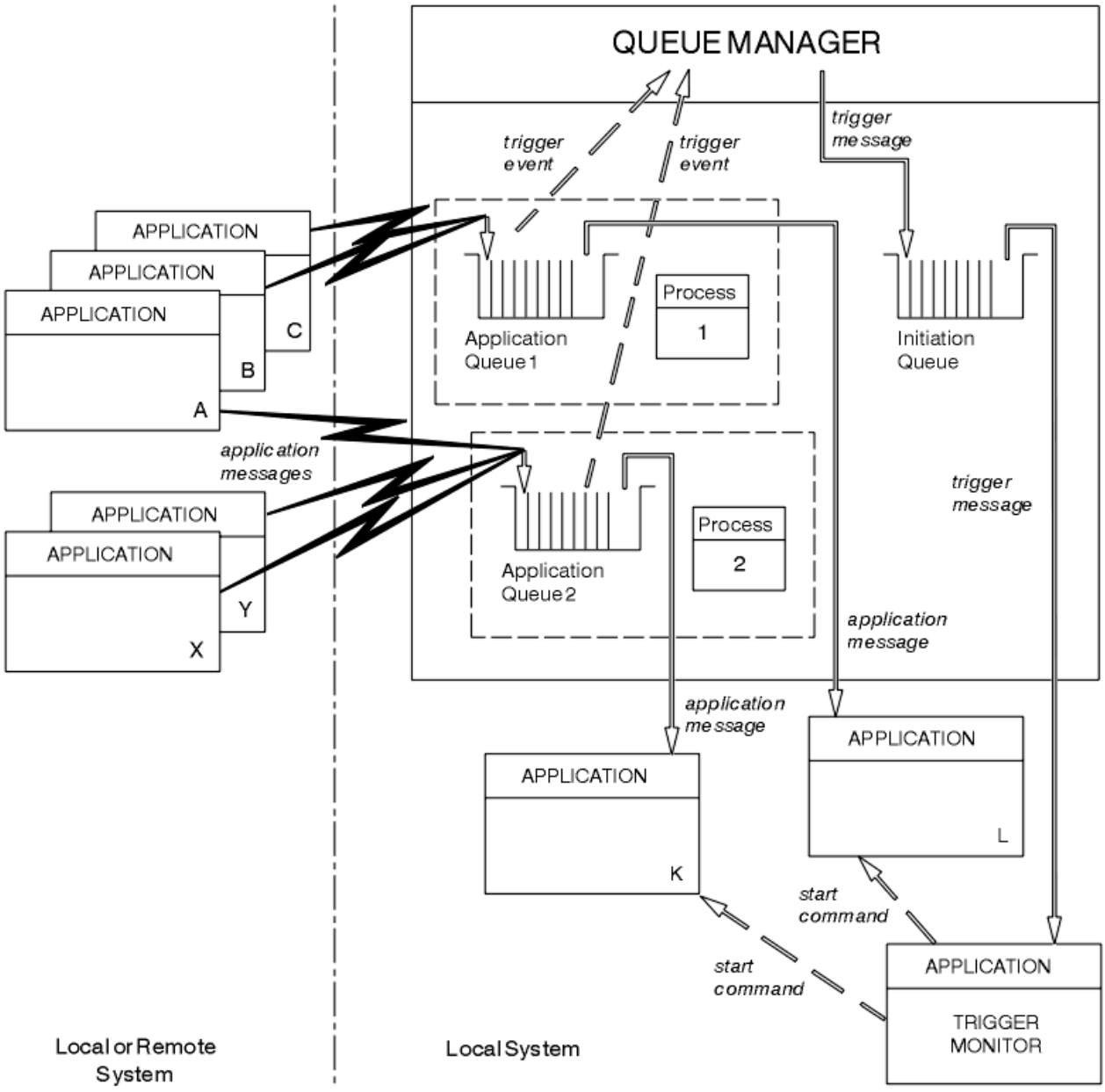
1. Aplikacja A, która może być lokalna lub zdalna w stosunku do menedżera kolejek, umieszcza komunikat w kolejce aplikacji. Żadna aplikacja nie ma tej kolejki otwartej na dane wejściowe. Jednak ten fakt ma znaczenie tylko w przypadku wyzwalaczy typu FIRST i DEPTH.
2. Menedżer kolejek sprawdza, czy spełnione są warunki, na podstawie których musi wygenerować zdarzenie wyzwalające. Są one generowane i generowane jest zdarzenie wyzwalające. Informacje przechowywane w powiązanym obiekcie definicji procesu są używane podczas tworzenia komunikatu wyzwalacza.
3. Menedżer kolejek tworzy komunikat wyzwalacza i umieszcza go w kolejce inicjuj. powiązanej z tą kolejką aplikacji, ale tylko wtedy, gdy aplikacja (monitor wyzwalacza) ma otwartą kolejkę inicjują do wejścia.
4. Monitor wyzwalacza pobiera komunikat wyzwalacza z kolejki inicjuj.
5. Monitor wyzwalacza wysyła komendę do uruchomienia aplikacji B (aplikacji serwera).
6. Aplikacja B otwiera kolejkę aplikacji i pobiera komunikat.

Uwaga:

1. Jeśli kolejka aplikacji jest otwarta dla danych wejściowych, przez dowolny program i ma ustawioną wartość FIRST lub DEPTH, żadne zdarzenie wyzwalające nie będzie miało miejsca, ponieważ kolejka jest już obsługiwana.
2. Jeśli kolejka inicjujący nie jest otwarta dla danych wejściowych, menedżer kolejek nie generuje komunikatów wyzwalacza; oczekuje do momentu otwarcia przez aplikację kolejki inicjujowej dla danych wejściowych.

3. W przypadku wyzwalania dla kanałów należy użyć wyzwalacza typu FIRST lub DEPTH.
4. Wyzwalane aplikacje działają pod identyfikatorem użytkownika i grupą użytkownika, który uruchomił monitor wyzwalacza, użytkownika programu CICS lub użytkownika, który uruchomił menedżer kolejek.

Dotychczas relacja między kolejkami w ramach wyzwalania była tylko na jednej, na jednej podstawie. Rozważ Rysunek 69 na stronie 341.



Rysunek 69. Relacja kolejek w wyzwalaniu

Z kolejką aplikacji jest powiązany obiekt definicji procesu, który zawiera szczegółowe informacje na temat aplikacji, która będzie przetwarzała komunikat. Menedżer kolejek umieszcza informacje w komunikacie wyzwalacza, dlatego konieczna jest tylko jedna kolejka inicjacyjna. Monitor wyzwalacza wyodrębnia te informacje z komunikatu wyzwalacza i uruchamia odpowiednią aplikację w celu zajmowania się komunikatem w każdej kolejce aplikacji.

Należy pamiętać, że aby wyzwolić początek kanału, nie trzeba definiować obiektu definicji procesu. Definicja kolejki transmisji może określić kanał, który ma zostać wyzwolony.

Użyj poniższych odsyłaczy, aby dowiedzieć się więcej na temat uruchamiania aplikacji WebSphere MQ za pomocą wyzwalaczy:

- “Wymagania wstępne dla wyzwalania” na stronie 342
- “Warunki dla zdarzenia wyzwalającego” na stronie 344
- “Sterowanie zdarzeniami wyzwalającymi” na stronie 348
- “Projektowanie aplikacji, która korzysta z wyzwalanych kolejek” na stronie 350
- “Przetwarzanie kolejki inicjuj przez monitory wyzwalacza” na stronie 351
- “Właściwości komunikatów wyzwalacza” na stronie 354
- “Gdy wyzwalanie nie działa” na stronie 355

Pojęcia pokrewne

“Interfejs kolejki komunikatów-przegląd” na stronie 199
Informacje na temat komponentów interfejsu kolejek komunikatów (Message Queue Interface-MQI).

“Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego” na stronie 211
Aby można było korzystać z usług programistycznych produktu WebSphere MQ , program musi mieć połączenie z menedżerem kolejek. Informacje zawarte w tej sekcji umożliwiają zapoznanie się z informacjami na temat nawiązywania połączenia z menedżerem kolejek i z jego rozłączeniem.

“Otwieranie i zamykanie obiektów” na stronie 220
Te informacje udostępniają wgląd w otwieranie i zamykanie obiektów WebSphere MQ .

“Umieszczanie komunikatów w kolejce” na stronie 231
Te informacje umożliwiają zapoznanie się z informacjami na temat umieszczania komunikatów w kolejce.

“Pobieranie komunikatów z kolejki” na stronie 246
Ta sekcja zawiera informacje na temat pobierania komunikatów z kolejki.

“Sprawdzanie i ustawianie atrybutów obiektu” na stronie 328
Atrybuty to właściwości, które definiują parametry obiektu WebSphere MQ .

“Zatwierdzanie i wycofywanie jednostek pracy” na stronie 331
W tej sekcji opisano sposób zatwierdzania i tworzenia kopii zapasowych wszystkich możliwych do odtworzenia operacji get i put, które wystąpiły w jednostce pracy.

“Praca z interfejsem MQI i klastrami” na stronie 355
Istnieją specjalne opcje dotyczące wywołań i kodów powrotu, które odnoszą się do technologii klastrowej.

Wymagania wstępne dla wyzwalania

Te informacje umożliwiają zapoznanie się z krokami, które należy wykonać przed uruchomieniem wyzwalania.

Zanim aplikacja będzie mogła skorzystać z wyzwolenia, wykonaj następujące kroki:

1. Albo:
   a. Utwórz kolejkę inicjujową dla kolejki aplikacji. Na przykład:

```
DEFINE QLOCAL (initiation.queue) REPLACE      +
       LIKE (SYSTEM.DEFAULT.INITIATION.QUEUE)      +
       DESCR ('initiation queue description')
```

   lub wersji

   b. Określ nazwę kolejki lokalnej, która istnieje i może być używana przez aplikację (zwykle jest to nazwa SYSTEM.DEFAULT.INITIATION.QUEUE lub, jeśli uruchamiasz kanały z wyzwalaczami, SYSTEM.CHANNEL.INITQ) i podaj jego nazwę w polu *InitiationQName* w kolejce aplikacji.

2. Powiąż kolejkę inicjujową z kolejką aplikacji. Menedżer kolejek może być właścicielem więcej niż jednej kolejki inicjuj. Niektóre kolejki aplikacji mogą być obsługiwane przez różne programy. W takim

przypadku można użyć jednej kolejki inicjuj dla każdego programu obsługującego, mimo że nie jest to konieczne. Poniżej przedstawiono przykład tworzenia kolejki aplikacji:

```
DEFINE QLOCAL (application.queue) REPLACE   +
LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE)           +
DESCR ('appl queue description')    +
INITQ ('initiation.queue')          +
PROCESS ('process.name')            +
TRIGGER                                     +
TRIGTYPE (FIRST)
```

3. Jeśli wyzwalana jest aplikacja, utwórz obiekt definicji procesu, który będzie zawierał informacje dotyczące aplikacji, która ma obsługiwać kolejkę aplikacji. Na przykład, aby wyzwolić-uruchom transakcję CICS payroll o nazwie PAYR:

```
DEFINE PROCESS (process.name) +
       REPLACE +
       DESCR ('process description') +
       APPLICID ('PAYR') +
       APPLTYPE (CICS) +
       USERDATA ('Payroll data')
```

   Gdy menedżer kolejek tworzy komunikat wyzwalacza, kopiuje on informacje z atrybutów obiektu definicji procesu do komunikatu wyzwalacza.

Platforma	**Aby utworzyć obiekt definicji procesu**
Systemy UNIX, Linuxi Windows	Użyj opcji DEFINE PROCESS lub SYSTEM.DEFAULT.PROCESS i zmodyfikuj za pomocą instrukcji ALTER PROCESS

4. Opcjonalnie: utwórz definicję kolejki transmisji i użyj odstępów dla atrybutu *ProcessName* .

   Atrybut *TrigData* może zawierać nazwę kanału, który ma zostać wyzwolony, lub może on pozostać pusty. Jeśli pole IBM WebSphere MQ for z/OSpozostanie puste, inicjator kanału przeszukuje pliki definicji kanału, dopóki nie znajdzie kanału, który jest powiązany z nazwaną kolejką transmisji. Gdy menedżer kolejek tworzy komunikat wyzwalacza, kopiuje on informacje z atrybutu *TrigData* definicji kolejki transmisji do komunikatu wyzwalacza.

5. Jeśli obiekt definicji procesu został utworzony w celu określenia właściwości aplikacji, która ma obsługiwać kolejkę aplikacji, powiąże obiekt procesu z kolejką aplikacji, nadając mu nazwę w atrybucie *ProcessName* w kolejce.

Platforma	**Użyj komend**
Systemy UNIX, Linuxi Windows	ALTER QLOCAL

6. Uruchomienie instancji monitorów wyzwalacza , które mają służyć kolejkom inicjujowym, które zostały zdefiniowane. Więcej informacji na ten temat zawiera sekcja “Przetwarzanie kolejki inicjuj przez monitory wyzwalacza” na stronie 351 .

Aby uzyskać informacje o wszystkich niedostarczonych komunikatach wyzwalających, należy upewnić się, że menedżer kolejek ma zdefiniowaną kolejkę niedostarczonych komunikatów (niedostarczonych komunikatów). Podaj nazwę kolejki w polu menedżera kolejek produktu *DeadLetterQName* .

Następnie można ustawić wymagane warunki wyzwalacza, korzystając z atrybutów obiektu kolejki, które definiują kolejkę aplikacji. Więcej informacji na ten temat zawiera sekcja “Sterowanie zdarzeniami wyzwalającymi” na stronie 348.

Warunki dla zdarzenia wyzwalającego

Odwołania do kolejek współużytkowanych w tym temacie oznaczają kolejki współużytkowane w grupie współużytkowania kolejek, które są dostępne tylko w produkcie WebSphere MQ for z/OS.

Menedżer kolejek tworzy komunikat wyzwalacza, gdy spełnione są następujące warunki:

1. Komunikat jest *umieszczany* w kolejce.
2. Priorytet komunikatu ma priorytet większy niż lub równy progowi, który jest priorytetem kolejki. Ten priorytet jest ustawiany w atrybucie kolejki lokalnej *TriggerMsgPriority* ; jeśli jest ustawiony na zero, każdy komunikat kwalifikuje się do jakości.
3. Liczba komunikatów w kolejce z priorytetem większym lub równym *TriggerMsgPriority* była poprzednio, w zależności od *TriggerType*:
   - Zero (dla wyzwalacza typu MQTT_FIRST)
   - Dowolna liczba (dla wyzwalacza typu MQTT_EVERY)
   - *TriggerDepth* minus 1 (dla typu wyzwalacza MQTT_DEPTH)

   Uwaga:

   a. W przypadku niewspółużytkowanych kolejek lokalnych menedżer kolejek zlicza zarówno zatwierdzone, jak i niezatwierdzone komunikaty, gdy ocenia, czy istnieją warunki dla zdarzenia wyzwalającego. W rezultacie aplikacja może zostać uruchomiona, gdy nie ma żadnych komunikatów do pobrania, ponieważ komunikaty w kolejce nie zostały zatwierdzone. W takiej sytuacji należy rozważyć użycie opcji wait z odpowiednim produktem *WaitInterval*, tak aby aplikacja oczekiwała na dotarcie do niego komunikatów.

   b. W przypadku lokalnych kolejek współużytkowanych menedżer kolejek liczy tylko zatwierdzone komunikaty.
4. Dla wyzwalania typu FIRST lub DEPTH, żaden program nie ma otwartej kolejki aplikacji na potrzeby usuwania komunikatów (oznacza to, że atrybut kolejki lokalnej *OpenInputCount* jest równy zero).

   Uwaga:

   a. W przypadku kolejek współużytkowanych warunki specjalne mają zastosowanie, gdy wiele menedżerów kolejek wyzwala monitory działające względem kolejki. W takiej sytuacji, jeśli co najmniej jeden menedżer kolejek ma otwartą kolejkę dla danych wejściowych współużytkowanych, kryteria wyzwalacza dla pozostałych menedżerów kolejek są traktowane jako *TriggerType* MQTT_FIRST i *TriggerMsgPriority* zero. Gdy wszystkie menedżery kolejek zamkną kolejkę dla danych wejściowych, warunki wyzwalacza są przywracane do warunków określonych w definicji kolejki.

      Przykładowym scenariuszem, na który wpływa ten warunek, jest wiele menedżerów kolejek QM1, QM2i QM3 z monitorem wyzwalacza uruchomionym dla kolejki aplikacji A. Komunikat dociera do A spełniającego warunki wyzwalania, a w kolejce inicjacji generowany jest komunikat wyzwalacza. Monitor wyzwalacza w menedżerze kolejek QM1 pobiera komunikat wyzwalacza i wyzwala aplikację. Wyzwalana aplikacja otwiera kolejkę aplikacji dla współużytkowanych danych wejściowych. Od tego momentu w przypadku warunków wyzwalacza dla kolejki aplikacji A są one oceniane jako *TriggerType* MQTT_FIRST, a *TriggerMsgPriority* zero w menedżerach kolejek QM2 i QM3, dopóki QM1 nie zamknie kolejki aplikacji.

   b. W przypadku kolejek współużytkowanych ten warunek jest stosowany dla każdego menedżera kolejek. Oznacza to, że *OpenInputCount* dla kolejki menedżera kolejek musi być zerem dla komunikatu wyzwalacza, który ma zostać wygenerowany dla kolejki przez tego menedżera kolejek. Jeśli jednak dowolny menedżer kolejek w grupie współużytkowania kolejki ma otwartą kolejkę przy użyciu opcji MQOO_INPUT_EXCLUSIVE, dla tej kolejki nie zostanie wygenerowany żaden komunikat wyzwalacza przez żaden z menedżerów kolejek w grupie współużytkowania kolejki.

      Zmiana sposobu wartościowania warunków wyzwalacza jest wykonywana, gdy wyzwalana aplikacja otwiera kolejkę dla danych wejściowych. W scenariuszach, w których działa tylko jeden monitor wyzwalacza, inne aplikacje mogą mieć ten sam efekt, ponieważ w podobny sposób

otwierają kolejkę aplikacji na potrzeby wprowadzania danych. Nie ma znaczenia, czy kolejka aplikacji została otwarta przez aplikację, która jest uruchamiana przez monitor wyzwalacza, czy przez inną aplikację. Jest to fakt, że kolejka jest otwarta dla wejścia w innym menedżerze kolejek, który powoduje zmianę kryteriów wyzwalacza.

5. W produkcie WebSphere MQ for z/OS, jeśli kolejka aplikacji jest kolejką z atrybutem *Usage* o wartości MQUS_NORMAL, żądania pobrania dla niej nie są zablokowane (to znaczy atrybut kolejki *InhibitGet* to MQQA_GET_ALLOWED). Ponadto, jeśli wyzwalana kolejka aplikacji jest kolejką z atrybutem *Usage* o wartości MQUS_XMITQ, żądania pobrania dla niej nie są blokowane.
6. Albo:
   - Atrybut kolejki lokalnej *ProcessName* dla kolejki nie jest pusty, a obiekt definicji procesu identyfikowany przez ten atrybut został utworzony, lub
   - Atrybut kolejki lokalnej *ProcessName* dla kolejki jest pusty, ale kolejka jest kolejką transmisji. Ponieważ definicja procesu jest opcjonalna, atrybut *TriggerData* może również zawierać nazwę kanału, który ma zostać uruchomiony. W tym przypadku komunikat wyzwalacza zawiera atrybuty z następującymi wartościami:
     - *QName*: nazwa kolejki
     - *ProcessName*: odstępy
     - *TriggerData*: wyzwalanie danych
     - *ApplType*: MQAT_UNKNOWN
     - *ApplId*: odstępy
     - *EnvData*: odstępy
     - *UserData*: odstępy
7. Utworzono kolejkę inicjujową i określono ją w atrybucie kolejki lokalnej *InitiationQName* . Ponadto:
   - Żądania pobierania nie są zablokowane dla kolejki inicjuj (to znaczy, że atrybut kolejki *InhibitGet* to MQQA_GET_ALLOWED).
   - Żądania umieszczania nie mogą być blokowane dla kolejki inicjuj (to znaczy, że atrybut kolejki *InhibitPut* musi mieć wartość MQQA_PUT_ALLOWED).
   - Atrybut *Usage* kolejki inicjujący musi mieć wartość MQUS_NORMAL.
   - W środowiskach, w których obsługiwane są kolejki dynamiczne, kolejka inicjujący nie może być kolejką dynamiczną, która została oznaczona jako logicznie usunięta.
8. Monitor wyzwalacza aktualnie ma otwartą kolejkę inicjują do usuwania komunikatów (oznacza to, że atrybut kolejki lokalnej *OpenInputCount* jest większy od zera).
9. Element sterujący wyzwalacza (atrybut kolejki lokalnej*TriggerControl*) dla kolejki aplikacji jest ustawiony na wartość MQTC_ON. Aby to zrobić, należy ustawić atrybut *trigger* podczas definiowania kolejki lub użyć komendy ALTER QLOCAL.
10. Typ wyzwalacza (atrybut kolejki lokalnej*TriggerType*) nie jest typu MQTT_NONE.

    Jeśli wszystkie wymagane warunki są spełnione, a komunikat, który spowodował warunek wyzwalacza, jest umieszczany jako część jednostki pracy, komunikat wyzwalacza nie będzie dostępny do pobrania przez aplikację monitora wyzwalacza do momentu zakończenia jednostki pracy, niezależnie od tego, czy jednostka pracy została zatwierdzona, czy też dla typu wyzwalacza MQTT_FIRST lub MQTT_DEPTH, dla którego utworzono kopię zapasową.
11. Odpowiedni komunikat jest umieszczany w kolejce, dla elementu *TriggerType* o wartości MQTT_FIRST lub MQTT_DEPTH, a także w kolejce:
    - Wcześniej nie było puste (MQTT_FIRST), lub
    - W przypadku produktu *TriggerDepth* lub większej liczby komunikatów (MQTT_DEPTH)

    i warunki od “2” na stronie 344 do “10” na stronie 345 (z wyjątkiem “3” na stronie 344) są spełnione, jeśli w przypadku MQTT_FIRST upłynęło wystarczająco dużo czasu (atrybut menedżera

kolejek*TriggerInterval*), który upłynął od ostatniego komunikatu wyzwalacza, który został zapisany dla tej kolejki.

Ma to na celu umożliwienie serwerowi kolejek, który kończy się przed przetworami wszystkich komunikatów w kolejce. Celem odstępu czasu wyzwalacza jest zmniejszenie liczby wygenerowanych duplikatów komunikatów wyzwalacza.

Uwaga: Jeśli menedżer kolejek zostanie zatrzymany i zrestartowany, zostanie zresetowany *licznik czasu* *TriggerInterval* . Istnieje małe okno, podczas którego możliwe jest utworzenie dwóch komunikatów wyzwalacza. Okno istnieje, gdy atrybut wyzwalacza kolejki jest ustawiony na włączony w tym samym czasie co komunikat, a kolejka nie była wcześniej pusta (MQTT_FIRST) lub miała *TriggerDepth* lub więcej komunikatów (MQTT_DEPTH).

12. Jedyna aplikacja obsługując kolejkę wywołuje wywołanie MQCLOSE dla partycji *TriggerType* z MQTT_FIRST lub MQTT_DEPTH, a ponadto istnieje co najmniej:

    - Jedna (MQTT_FIRST), lub
    - *TriggerDepth* (MQTT_DEPTH)

    Komunikaty w kolejce o odpowiednim priorytecie (warunek “2” na stronie 344) i warunki “6” na stronie 345 za pośrednictwem “10” na stronie 345 są również spełnione.

    Ma to na celu umożliwienie serwerowi kolejek, który zgłasza wywołanie MQGET, opróżnia kolejkę i tak się kończy. Jednak w przedziale czasu między wywołaniami MQGET i MQCLOSE należy dotrzeć do jednego lub większej liczby komunikatów.

    Uwaga:

    a. Jeśli program obsługujący kolejkę aplikacji nie pobierze wszystkich komunikatów, może to spowodować pętlę zamkniętą. Za każdym razem, gdy program zamknie kolejkę, menedżer kolejek tworzy kolejny komunikat wyzwalacza, który powoduje, że monitor wyzwalacza ponownie uruchomi program serwera.

    b. Jeśli program obsługujący kolejkę aplikacji wycofał żądanie pobrania (lub jeśli program się przerwie) przed zamknięciem kolejki, to samo dzieje się. Jeśli jednak program zamknie kolejkę przed utworzeniem kopii zapasowej żądania pobrania, a kolejka jest pusta, nie zostanie utworzony żaden komunikat wyzwalacza.

    c. Aby zapobiec występowaniu takiej pętli, należy użyć pola *BackoutCount* deskryptora MQMD w celu wykrycia komunikatów, które są wielokrotnie wycofane. Więcej informacji na ten temat zawiera sekcja “Komunikaty, których kopie zapasowe zostały wycofane” na stronie 38.

13. Przy użyciu komendy MQSET lub komendy spełnione są następujące warunki:

    a. • *TriggerControl* zmieniono na MQTC_ON, lub
    • *TriggerControl* jest już MQTC_ON, a wartość *TriggerType*, *TriggerMsgPriority*lub *TriggerDepth* (jeśli jest odpowiednia) została zmieniona,

    i jest co najmniej:

    - Jedna (MQTT_FIRST lub MQTT_EVERY), lub
    - *TriggerDepth* (MQTT_DEPTH)

    Komunikaty w kolejce o odpowiednim priorytecie (warunek “2” na stronie 344) i warunki od “4” na stronie 344 do “10” na stronie 345 (z wyjątkiem “8” na stronie 345) są również spełnione.

    Jest to możliwe, aby aplikacja lub operator zmieniał kryteria wyzwalania, gdy warunki dla wyzwalacza do wystąpienia są już spełnione.

    b. Atrybut kolejki *InhibitPut* kolejki inicjujący zmienia się z MQQA_PUT_INHIBITED na MQQA_PUT_ALLOWED, a istnieje co najmniej:

    - Jedna (MQTT_FIRST lub MQTT_EVERY), lub
    - *TriggerDepth* (MQTT_DEPTH)

    Komunikaty o odpowiednim priorytecie (warunek “2” na stronie 344) dla dowolnej kolejki, dla której jest to kolejka inicjujący, oraz warunki “4” na stronie 344 za pomocą “10” na stronie 345

są również spełnione. (Jeden komunikat wyzwalacza jest generowany dla każdej takiej kolejki spełniający warunki.)

Polega to na tym, że komunikaty wyzwalacza nie są generowane z powodu warunku MQQA_PUT_INHIBITED w kolejce inicjuj, ale ten warunek został zmieniony.

c. Atrybut kolejki *InhibitGet* kolejki aplikacji zmienia się z tabeli MQQA_GET_INHIBITED na wartość MQQA_GET_ALLOWED, a ponadto istnieje co najmniej:

- Jedna (MQTT_FIRST lub MQTT_EVERY), lub
- *TriggerDepth* (MQTT_DEPTH)

Komunikaty o odpowiednim priorytecie (warunek "2" na stronie 344) w kolejce i warunki od "4" na stronie 344 do "10" na stronie 345, z wyjątkiem "5" na stronie 345, są również spełnione.

Dzięki temu aplikacje mogą być wyzwalane tylko wtedy, gdy będą mogły pobierać komunikaty z kolejki aplikacji.

d. Aplikacja wyzwalacza-monitor wywołuje wywołanie MQOPEN dla danych wejściowych z kolejki inicjuj-i jest to co najmniej:

- Jedna (MQTT_FIRST lub MQTT_EVERY), lub
- *TriggerDepth* (MQTT_DEPTH)

Komunikaty o odpowiednim priorytecie (warunek "2" na stronie 344) dla dowolnej kolejki aplikacji, dla której jest to kolejka inicjujący, oraz warunki od "4" na stronie 344 do "10" na stronie 345 (z wyjątkiem "8" na stronie 345) są również spełnione, a żadna inna aplikacja nie ma otwartej kolejki inicjowania dla danych wejściowych (dla każdej takiej kolejki spełniający warunki generowane jest jeden komunikat wyzwalacza).

Ma to na celu umożliwienie komunikatów przychodzących do kolejek, gdy monitor wyzwalacza nie jest uruchomiony, a dla menedżera kolejek restartowanie i wyzwalanie komunikatów (które są nietrwałe) są tracone.

14. Parametr MSGDLVSQ jest ustawiony poprawnie. Jeśli parametr MSGDLVSQ=FIFO zostanie ustawiony, komunikaty będą dostarczane do kolejki w pierwszej pierwszej bazie danych. Priorytet komunikatu jest ignorowany, a domyślny priorytet kolejki jest przypisywany do komunikatu. Jeśli wartość *TriggerMsgPriority* jest ustawiona na wartość wyższą niż domyślny priorytet kolejki, komunikaty nie są wyzwalane. Jeśli parametr *TriggerMsgPriority* jest ustawiony na wartość równą lub niższą niż domyślny priorytet kolejki, wyzwalanie odbywa się dla typu FIRST, EVERY i DEPTH. Informacje na temat tych typów można znaleźć w opisie pola *TriggerType* w sekcji "Sterowanie zdarzeniami wyzwalającymi" na stronie 348.

    Jeśli parametr MSGDLVSQ=PRIORYTET zostanie ustawiony, a priorytet komunikatu jest równy lub większy niż pole *TriggerMsgPriority* , komunikaty będą tylko *liczyć* w kierunku zdarzenia wyzwalającego. W tym przypadku wyzwalanie odbywa się dla typu FIRST, EVERY i DEPTH. Na przykład: jeśli zostanie wstawione 100 komunikatów o niższym priorytecie niż *TriggerMsgPriority*, efektywna głębokość kolejki dla celów wyzwalających będzie nadal zerowa. Jeśli następnie w kolejce zostanie wstawiony inny komunikat, ale tym razem priorytet jest większy lub równy *TriggerMsgPriority*, efektywna głębokość kolejki zwiększa się z zera do jednego, a warunek dla *TriggerType* FIRST jest spełniony.

Uwaga:

1. W kroku "12" na stronie 346 (w przypadku gdy komunikaty wyzwalacza są generowane w wyniku zdarzenia innego niż komunikat przylot do kolejki aplikacji), komunikat wyzwalacza nie jest umieszczany w jednostce pracy. Ponadto, jeśli parametr *TriggerType* ma wartość MQTT_EVERY, a w kolejce aplikacji występuje co najmniej jeden komunikat, generowany jest tylko jeden komunikat wyzwalacza.
2. Jeśli produkt WebSphere MQ segmentuje komunikat podczas operacji MQPUT, zdarzenie wyzwalające nie zostanie przetworzone, dopóki wszystkie segmenty nie zostaną pomyślnie umieszczone w kolejce. Jednak po tym, jak segmenty komunikatów znajdują się w kolejce, produkt WebSphere MQ traktuje je jako pojedyncze komunikaty do celów wyzwalania. Na przykład pojedynczy komunikat logiczny podzielony na trzy części powoduje przetwarzanie tylko jednego zdarzenia wyzwalającego, gdy jest to

pierwsza operacja MQPUT i segmentowana. Jednak każdy z tych trzech segmentów powoduje, że ich własne zdarzenia wyzwalające są przetwarzane w miarę ich przenoszenia przez sieć WebSphere MQ .

Sterowanie zdarzeniami wyzwalającymi

Zdarzenia wyzwalające sterują za pomocą niektórych atrybutów, które definiują kolejkę aplikacji. Informacje te zawierają również przykłady użycia typów wyzwalaczy: EVERY, FIRST i DEPTH.

Można włączyć i wyłączyć wyzwalanie, a także wybrać liczbę lub priorytet komunikatów, które są liczone w kierunku zdarzenia wyzwalającego. W sekcji Atrybuty obiektówznajduje się pełny opis tych atrybutów.

Odpowiednie atrybuty są następujące:

TriggerControl
Ten atrybut służy do włączania i wyłączania wyzwalania dla kolejki aplikacji.

TriggerMsgPriority
Minimalny priorytet, jaki musi być dla niego komunikat, aby był on liczony w kierunku zdarzenia wyzwalającego. Jeśli komunikat o priorytecie mniejszym niż *TriggerMsgPriority* dociera do kolejki aplikacji, menedżer kolejek zignoruje komunikat, gdy określi, czy ma zostać utworzony komunikat wyzwalacza. Jeśli wartość *TriggerMsgPriority* jest ustawiona na zero, wszystkie komunikaty są liczone w kierunku zdarzenia wyzwalającego.

TriggerType
Oprócz typu wyzwalacza NONE (który wyłącza wyzwalanie tak, jak ustawienie opcji *TriggerControl* na OFF), można użyć następujących typów wyzwalaczy, aby ustawić czułość kolejki w celu wyzwolenia zdarzeń:

Każdy	Zdarzenie wyzwalające występuje za każdym razem, gdy komunikat pojawia się w kolejce aplikacji. Tego typu wyzwalacza należy użyć w przypadku, gdy uruchomiono wiele instancji aplikacji.
pierwsza	Zdarzenie wyzwalające występuje tylko wtedy, gdy liczba komunikatów w kolejce aplikacji zostanie zmieniona z zera na jeden. Użyj tego typu wyzwalacza, jeśli chcesz, aby program obsługi był uruchamiany po nadejściu pierwszego komunikatu do kolejki, kontynuuj do momentu, aż nie będzie już więcej komunikatów do przetworzenia, a następnie zakończysz. Kolejka musi być zawsze przetwarzana, dopóki nie będzie pusta. Patrz także “Szczególny przypadek typu wyzwalacza FIRST” na stronie 349 .
Głębokość	Zdarzenie wyzwalające występuje tylko wtedy, gdy liczba komunikatów w kolejce aplikacji osiągnie wartość atrybutu *TriggerDepth* . Typowym zastosowaniem tego typu wyzwalania jest uruchomienie programu, gdy wszystkie odpowiedzi na zestaw żądań zostaną odebrane. **Wyzwalanie według głębokości:** W przypadku wyzwolenia przez głębokość menedżer kolejek wyłącza wyzwalanie (przy użyciu atrybutu < xph> < pv>TriggerControl< /pv> < /xph>) po utworzeniu komunikatu wyzwalacza. Aplikacja musi ponownie włączyć wyzwalanie samego siebie (za pomocą wywołania MQSET) po tym, jak to się stało. Działanie wyłączenia wyzwalania nie jest wykonywane w ramach elementu sterującego punktu synchronizacji, dlatego nie można ponownie włączyć wyzwalania, przy użyciu jednostki pracy. Jeśli program wycofuje żądanie umieszczenia, które spowodowało zdarzenie wyzwalające, lub jeśli program się przerwie, należy ponownie włączyć wyzwalanie za pomocą wywołania MQSET lub komendy ALTER QLOCAL.

TriggerDepth
Liczba komunikatów w kolejce, które powodują zdarzenie wyzwalające, gdy jest używane wyzwalanie przez głębokość.

Warunki, które muszą być spełnione dla menedżera kolejek w celu utworzenia komunikatu wyzwalacza, są opisane w sekcji “Warunki dla zdarzenia wyzwalającego” na stronie 344 .

Przykład użycia wyzwalacza typu BARDZO

Rozważ zastosowanie aplikacji, która generuje żądania dotyczące ubezpieczenia ruchowego. Aplikacja może wysyłać komunikaty żądań do wielu towarzystw ubezpieczeniowych, podając tę samą kolejkę odpowiedzi do kolejki za każdym razem. Może on ustawić wyzwalacz typu EVERY w tej kolejce odpowiedzi, tak aby za każdym razem, gdy nadejdzie odpowiedź, odpowiedź może wyzwoliwać instancję serwera w celu przetworzenia odpowiedzi.

Przykład użycia typu wyzwalacza FIRST

Należy wziąć pod uwagę organizację z wieloma biurami oddziałów, z których każda przekazuje szczegóły dotyczące dni pracy do centrali. Wszyscy robią to jednocześnie, pod koniec dnia pracy, a w centrali znajduje się aplikacja, która przetwarza detale ze wszystkich oddziałów oddziału. Pierwszy komunikat, który ma zostać dostarczony do centrali, może spowodować zdarzenie wyzwalające, które uruchamia tę aplikację. Ta aplikacja będzie kontynuować przetwarzanie, dopóki nie będzie więcej komunikatów znajdujących się w kolejce.

Przykład użycia wyzwalacza typu DEPTH

Należy rozważyć aplikację biura podróży, która tworzy pojedynczy wniosek o potwierdzenie rezerwacji lotu, potwierdzenie rezerwacji pokoju hotelowego, wynajem samochodu oraz zameldowanie niektórych podróżników. Aplikacja może rozdzielić te elementy na cztery komunikaty żądań, wysyłając każde z nich do osobnego miejsca docelowego. Może on ustawić wyzwalacz typu DEPTH w kolejce odpowiedzi do kolejki (z ustawioną głębią ustawioną na wartość 4), tak aby był on restartowany tylko wtedy, gdy wszystkie cztery odpowiedzi zostały nadesłane.

Jeśli w kolejce odpowiedzi przed ostatnią z czterech odpowiedzi nadchodzi inny komunikat (prawdopodobnie z innego żądania), aplikacja żądająca zostanie uruchomiona wcześniej. Aby tego uniknąć, podczas używania wyzwalacza DEPTH do gromadzenia wielu odpowiedzi na żądanie zawsze należy używać nowej kolejki odpowiedzi dla każdego żądania.

Szczególny przypadek typu wyzwalacza FIRST

W przypadku pierwszego typu wyzwalacza, jeśli w kolejce aplikacji istnieje już komunikat po nadejściu innego komunikatu, menedżer kolejek zwykle nie tworzy innego komunikatu wyzwalacza.

Jednak aplikacja obsługując kolejkę może w rzeczywistości nie otworzyć kolejki (na przykład może to zakończyć się aplikacją, prawdopodobnie z powodu problemu systemowego). Jeśli do obiektu definicji procesu została wstawiona niepoprawna nazwa aplikacji, aplikacja obsługując tę kolejkę nie odbierze żadnego z komunikatów. W takich sytuacjach, jeśli w kolejce aplikacji pojawi się inny komunikat, nie jest uruchomiony żaden serwer, który może przetworzyć ten komunikat (oraz inne komunikaty w kolejce).

Aby rozwiązać ten problem, menedżer kolejek tworzy kolejne komunikaty wyzwalacza w następujących okolicznościach:

- Jeśli w kolejce aplikacji pojawi się inny komunikat, ale tylko wtedy, gdy upłynął predefiniowany odstęp czasu od momentu utworzenia przez menedżer kolejek ostatniego komunikatu wyzwalacza dla tej kolejki. Ten przedział czasu jest zdefiniowany w atrybucie *TriggerInterval*menedżera kolejek. Jego wartość domyślna to 999 999 999 milisekund.
- W produkcie WebSphere MQ for z/OSkolejki aplikacji, których nazwa jest nazwą otwartej kolejki inicjuj, są okresowo skanowane. Jeśli od momentu wysłania ostatniego komunikatu wyzwalacza upłynono *TRIGINT* milisekund, a kolejka spełnia warunki dla zdarzenia wyzwalającego, a wartość CURDEPTH jest większa od zera, generowany jest komunikat wyzwalacza. Ten proces jest nazywany wyzwalaniem backstop.

Podczas podejmowania decyzji o wartości dla odstępu czasu wyzwalacza, który ma być używany w aplikacji, należy wziąć pod uwagę następujące kwestie:

- Jeśli wartość *TriggerInterval* jest ustawiona na niską wartość i nie ma aplikacji obsługując kolejkę aplikacji, wyzwalacz typu FIRST może zachowywać się jak typ wyzwalacza EVERY. Zależy to od szybkości umieszczania komunikatów w kolejce aplikacji, która z kolei może zależeć od innych działań systemu. Wynika to z faktu, że jeśli przedział czasu wyzwalacza jest bardzo mały, za każdym razem, gdy komunikat jest umieszczany w kolejce aplikacji, generowany jest inny komunikat wyzwalacza, nawet jeśli typ wyzwalacza jest typu FIRST, a nie EVERY. (Typ wyzwalacza FIRST z odstępem czasu wyzwalacza równy zero jest równoznaczny z typem wyzwalacza EVERY.)
- W produkcie WebSphere MQ for z/OS , jeśli wartość *TRIGINT* jest ustawiona na niską wartość i nie ma aplikacji obsługuającej kolejkę aplikacji FIRST typu wyzwalacza, wyzwalanie backstop wygeneruje komunikat wyzwalacza za każdym razem, gdy następuje okresowe skanowanie kolejek aplikacji o nazwach otwartych kolejek inicjujych.
- Jeśli tworzona jest kopia zapasowa jednostki pracy (patrz sekcja Wyzwalanie komunikatów i jednostek pracy), a przedział czasu wyzwalacza został ustawiony na wartość wysoką (lub wartość domyślną), to po wycofaniu jednostki pracy generowany jest jeden komunikat wyzwalacza. Jeśli jednak odstęp czasu wyzwalacza został ustawiony na niską wartość lub wartość zero (powoduje, że wyzwalacz typu FIRST to zachowuje się jak typ wyzwalacza EVERY), może zostać wygenerowany wiele komunikatów wyzwalacza. Jeśli kopia zapasowa jednostki pracy jest wycofana, wszystkie komunikaty wyzwalacza są nadal dostępne. Liczba komunikatów wyzwalacza, które są generowane, zależy od przedziału czasu wyzwalacza. Jeśli przedział czasu wyzwalacza jest ustawiony na zero, to generowana jest maksymalna liczba komunikatów.

Projektowanie aplikacji, która korzysta z wyzwalanych kolejek

Przekonałeś się, jak skonfigurować i kontrolować, wyzwalając dla swoich aplikacji. Poniżej znajdują się wskazówki, które należy wziąć pod uwagę podczas projektowania aplikacji.

Komunikaty wyzwalacza i jednostki pracy

Komunikaty wyzwalacza utworzone ze względu na zdarzenia wyzwalające, które nie są częścią jednostki pracy, są umieszczane w kolejce inicjujący, poza dowolną jednostką pracy, bez zależności od innych komunikatów i są dostępne do pobrania przez monitor wyzwalacza natychmiast.

Komunikaty wyzwalacza utworzone ze względu na zdarzenia wyzwalające, które są częścią jednostki pracy, są udostępniane w kolejce inicjuj, gdy jednostka pracy jest rozstrzygnięta, niezależnie od tego, czy jednostka pracy została zatwierdzona, czy wycofana

Jeśli menedżer kolejek nie umieje umieścić komunikatu wyzwalacza w kolejce inicjujący, zostanie on umieszczony w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Uwaga:

1. Menedżer kolejek liczy zarówno zatwierdzone, jak i niezatwierdzone komunikaty, gdy ocenia, czy istnieją warunki dla zdarzenia wyzwalającego.

   W przypadku wyzwalania typu FIRST lub DEPTH komunikaty wyzwalacza są udostępniane nawet wtedy, gdy jednostka pracy jest wycofana, tak aby komunikat wyzwalacza był zawsze dostępny, gdy spełnione są wymagane warunki. Na przykład rozważmy żądanie umieszczenia w jednostce pracy dla kolejki, która jest wyzwalana z typem wyzwalacza FIRST. Powoduje to utworzenie komunikatu wyzwalacza przez menedżer kolejek. Jeśli wystąpi inne żądanie umieszczenia z innej jednostki pracy, nie powoduje to innego zdarzenia wyzwalającego, ponieważ liczba komunikatów w kolejce aplikacji została zmieniona z jednego na dwa, co nie spełnia warunków dla zdarzenia wyzwalającego. Jeśli zostanie utworzona kopia zapasowa pierwszej jednostki pracy, ale druga jest zatwierdzona, komunikat wyzwalacza jest nadal tworzony.

   Oznacza to jednak, że komunikaty wyzwalacza są czasami tworzone, gdy warunki dla zdarzenia wyzwalającego *nie* są spełnione. Aplikacje, które wykorzystują wyzwalanie, muszą być zawsze przygotowane do obsługi tej sytuacji. Zaleca się użycie opcji wait z wywołaniem MQGET, ustawiając odpowiednią wartość na *WaitInterval* .

   Utworzone komunikaty wyzwalacza są zawsze udostępniane, niezależnie od tego, czy jednostka pracy jest wycofana, czy zatwierdzana.

2. W przypadku lokalnych kolejek współużytkowanych (czyli współużytkowanych kolejek w grupie współużytkowania kolejek) menedżer kolejek zlicza tylko zatwierdzone komunikaty.

Pobieranie komunikatów z wyzwalanej kolejki

Podczas projektowania aplikacji, które korzystają z wyzwalania, należy pamiętać o tym, że może wystąpić opóźnienie między uruchomieniem programu monitorującego wyzwalacz a innymi komunikatami dostępnymi w kolejce aplikacji. Może się to zdarzyć, gdy komunikat, który powoduje, że zdarzenie wyzwalające jest zatwierdzane przed innymi.

Aby umożliwić dotarcie komunikatów, należy zawsze używać opcji wait, gdy używane jest wywołanie MQGET w celu usunięcia komunikatów z kolejki, dla której ustawione są warunki wyzwalacza. Wartość *WaitInterval* musi być wystarczająca, aby umożliwić najdłuższy rozsądny czas między umieszczonym komunikatem i zatwierdzonym wywołaniem. Jeśli komunikat jest wysyłany ze zdalnego menedżera kolejek, ten czas ma wpływ na:

- Liczba komunikatów, które zostały umieszczone przed zatwierdzoną
- Szybkość i dostępność łącza komunikacyjnego
- Wielkości komunikatów

Przykład sytuacji, w której należy użyć wywołania MQGET z opcją oczekiwania, należy wziąć pod uwagę ten sam przykład, który był używany podczas opisywania jednostek pracy. To było żądanie umieszczenia w jednostce pracy dla kolejki, która jest wyzwalana z typem wyzwalacza FIRST. To zdarzenie powoduje, że menedżer kolejek tworzy komunikat wyzwalacza. Jeśli wystąpi inne żądanie umieszczenia z innej jednostki pracy, nie powoduje to innego zdarzenia wyzwalającego, ponieważ liczba komunikatów w kolejce aplikacji nie została zmieniona z 0 na 1. Jeśli zostanie utworzona kopia zapasowa pierwszej jednostki pracy, ale druga jest zatwierdzona, komunikat wyzwalacza jest nadal tworzony. Dlatego komunikat wyzwalacza jest tworzony w czasie, gdy tworzona jest kopia zapasowa pierwszej jednostki pracy. Jeśli przed zatwierdzoną drugą wiadomą opóźnienie zostanie opóźnione, wyzwolona aplikacja może zaczekać na jego działanie.

W przypadku wyzwalania typu DEPTH, opóźnienie może wystąpić nawet wtedy, gdy wszystkie istotne komunikaty zostaną ostatecznie zatwierdzone. Załóżmy, że atrybut kolejki *TriggerDepth* ma wartość 2. Po pojawie się dwóch komunikatów w kolejce, drugi komunikat powoduje utworzenie komunikatu wyzwalacza. Jeśli jednak drugi komunikat jest pierwszym, który ma zostać zatwierdzony, jest w tym momencie dostępny komunikat wyzwalacza. Monitor wyzwalacza uruchamia program serwera, ale program może pobrać tylko drugi komunikat, dopóki nie zostanie zatwierdzony pierwszy. Dlatego program może wymagać oczekiwania na udostępnienie pierwszego komunikatu.

Należy zaprojektować aplikację tak, aby była ona przerwana, jeśli nie będą dostępne żadne komunikaty do pobrania, gdy upłynie okres oczekiwania. Jeśli jeden lub więcej komunikatów przybędzie później, należy użyć ponownie uruchamianej aplikacji w celu ich przetworzenia. Ta metoda uniemożliwia bezczynność aplikacji i niepotrzebnie korzystanie z zasobów.

Przetwarzanie kolejki inicjuj przez monitory wyzwalacza

Dla menedżera kolejek monitor wyzwalacza jest podobny do dowolnej innej aplikacji, która obsługuje kolejkę. Monitor wyzwalacza obsługuje jednak kolejki inicjujący.

Monitor wyzwalacza jest zwykle programem ciągłym. Po nadejściu komunikatu wyzwalacza w kolejce inicjujący monitor wyzwalacza pobiera ten komunikat. Używa on informacji w komunikacie do wydania komendy uruchamiających aplikację, która przetwarza komunikaty w kolejce aplikacji.

Monitor wyzwalacza musi przekazać wystarczającą ilość informacji do programu, który jest uruchamiany, aby program mógł wykonać poprawne działania w poprawnej kolejce aplikacji.

Inicjator kanału jest przykładem specjalnego typu monitora wyzwalacza dla agentów kanałów komunikatów. Jednak w takiej sytuacji należy użyć albo wyzwalacza typu FIRST, albo DEPTH.

Monitory wyzwalacza w systemach UNIX i Windows

Ten temat zawiera informacje na temat monitorów wyzwalaczy udostępnianych w systemach UNIX i Windows .

Dla środowiska serwera dostępne są następujące monitory wyzwalacza:

amqstrg0
To jest przykładowy monitor wyzwalacza, który udostępnia podzbiór funkcji udostępnianej przez program **runmqtrm**. Więcej informacji na temat amqstrg0zawiera sekcja “Przykładowe programy dla platform rozproszonych” na stronie 99 .

runmqtrm
Składnia tej komendy jest następująca: **runmqtrm** [*-m QMgrName*] [*-q InitQ*], gdzie QMgrName to menedżer kolejek, a InitQ to kolejka inicjujący. Domyślna kolejka to SYSTEM.DEFAULT.INITIATION.QUEUE w domyślnym menedżerze kolejek. Wywołuje on programy dla odpowiednich komunikatów wyzwalacza. Ten monitor wyzwalacza obsługuje domyślny typ aplikacji.

Łańcuch komendy przekazywany przez monitor wyzwalacza do systemu operacyjnego jest zbudowany w następujący sposób:

1. *ApplId* z odpowiedniej definicji PROCESS (jeśli została utworzona)
2. Struktura MQTMC2 ujęta w znaki podwójnego cudzysłowu.
3. *EnvData* z odpowiedniej definicji PROCESS (jeśli została utworzona)

gdzie *ApplId* to nazwa programu, który ma być uruchamiany w takiej postaci, w której jest wprowadzany w wierszu komend.

Przekazany parametr jest strukturą znaków MQTMC2 . Wywołano łańcuch komendy, który ma ten łańcuch, dokładnie tak, jak podano w podwójnych cudzysłowach, aby komenda systemowa zaakceptowała ją jako jeden parametr.

Monitor wyzwalacza nie widzi, czy istnieje inny komunikat w kolejce inicjujący do czasu zakończenia uruchomionego przez niego aplikacji. Jeśli aplikacja ma dużo przetwarzania, monitor wyzwalacza może nie być w stanie nadążać za liczbą przybywających komunikatów wyzwalacza. Dostępne są dwie opcje:

- Czy uruchomiono więcej monitorów wyzwalaczy
- Uruchom uruchomione aplikacje w tle

Jeśli uruchomionych jest więcej monitorów wyzwalacza, można sterować maksymalną liczbą aplikacji, które mogą być uruchamiane w dowolnym momencie. W przypadku uruchamiania aplikacji w tle nie ma ograniczeń narzuconych przez produkt WebSphere MQ w odniesieniu do liczby aplikacji, które mogą być uruchamiane.

Aby uruchomić uruchomionym aplikację w tle w systemach Windows , w polu *ApplId* , należy poprzedzić nazwę aplikacji komendą START. Na przykład:

```
START ?B AMQSECHA
```

Aby uruchomić uruchomionym aplikację w tle w systemach UNIX , należy umieścić & na końcu *EnvData* definicji PROCESS.

Uwaga: Jeśli ścieżka systemu Windows zawiera spacje jako część nazwy ścieżki, należy je ująć w znaki cudzysłowu ("). aby upewnić się, że jest ona obsługiwana jako pojedynczy argument. Na przykład `" C:\Program Files\Application Directory\Application.exe"`.

Poniżej znajduje się przykład łańcucha APPLICID, w którym nazwa pliku zawiera spacje jako część ścieżki:

```
START "" /B "C:\Program Files\Application Directory\Application.exe"
```

Składnia komendy START systemu Windows w przykładzie zawiera pusty łańcuch ujęty w podwójne cudzysłowy. Komenda START określa, że pierwszy argument w cudzysłowie będzie traktowany jako

tytuł nowej komendy. Aby upewnić się, że system Windows nie pomylił ścieżki aplikacji dla argumentu 'title', należy dodać do komendy łańcuch tytułu ujęty w podwójny cudzysłów przed nazwą aplikacji.

Dla klienta WebSphere MQ dostępne są następujące monitory wyzwalacza:

runmqtmc
Jest to takie samo, jak runmqtrm, z wyjątkiem tego, że łączy się z bibliotekami klienta MQI produktu WebSphere MQ .

Dla CICS
Monitor wyzwalacza amqltmc0 jest udostępniany dla programu CICS. Działa on w taki sam sposób, jak standardowy monitor wyzwalacza, runmqtrm, ale uruchamia się go w inny sposób i wyzwala transakcje CICS .

Ten temat dotyczy tylko systemów Windows, UNIXi Linux .

Jest on dostarczany jako program CICS . Zdefiniuj go przy użyciu 4-znakowej nazwy transakcji. Wprowadź 4-znakową nazwę, aby uruchomić monitor wyzwalacza. Używa domyślnego menedżera kolejek (o nazwie określonej w pliku qm.ini lub w produkcie WebSphere MQ dla Windows, rejestru) oraz SYSTEM.CICS.INITIATION.QUEUE.

Jeśli ma być używany inny menedżer kolejek lub inny menedżer kolejek, należy zbudować strukturę monitora wyzwalacza MQTMC2 . Wymaga to napisania programu za pomocą wywołania EXEC CICS START, ponieważ struktura jest zbyt długa, aby można ją było dodać jako parametr. Następnie przekaz strukturę MQTMC2 jako dane do żądania START dla monitora wyzwalacza.

Gdy używana jest struktura MQTMC2 , należy podać tylko parametry *StrucId*, *Version*, *QName*i *QMgrName* do monitora wyzwalacza, ponieważ nie odwołuje się do żadnych innych pól.

Komunikaty są odczytyane z kolejki inicjuj i służą do uruchamiania transakcji CICS przy użyciu programu EXEC CICS START, przy założeniu, że parametr APPL_TYPE w komunikacie wyzwalacza ma wartość MQAT_CICS. Odczyt komunikatów z kolejki inicjujący jest wykonywany w ramach elementu sterującego punktu synchronizacji CICS .

Komunikaty są generowane, gdy monitor jest uruchamiany i zatrzymywane, a także w przypadku wystąpienia błędu. Komunikaty te są wysyłane do przejściowej kolejki danych CSMT.

Poniżej przedstawiono dostępne wersje monitora wyzwalacza:

Wersja	**Użyj**
amqltmc0	TXSeries dla systemów AIX, HP-UXi Sun Solaris, wersja 5.1
amqltmc4	TXSeries dla systemu Windows, wersja 5.1
amqltmcc	Wersja skonsolidowana monitora wyzwalacza CICS

Jeśli potrzebny jest monitor wyzwalacza dla innych środowisk, napisz program, który może przetwarzać komunikaty wyzwalacza, które menedżer kolejek umieszcza w kolejkach inicjujący. Taki program powinien wykonać następujące działania:

1. Użyj wywołania MQGET, aby zaczekać na pojaw się komunikatu w kolejce inicjuj.
2. Sprawdź pola struktury MQTM komunikatu wyzwalacza, aby znaleźć nazwę aplikacji, która ma zostać uruchomiona, oraz środowisko, w którym jest ona uruchamiana.
3. Wydaj komendę uruchomienia specyficzną dla środowiska.
4. W razie potrzeby przekształć strukturę MQTM w strukturę MQTMC2 .
5. Przekaz strukturę MQTMC2 lub MQTM do uruchomionej aplikacji. Może to zawierać dane użytkownika.
6. Powiąż z kolejką aplikacji aplikację, która ma obsługiwać tę kolejkę. W tym celu należy nadawać nazwę obiektowi definicji procesu (jeśli jest to utworzone) w atrybucie *ProcessName* kolejki.

Więcej informacji na temat interfejsu monitora wyzwalacza można znaleźć w sekcji MQTMC2.

Właściwości komunikatów wyzwalacza

W poniższych tematach opisano niektóre inne właściwości komunikatów wyzwalacza.

Trwałość i priorytet komunikatów wyzwalacza

Komunikaty wyzwalacza nie są trwałe, ponieważ nie ma potrzeby, aby je tak było.

Jednak warunki generowania zdarzeń wyzwalających są trwałe, dlatego komunikaty wyzwalacza są generowane zawsze wtedy, gdy warunki te są spełnione. Jeśli komunikat wyzwalacza zostanie utracony, dalsze istnienie komunikatu aplikacji w kolejce aplikacji gwarantuje, że menedżer kolejek wygeneruje komunikat wyzwalający, gdy tylko spełnione zostaną wszystkie warunki.

Jeśli jednostka pracy jest wycofana, wszystkie wygenerowane przez niego komunikaty wyzwalacza są zawsze dostarczane.

Komunikaty wyzwalacza przyjmują domyślny priorytet kolejki inicjujący.

Komunikaty restartu i wyzwalacza menedżera kolejek

Po zrestartowaniu menedżera kolejek po kolejnym otwarciu kolejki inicjuj. dla danych wejściowych, komunikat wyzwalacza może zostać umieszczony w tej kolejce inicjuj, jeśli kolejka aplikacji powiązana z nią zawiera komunikaty i jest zdefiniowana dla wyzwalania.

Wyzwalanie komunikatów i wprowadzanie zmian w atrybutach obiektów

Komunikaty wyzwalacza są tworzone zgodnie z wartościami atrybutów wyzwalacza, które są używane w czasie zdarzenia wyzwalającego.

Jeśli komunikat wyzwalacza nie jest dostępny dla monitora wyzwalacza do czasu późniejszego (ponieważ komunikat, który spowodował jego wygenerowanie, został umieszczony w jednostce pracy), wszystkie zmiany atrybutów wyzwalacza w międzyczasie nie mają wpływu na komunikat wyzwalacza. W szczególności wyłączenie wyzwalania nie zapobiega udostępnianiu komunikatu wyzwalacza po jego utworzeniu. Ponadto kolejka aplikacji może nie istnieć w momencie, gdy komunikat wyzwalacza jest udostępniony.

Format komunikatów wyzwalacza

Format komunikatu wyzwalacza jest definiowany przez strukturę MQTM.

Ma ona następujące pola, które menedżer kolejek wypełnia podczas tworzenia komunikatu wyzwalacza, używając informacji w definicjach obiektów kolejki aplikacji i procesu powiązanego z tą kolejką:

StrucId
Identyfikator struktury.

Version
Wersja struktury.

QName
Nazwa kolejki aplikacji, w której wystąpiło zdarzenie wyzwalające. Gdy menedżer kolejek tworzy komunikat wyzwalacza, wypełni to pole za pomocą atrybutu *QName* w kolejce aplikacji.

ProcessName
Nazwa obiektu definicji procesu, który jest powiązany z kolejką aplikacji. Gdy menedżer kolejek tworzy komunikat wyzwalacza, wypełni to pole za pomocą atrybutu *ProcessName* w kolejce aplikacji.

TriggerData
Pole w formacie wolnoformatowym używane przez monitor wyzwalacza. Gdy menedżer kolejek tworzy komunikat wyzwalacza, wypełni to pole za pomocą atrybutu *TriggerData* w kolejce aplikacji. W każdym produkcie WebSphere MQ z wyjątkiem produktu WebSphere MQ for z/OSto pole może zostać użyte do określenia nazwy kanału, który ma zostać wyzwolony.

ApplType
Typ aplikacji, która ma zostać uruchomiona przez monitor wyzwalacza. Gdy menedżer kolejek tworzy komunikat wyzwalacza, wypełni to pole za pomocą atrybutu *ApplType* obiektu definicji procesu określonego w produkcie *ProcessName*.

ApplId
Łańcuch znaków identyfikujący aplikację, która ma zostać uruchomiona przez monitor wyzwalacza. Gdy menedżer kolejek tworzy komunikat wyzwalacza, wypełni to pole za pomocą atrybutu *ApplId* obiektu definicji procesu określonego w produkcie *ProcessName*. Jeśli używany jest monitor wyzwalacza CKTI lub CSQQTRMN dostarczany przez produkt WebSphere MQ for z/OS, atrybut *ApplId* obiektu definicji procesu jest identyfikatorem transakcji CICS lub IMS .

EnvData
Pole znakowe zawierające dane związane ze środowiskiem do użycia przez monitor wyzwalacza. Gdy menedżer kolejek tworzy komunikat wyzwalacza, wypełni to pole za pomocą atrybutu *EnvData* obiektu definicji procesu określonego w produkcie *ProcessName*. W przypadku monitorów wyzwalaczy dostarczonych przez produkt WebSphere MQ for z/OS(CKTI lub CSQQTRMN) nie należy używać tego pola, ale inne monitory wyzwalacza mogą z niego korzystać.

UserData
Pole znakowe zawierające dane użytkownika przeznaczone do użycia przez monitor wyzwalacza. Gdy menedżer kolejek tworzy komunikat wyzwalacza, wypełni to pole za pomocą atrybutu *UserData* obiektu definicji procesu określonego w produkcie *ProcessName*. To pole może być użyte do określenia nazwy kanału, który ma zostać wyzwolony.

W programie MQTMznajduje się pełny opis struktury komunikatu wyzwalacza.

Gdy wyzwalanie nie działa

Program nie jest wyzwalany, jeśli monitor wyzwalacza nie może uruchomić programu lub menedżer kolejek nie może dostarczyć komunikatu wyzwalacza. Na przykład wartość applid w obiekcie procesu musi określać, że program ma być uruchomiony w tle. W przeciwnym razie monitor wyzwalacza nie może uruchomić programu.

Jeśli komunikat wyzwalacza został utworzony, ale nie można go umieścić w kolejce inicjuj (na przykład, ponieważ kolejka jest pełna lub długość komunikatu wyzwalacza jest większa niż maksymalna długość komunikatu określona dla kolejki inicjuj), komunikat wyzwalacza jest umieszczany w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Jeśli operacja put dla kolejki niedostarczonych komunikatów nie może zostać zakończona pomyślnie, komunikat wyzwalacza jest odrzucany, a komunikat ostrzegawczy jest wysyłany do konsoli z/OS lub do operatora systemu lub jest umieszczany w protokole błędów.

Umieszczenie komunikatu wyzwalacza w kolejce niedostarczonych komunikatów może spowodować wygenerowanie komunikatu wyzwalacza dla tej kolejki. Ten drugi komunikat wyzwalacza jest odrzucany, jeśli do kolejki niedostarczonych komunikatów zostanie dodany komunikat.

Jeśli program jest wyzwalany pomyślnie, ale kończy działanie przed odebraniem komunikatu z kolejki, należy użyć narzędzia śledzenia (na przykład CICS AUXTRACE, jeśli program działa w systemie CICS), aby znaleźć przyczynę niepowodzenia.

Praca z interfejsem MQI i klastrami

Istnieją specjalne opcje dotyczące wywołań i kodów powrotu, które odnoszą się do technologii klastrowej.

Użyj poniższych odsyłaczy, aby dowiedzieć się więcej na temat opcji dostępnych w wywołaniach i kodach powrotu do użycia z klastrami:

- “MQOPEN i klastry” na stronie 356
- “MQPUT, MQPUT1 i klastry” na stronie 357
- “MQINQ i klastry” na stronie 358
- “MQSET i klastry” na stronie 358
- “Kody powrotu” na stronie 359

Pojęcia pokrewne

“Interfejs kolejki komunikatów-przegląd” na stronie 199
Informacje na temat komponentów interfejsu kolejek komunikatów (Message Queue Interface-MQI).

“Nawiąże połączenie z menedżerem kolejek i odłączenie go od niego” na stronie 211
Aby można było korzystać z usług programistycznych produktu WebSphere MQ , program musi mieć połączenie z menedżerem kolejek. Informacje zawarte w tej sekcji umożliwiają zapoznanie się z informacjami na temat nawiązywania połączenia z menedżerem kolejek i z jego rozłączeniem.

“Otwieranie i zamykanie obiektów” na stronie 220
Te informacje udostępniają wgląd w otwieranie i zamykanie obiektów WebSphere MQ .

“Umieszczanie komunikatów w kolejce” na stronie 231
Te informacje umożliwiają zapoznanie się z informacjami na temat umieszczania komunikatów w kolejce.

“Pobieranie komunikatów z kolejki” na stronie 246
Ta sekcja zawiera informacje na temat pobierania komunikatów z kolejki.

“Sprawdzanie i ustawianie atrybutów obiektu” na stronie 328
Atrybuty to właściwości, które definiują parametry obiektu WebSphere MQ .

“Zatwierdzanie i wycofywanie jednostek pracy” na stronie 331
W tej sekcji opisano sposób zatwierdzania i tworzenia kopii zapasowych wszystkich możliwych do odtworzenia operacji get i put, które wystąpiły w jednostce pracy.

“Uruchamianie aplikacji produktu IBM WebSphere MQ przy użyciu wyzwalaczy” na stronie 338
Informacje na temat wyzwalaczy i sposobów uruchamiania aplikacji IBM WebSphere MQ przy użyciu wyzwalaczy.

MQOPEN i klastry

Kolejka, do której komunikat jest umieszczany lub odczytany, gdy kolejka klastra jest otwierana, zależy od wywołania MQOPEN .

Wybieranie kolejki docelowej

Jeśli w deskryptorze obiektu nie zostanie podana nazwa menedżera kolejek, MQOD, menedżer kolejek wybiera menedżera kolejek, do którego ma zostać wysłany komunikat. Jeśli w deskryptorze obiektu zostanie podana nazwa menedżera kolejek, komunikaty będą zawsze wysyłane do wybranego menedżera kolejek.

Jeśli menedżer kolejek wybiera docelowy menedżer kolejek, wybór zależy od opcji powiązania, MQOO_BIND_* oraz, jeśli istnieje kolejka lokalna. Jeśli istnieje lokalna instancja kolejki, jest ona zawsze otwierana w preferencjach do zdalnej instancji, chyba że atrybut CLWLUSEQ jest ustawiony na wartość ANY. W przeciwnym razie wybór zależy od opcji powiązania. W przypadku korzystania z grup komunikatów z klastrami należy określić wartość MQOO_BIND_ON_OPEN lub MQOO_BIND_ON_GROUP , aby zapewnić, że wszystkie komunikaty w grupie są przetwarzane w tym samym miejscu docelowym.

Jeśli menedżer kolejek wybiera docelowy menedżer kolejek, robi to w sposób zaokrąglany, korzystając z algorytmu zarządzania obciążeniem. Patrz sekcja Równoważenie obciążenia.

MQOO_BIND_ON_OPEN
Opcja MQOO_BIND_ON_OPEN w wywołaniu MQOPEN określa, że docelowy menedżer kolejek ma zostać naprawiony. Opcji MQOO_BIND_ON_OPEN należy użyć, jeśli w klastrze istnieje wiele instancji tej samej kolejki. Wszystkie komunikaty umieszczone w kolejce, określające uchwyt obiektu zwróconego z wywołania MQOPEN , są kierowane do tego samego menedżera kolejek.

- Jeśli komunikaty mają powinowactwa, należy użyć opcji MQOO_BIND_ON_OPEN . Na przykład, jeśli zadanie wsadowe komunikatów ma być przetwarzane przez ten sam menedżer kolejek, należy określić wartość MQOO_BIND_ON_OPEN po otwarciu kolejki. Program IBM WebSphere MQ naprawia menedżera kolejek i trasę, która ma być podejmowana przez wszystkie komunikaty umieszczone w tej kolejce.
- Jeśli podano opcję MQOO_BIND_ON_OPEN , należy ponownie otworzyć kolejkę dla nowej instancji kolejki, która ma zostać wybrana.

MQOO_BIND_NOT_FIXED
Opcja MQOO_BIND_NOT_FIXED w wywołaniu MQOPEN określa, że docelowy menedżer kolejek nie został naprawiony. Komunikaty zapisywane w kolejce, określające uchwyt obiektu zwróconego z wywołania MQOPEN , są kierowane do menedżera kolejek w czasie MQPUT na podstawie komunikatów typu message-by-message. Opcji MQOO_BIND_NOT_FIXED należy używać, jeśli nie ma potrzeby wymuszania zapisywania wszystkich komunikatów w tym samym miejscu docelowym.

- Nie należy jednocześnie podawać opcji MQOO_BIND_NOT_FIXED i MQMF_SEGMENTATION_ALLOWED . W takim przypadku segmenty komunikatu mogą być dostarczane do różnych menedżerów kolejek, rozrzuconych po całym klastrze.

MQOO_BIND_ON_GROUP
Umożliwia aplikacji żądanie, aby grupa komunikatów została przydzielona do tej samej instancji docelowej. Ta opcja jest poprawna tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja jest określona dla kolejki, która nie jest kolejką klastra, opcja jest ignorowana.

- Grupy są kierowane do jednego miejsca docelowego tylko wtedy, gdy w tabeli MQPUT określono parametr MQPMO_LOGICAL_ORDER. Jeśli określono atrybut MQOO_BIND_ON_GROUP, ale komunikat nie jest częścią grupy, zamiast niego używane jest zachowanie BIND_NOT_FIXED.

MQOO_BIND_AS_Q_DEF
Jeśli użytkownik nie poda opcji MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED lub MQOO_BIND_ON_GROUP, domyślną opcją będzie MQOO_BIND_AS_Q_DEF. Użycie produktu MQOO_BIND_AS_Q_DEF powoduje, że powiązanie jest używane dla uchwytu kolejki, który ma zostać użyty z atrybutu kolejki DefBind .

Istotność opcji produktu MQOPEN

Opcje MQOPEN MQOO_BROWSE , MQOO_INPUT_*lub MQOO_SET wymagają, aby lokalna instancja kolejki klastra dla MQOPEN powiodła się.

The MQOPEN options MQOO_OUTPUT, MQOO_BIND_*, or MQOO_INQUIRE do not require a local instance of the cluster queue to succeed.

Rozstrzygnięta nazwa menedżera kolejek

Gdy nazwa menedżera kolejek jest rozstrzygana w czasie MQOPEN , rozstrzygnięta nazwa jest zwracana do aplikacji. Jeśli aplikacja podejmie próbę użycia tej nazwy w kolejnym wywołaniu programu MQOPEN , może się okazać, że nie ma uprawnień dostępu do nazwy.

MQPUT, MQPUT1 i klastry

Jeśli MQOO_BIND_NOT_FIXED jest określone na MQOPEN , procedury zarządzania obciążeniem wybierają miejsce docelowe MQPUT lub MQPUT1 .

Jeśli w wywołaniu MQOPEN określono wartość MQOO_BIND_NOT_FIXED , każde kolejne wywołanie programu MQPUT wywołuje procedurę zarządzania obciążeniem w celu określenia menedżera kolejek, do którego ma zostać wysłany komunikat. Miejsce docelowe i trasa do podjęcia są wybierane na podstawie komunikatu. Miejsce docelowe i trasa mogą ulec zmianie po umieszczniu komunikatu w razie zmiany warunków w sieci. Wywołanie MQPUT1 zawsze działa tak, jakby MQOO_BIND_NOT_FIXED było aktywne, to znaczy zawsze wywołuje procedurę zarządzania obciążeniem.

Po wybraniu przez procedurę zarządzania obciążeniem menedżera kolejek lokalny menedżer kolejek kończy operację put. Komunikat może być umieszczony w różnych kolejkach:

1. Jeśli miejscem docelowym jest lokalna instancja kolejki, komunikat jest umieszczany w kolejce lokalnej.
2. Jeśli miejsce docelowe jest menedżerem kolejek w klastrze, komunikat jest umieszczany w kolejce transmisji klastra.
3. Jeśli miejsce docelowe jest menedżerem kolejek poza klastrem, komunikat jest umieszczany w kolejce transmisji o tej samej nazwie, co docelowy menedżer kolejek.

Jeśli w wywołaniu MQOPEN określono wartość MQOO_BIND_ON_OPEN , wywołania MQPUT nie wywołują procedury zarządzania obciążeniem, ponieważ miejsce docelowe i trasa zostały już wybrane.

MQINQ i klastry

Wybór kolejki klastra zależy od opcji, które łączą się z produktem MQOO_INQUIRE.

Zanim możliwe będzie sprawdzenie kolejki, należy otworzyć ją za pomocą wywołania MQOPEN i określić MQOO_INQUIRE.

Aby uzyskać informacje na temat kolejki klastra, należy użyć wywołania MQOPEN i połączyć inne opcje z programem MQOO_INQUIRE. Atrybuty, które można sprawdzić, zależą od tego, czy istnieje lokalna instancja kolejki klastra, oraz od tego, w jaki sposób zostanie otwarta kolejka:

- Łączenie produktów MQOO_BROWSE, MQOO_INPUT_*lub MQOO_SET z produktem MQOO_INQUIRE wymaga lokalnej instancji kolejki klastra, aby możliwe było pomyślne wykonanie tej operacji. W tym przypadku można zapytać o wszystkie atrybuty, które są poprawne dla kolejek lokalnych.
- Łączenie produktu MQOO_OUTPUT z produktem MQOO_INQUIREi określenie żadnej z poprzednich opcji powoduje, że instancja otwarta jest:
  - Instancja w lokalnym menedżerze kolejek, jeśli istnieje. W tym przypadku można zapytać o wszystkie atrybuty, które są poprawne dla kolejek lokalnych.
  - Instancja w innym miejscu w klastrze, jeśli nie istnieje lokalna instancja menedżera kolejek. W tym przypadku można uzyskać dostęp tylko do następujących atrybutów. W tym przypadku atrybut QType ma wartość MQQT_CLUSTER .

    - DefBind
    - DefPersistence
    - DefPriority
    - InhibitPut
    - QDesc
    - Nazwa QName
    - QTYPE

Aby uzyskać informacje na temat atrybutu DefBind w kolejce klastra, należy użyć wywołania MQINQ z selektorem MQIA_DEF_BIND. Zwrócona wartość to MQBND_BIND_ON_OPEN lub MQBND_BIND_NOT_FIXEDalbo MQBND_BIND_ON_GROUP. W przypadku używania grup z klastrami należy określić wartość MQBND_BIND_ON_OPEN lub MQBND_BIND_ON_GROUP .

Aby uzyskać informacje na temat atrybutów CLUSTER i CLUSNL lokalnej instancji kolejki, należy użyć wywołania MQINQ z selektorem MQCA_CLUSTER_NAME lub selektorem MQCA_CLUSTER_NAMELIST.

Uwaga: Jeśli kolejka klastra zostanie otwarta bez konieczności naprawiać kolejki, z którą powiązana jest MQOPEN , kolejne wywołania programu MQINQ mogą zapytać o różne instancje kolejki klastra.

Pojęcia pokrewne

“Opcja MQOPEN dla kolejki klastra” na stronie 226

Powiązanie używane dla uchwytu kolejki jest pobierane z atrybutu kolejki *DefBind* , który może mieć wartość MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXEDlub MQBND_BIND_ON_GROUP.

MQSET i klastry

Opcja MQOPEN (opcja MQOO_SET) wymaga, aby lokalna instancja kolejki klastra dla MQSET powiodła się.

Nie można użyć wywołania MQSET do ustawienia atrybutów kolejki w innym miejscu w klastrze.

Można otworzyć lokalny alias lub kolejkę zdalną zdefiniowaną za pomocą atrybutu klastra, a następnie użyć wywołania MQSET . Istnieje możliwość ustawienia atrybutów lokalnego aliasu lub kolejki zdalnej. Nie ma znaczenia, czy kolejka docelowa jest kolejką klastra zdefiniowaną w innym menedżerze kolejek.

Kody powrotu

Kody powrotu specyficzne dla klastrów

MQRC_CLUSTER_EXIT_ERROR (2266 X'8DA')

Wywołano komendę MQOPEN, MQPUTlub MQPUT1 , aby otworzyć kolejkę klastra lub umieścić na niej komunikat. Wyjście obciążenia klastra, zdefiniowane za pomocą atrybutu ClusterWorkloadExit menedżera kolejek, nieoczekiwanie kończy się niepowodzeniem lub nie reaguje w czasie.

A message is written to the system log on WebSphere MQ for z/OS giving more information about this error.

Kolejne wywołania funkcji MQOPEN, MQPUTi MQPUT1 dla tego uchwytu kolejki są przetwarzane tak, jakby atrybut ClusterWorkloadExit był pusty.

MQRC_CLUSTER_EXIT_LOAD_ERROR (2267 X'8DB')

W systemie z/OSnie można załadować wyjścia obciążenia klastra.

Komunikat jest zapisywany w dzienniku systemowym, a przetwarzanie jest kontynuowane tak, jakby atrybut ClusterWorkloadExit był pusty.

Na platformach innych niż z/OSwywołanie funkcji MQCONN lub MQCONNX jest wykonywane w celu nawiązania połączenia z menedżerem kolejek. Wywołanie nie powiodło się, ponieważ nie można załadować wyjścia obciążenia klastra zdefiniowanego przez atrybut menedżera kolejek ClusterWorkload menedżera kolejek.

MQRC_CLUSTER_PUT_INHIBITED (2268 X'8DC')

Wywołanie MQOPEN z opcjami MQOO_OUTPUT i MQOO_BIND_ON_OPEN w efekcie jest wydawane dla kolejki klastra. Wszystkie instancje kolejki w klastrze są obecnie wstrzymane przez ustawienie atrybutu InhibitPut ustawionego na wartość MQQA_PUT_INHIBITED. Ponieważ nie są dostępne żadne instancje kolejek do odbierania komunikatów, wywołanie MQOPEN nie powiodło się.

Ten kod przyczyny pojawia się tylko wtedy, gdy spełnione są oba poniższe warunki:

- Nie istnieje lokalna instancja kolejki. Jeśli istnieje instancja lokalna, wywołanie MQOPEN powiedzie się, nawet jeśli lokalna instancja jest zablokowana.
- Dla kolejki nie ma wyjścia obciążenia klastra lub jest wyjście obciążenia klastra, ale nie wybiera instancji kolejki. (Jeśli wyjście obciążenia klastra wybierze instancję kolejki, wywołanie MQOPEN zakończy się powodzeniem, nawet jeśli ta instancja jest zablokowana).

Jeśli w wywołaniu MQOPEN zostanie podana opcja MQOO_BIND_NOT_FIXED , wywołanie może zakończyć się powodzeniem, nawet jeśli wszystkie kolejki w klastrze zostaną zablokowane. Jednak kolejne wywołanie programu MQPUT może się nie powieść, jeśli wszystkie kolejki są nadal wstrzymane w czasie wywołania.

MQRC_CLUSTER_RESOLUTION_ERROR (2189 X'88D')

1. Wywołano komendę MQOPEN, MQPUTlub MQPUT1 , aby otworzyć kolejkę klastra lub umieścić na niej komunikat. Nie można poprawnie rozstrzygnąć definicji kolejki, ponieważ wymagana jest odpowiedź od menedżera kolejek pełnego repozytorium, ale żadna nie jest dostępna.
2. Wywołanie funkcji MQOPEN, MQPUT, MQPUT1 lub MQSUB jest wykonywane dla obiektu tematu z określeniem PUBSCOPE(ALL) lub SUBSCOPE(ALL). Nie można poprawnie rozstrzygnąć definicji tematu klastra, ponieważ wymagana jest odpowiedź z menedżera kolejek pełnego repozytorium, ale żadna z nich nie jest dostępna.

MQRC_CLUSTER_RESOURCE_ERROR (2269 X'8DD')

Wywołanie MQOPEN, MQPUTlub MQPUT1 jest wykonywane dla kolejki klastra. Wystąpił błąd podczas próby użycia zasobu wymaganego do łączenia w klastry.

MQRC_NO_DESTINATIONS_AVAILABLE (2270 X'8DE')

Wywołano komendę MQPUT lub MQPUT1 w celu umieszczenia komunikatu w kolejce klastra. W czasie wywołania nie ma już żadnych instancji kolejki w klastrze. MQPUT nie powiodło się, a komunikat nie został wysłany.

Błąd może wystąpić, jeśli w wywołaniu MQOPEN , która otwiera kolejkę, zostanie podana wartość MQOO_BIND_NOT_FIXED , lub MQPUT1 zostanie użyta do umieszczenia komunikatu.

MQRC_STOPPED_BY_CLUSTER_EXIT (2188 X'88C')

Wywołano komendę MQOPEN, MQPUTlub MQPUT1 , aby otworzyć lub umieścić komunikat w kolejce klastra. Wyjście obciążenia klastra odrzuca wywołanie.

Pisanie aplikacji klienckich

Co należy wiedzieć, aby pisać aplikacje klienckie w produkcie WebSphere MQ.

Aplikacje mogą być budowane i uruchamiane w środowisku klienta WebSphere MQ . Aplikacja musi zostać zbudowana i dowiązana do używanego klienta MQI produktu WebSphere MQ . Sposób budowania i tworzenia powiązanych aplikacji różni się w zależności od używanej platformy i języka programowania. Informacje na temat budowania aplikacji klienckich zawiera sekcja “Budowanie aplikacji dla klientów MQI produktu WebSphere MQ” na stronie 366.

Aplikację WebSphere MQ można uruchomić zarówno w pełnym środowisku WebSphere MQ , jak i w środowisku klienta MQI produktu WebSphere MQ bez konieczności zmiany kodu, pod warunkiem że spełnione są określone warunki. Więcej informacji na temat uruchamiania aplikacji w środowisku klienckim produktu WebSphere MQ zawiera sekcja “Uruchamianie aplikacji w środowisku klienta MQI produktu IBM WebSphere MQ” na stronie 368.

Jeśli do zapisu aplikacji w środowisku klienta MQI produktu WebSphere MQ używany jest interfejs kolejki komunikatów (MQI), to podczas wywołania MQI konieczne jest wprowadzenie dodatkowych elementów sterujących w celu upewniania się, że przetwarzanie aplikacji WebSphere MQ nie jest zakłócane. Więcej informacji na temat tych elementów sterujących zawiera sekcja “Korzystanie z interfejsu kolejki komunikatów (MQI) w aplikacji klienckiej” na stronie 361.

Poniższe tematy zawierają informacje na temat przygotowywania i uruchamiania innych typów aplikacji jako aplikacji klienckich:

- “Przygotowywanie i uruchamianie aplikacji CICS i Tuxedo” na stronie 380
- “Przygotowywanie i uruchamianie aplikacji serwera Microsoft Transaction Server” na stronie 41
- “Przygotowywanie i uruchamianie aplikacji JMS produktu WebSphere MQ” na stronie 383

Pojęcia pokrewne

“Pojęcia związane z projektowaniem aplikacji” na stronie 8
Do pisania aplikacji IBM WebSphere MQ można użyć wyboru języków proceduralnych lub obiektowych. Odsyłacze w tym temacie można znaleźć w informacjach dotyczących pojęć związanych z produktem IBM WebSphere MQ , które są przydatne dla programistów aplikacji.

“Wybór języka programowania do użycia” na stronie 80
W tej sekcji znajdują się informacje na temat języków programowania i środowisk obsługiwanych przez produkt IBM WebSphere MQ, a także niektóre zagadnienia dotyczące ich używania.

“Projektowanie aplikacji produktu IBM WebSphere MQ” na stronie 91
Po zdecydowaniu się, w jaki sposób aplikacje mogą korzystać z platform i środowisk, które są dostępne dla użytkownika, należy zdecydować, w jaki sposób korzystać z funkcji oferowanych przez produkt WebSphere MQ.

“Przykładowe programy produktu WebSphere MQ” na stronie 98

Ta kolekcja tematów zawiera informacje na temat przykładowych programów WebSphere MQ na różnych platformach.

"Pisanie aplikacji kolejkowania" na stronie 199
Ta sekcja zawiera informacje na temat pisania aplikacji kolejkowania, łączenia się i rozłączania z menedżerem kolejek, publikowania/subskrypcji oraz obiektów otwierających i zamykających.

"Korzystanie z usług Web Service w produkcie WebSphere MQ" na stronie 972
Aplikacje produktu IBM WebSphere MQ dla usług Web Services można tworzyć przy użyciu transportu IBM WebSphere MQ dla protokołu SOAP lub mostu IBM WebSphere MQ dla protokołu HTTP.

"Zapisywanie aplikacji publikowania/subskrypcji" na stronie 285
Rozpocznij pisanie aplikacji WebSphere MQ publikowania/subskrypcji.

"Budowanie aplikacji IBM WebSphere MQ" na stronie 439
Ta sekcja zawiera informacje na temat budowania aplikacji produktu IBM WebSphere MQ na różnych platformach.

"Obsługa błędów programu" na stronie 561
Te informacje wyjaśniają błędy związane z wywołaniami MQI aplikacji, gdy wywołuje połączenie, lub gdy jego komunikat jest dostarczany do miejsca docelowego.

Korzystanie z interfejsu kolejki komunikatów (MQI) w aplikacji klienckiej

Ta kolekcja tematów uwzględnia różnice między zapisaniem aplikacji WebSphere MQ w celu uruchomienia w środowisku klienta MQI produktu WebSphere MQ i uruchamianiem w pełnym środowisku menedżera kolejek produktu WebSphere MQ .

Podczas projektowania aplikacji należy wziąć pod uwagę elementy sterujące, które mają zostać narzucone podczas wywołania MQI, aby upewnić się, że przetwarzanie aplikacji WebSphere MQ nie zostało zakłócone.

Ograniczanie wielkości komunikatu w aplikacji klienckiej

Menedżer kolejek ma maksymalną długość komunikatu, ale maksymalna wielkość komunikatu, jaki można przesłać z aplikacji klienckiej, jest ograniczona przez definicję kanału.

Atrybut maksymalnej długości komunikatu (MaxMsgLength) menedżera kolejek to maksymalna długość komunikatu, który może być obsługiwany przez tego menedżera kolejek.

Na platformach innych niż z/OSmożna zwiększyć atrybut maksymalnej długości komunikatu menedżera kolejek. Szczegóły znajdują się w pliku ALTER QMGR.

Za pomocą wywołania MQINQ można znaleźć wartość parametru MaxMsg(Maksymalna długość komunikatu) dla menedżera kolejek.

Jeśli atrybut długości MaxMsgjest zmieniany, nie jest wykonywane sprawdzanie, czy nie istnieją jeszcze kolejki, a nawet komunikaty, których długość jest większa niż nowa wartość. Po zmianie tego atrybutu należy zrestartować aplikacje i kanały w celu upewnia się, że zmiana została uwzględniona. Nie jest możliwe wygenerowanie nowych komunikatów, które przekraczają długość MaxMsgmenedżera kolejek lub kolejki (jeśli segmentacja menedżera kolejek nie jest dozwolona).

Maksymalna długość komunikatu w definicji kanału ogranicza wielkość komunikatu, który można przesłać wzdłuż połączenia klienckiego. Jeśli aplikacja WebSphere MQ próbuje użyć wywołania MQPUT lub wywołania MQGET z komunikatem większym niż ten komunikat, do aplikacji zwracany jest kod błędu. Parametr maksymalnej wielkości komunikatu w definicji kanału nie ma wpływu na maksymalną wielkość komunikatu, która może zostać wykorzystana przy użyciu obiektu MQCB w połączeniu z klientem.

Wybieranie identyfikatora kodowanego zestawu znaków klienta lub serwera (CCSID)

Należy użyć lokalnego identyfikatora CCSID dla klienta. Menedżer kolejek wykonuje niezbędne konwersję. Użyj zmiennej środowiskowej MQCCSID, aby przesłonić identyfikator CCSID. Jeśli aplikacja wykonuje

wiele operacji PUTs, pola CCSID i pola kodowania deskryptora MQMD mogą zostać nadpisane po zakończeniu pierwszej operacji PUT.

Dane przekazywane przez interfejs MQI z aplikacji do kodu pośredniczącego klienta muszą znajdować się w lokalnym identyfikatorze CCSID, zakodowane dla klienta MQI produktu WebSphere MQ . Jeśli połączony menedżer kolejek wymaga konwersji danych do konwersji, to konwersja jest wykonywana przez kod obsługi klienta w menedżerze kolejek.

Klient Java w wersji V7może jednak wykonać konwersję, jeśli menedżer kolejek nie jest w stanie wykonać tej operacji. Więcej informacji znajduje się w sekcji "Klasy produktu WebSphere MQ dla połączeń klienckich Java" na stronie 685

W kodzie klienta założono, że dane znakowe przekraczające interfejs MQI w kliencie są w CCSID skonfigurowanym dla tej stacji roboczej. Jeśli ten identyfikator CCSID nie jest obsługiwany lub nie jest to wymagany identyfikator CCSID, można go przesłonić za pomocą zmiennej środowiskowej MQCCSID, używając jednej z następujących komend:

- W systemie Windows:

```
SET MQCCSID=850
```

- W systemach UNIX:

```
export MQCCSID=850
```

Jeśli ten parametr jest ustawiony w profilu, zakłada się, że wszystkie dane MQI znajdują się na stronie kodowej 850.

Uwaga: Założenie dotyczące strony kodowej 850 nie ma zastosowania do danych aplikacji w komunikacie.

Jeśli aplikacja wykonuje wiele operacji PUTs, które zawierają nagłówki WebSphere MQ po deskryptorze komunikatu (MQMD), należy pamiętać, że pola identyfikatora CCSID i kodowania deskryptora MQMD są nadpisywane po zakończeniu pierwszej operacji PUT.

Po pierwszym PUT, pola te zawierają wartość używaną przez połączonego menedżera kolejek w celu przekształcenia nagłówków WebSphere MQ . Upewnij się, że aplikacja zresetuje wartości do wymaganych przez nią wartości.

Użycie komendy MQINQ w aplikacji klienta

Niektóre wartości odpytywane za pomocą komendy MQINQ są modyfikowane przez kod klienta.

CCSID
jest ustawiony na identyfikator CCSID klienta, a nie identyfikator menedżera kolejek.

MaxMsgDługość
jest zmniejszony, jeśli jest ograniczony przez definicję kanału. Będzie to niższa z następujących wartości:

- Wartość zdefiniowana w definicji kolejki, lub
- Wartość zdefiniowana w definicji kanału

Więcej informacji na ten temat zawiera sekcja MQINQ.

Korzystanie z koordynacji punktów synchronizacji w aplikacji klienckiej

Aplikacja działająca na kliencie podstawowym może wydać komendę MQCMIT i MQBACK, ale zasięg elementu sterującego punktu synchronizacji jest ograniczony do zasobów MQI. Za pomocą zewnętrznego menedżera transakcji można korzystać z rozszerzonego klienta transakcyjnego.

W produkcie WebSphere MQjedną z ról menedżera kolejek jest sterowanie punktem synchronizacji w ramach aplikacji. Jeśli aplikacja działa na kliencie podstawowym produktu WebSphere MQ , może wydać komendę MQCMIT i MQBACK, ale zasięg elementu sterującego punktu synchronizacji jest

ograniczony do zasobów MQI. Komenda MQBEGIN produktu WebSphere MQ nie jest poprawna w podstawowym środowisku klienta.

Aplikacje działające w pełnym środowisku menedżera kolejek na serwerze mogą koordynować wiele zasobów (na przykład baz danych) za pośrednictwem monitora transakcji. Na serwerze można użyć monitora transakcji dostarczanego z produktami WebSphere MQ lub innego monitora transakcji, takiego jak CICS. Nie można użyć monitora transakcji z podstawową aplikacją kliencką.

Istnieje możliwość użycia zewnętrznego menedżera transakcji z rozszerzonym klientem transakcyjnym WebSphere MQ . Patrz Co to jest rozszerzony klient transakcyjny? .

Korzystanie z odczytu z wyprzedzeniem w aplikacji klienckiej

Można użyć odczytu z wyprzedzeniem na kliencie, aby zezwolić na wysyłanie nietrwałych komunikatów do klienta bez konieczności żądania przez aplikację kliencką komunikatów.

Gdy klient wymaga komunikatu z serwera, wysyła żądanie do serwera. Wysyła on osobne żądanie dla każdego z komunikatów, które konsumuje. Aby zwiększyć wydajność klientów korzystających z nietrwałych komunikatów przez uniknięcie konieczności wysyłania tych komunikatów żądań, klient może zostać skonfigurowany do używania odczytu z wyprzedzeniem. Odczyt z wyprzedzeniem umożliwia wysyłanie komunikatów do klienta bez konieczności żądania ich żądania.

Użycie odczytu z wyprzedzeniem może zwiększyć wydajność podczas korzystania z nietrwałych komunikatów z aplikacji klienckiej. Ta poprawa wydajności jest dostępna zarówno dla aplikacji MQI, jak i JMS. Aplikacje klienckie korzystające z wykorzystania MQGET lub asynchronicznego korzystają z ulepszeń wydajności podczas korzystania z nietrwałych komunikatów.

Po wywołaniu komendy MQOPEN z produktem MQOO_READ_AHEAD, klient WebSphere MQ umożliwia tylko odczyt z wyprzedzeniem, jeśli spełnione są określone warunki. Są one następujące:

- Zarówno klient, jak i menedżer kolejek zdalnych muszą być w wersji WebSphere MQ 7 lub nowszej.
- Aplikacja kliencka musi być skompilowana i powiązana z wątkami bibliotek klienta MQI produktu WebSphere MQ .
- Kanał klienta musi używać protokołu TCP/IP.
- Ustawienie SharingConversations (SHARECNV) kanału musi mieć wartość niezerową w definicji kanału zarówno klienta, jak i serwera.

Gdy opcja odczytu z wyprzedzeniem jest włączona, komunikaty są wysyłane do buforu pamięci na kliencie o nazwie bufor odczytu z wyprzedzeniem. Klient ma bufor odczytu z wyprzedzeniem dla każdej kolejki, która została otwarta z włączoną obsługą odczytu z wyprzedzeniem. Komunikaty w buforze odczytu z wyprzedzeniem nie są utrwalane. Klient okresowo aktualizuje serwer, podając informacje o ilości zużywanej przez niego danych.

Nie wszystkie projekty aplikacji klienckich są odpowiednie do korzystania z odczytu z wyprzedzeniem, ponieważ nie wszystkie opcje są obsługiwane. Niektóre opcje są wymagane, aby były spójne między wywołaniami MQGET, gdy opcja odczytu z wyprzedzeniem jest włączona. Jeśli klient zmienia kryteria wyboru między wywołaniami MQGET, komunikaty zapisywane w buforze odczytu z wyprzedzeniem pozostają bez zmian w buforze odczytu z wyprzedzeniem klienta. Więcej informacji na ten temat zawiera sekcja “Zwiększanie wydajności nietrwałych komunikatów” na stronie 265.

Konfiguracja odczytu z wyprzedzeniem jest kontrolowana przez trzy atrybuty: MaximumSize, PurgeTimei UpdatePercentage, które są określone w sekcji MessageBuffer pliku konfiguracyjnego klienta WebSphere MQ .

Korzystanie z funkcji asynchronicznej umieszczonej w aplikacji klienckiej

Za pomocą komendy put asynchroniczny aplikacja może umieścić komunikat w kolejce bez oczekiwania na odpowiedź z menedżera kolejek. W niektórych sytuacjach można użyć tego celu w celu zwiększenia wydajności przesyłania komunikatów.

Zwykle, gdy aplikacja umieszcza komunikat lub komunikaty w kolejce za pomocą operacji MQPUT lub MQPUT1, aplikacja musi czekać na potwierdzenie przez menedżer kolejek, że przetworzyła żądanie

MQI. Wydajność przesyłania komunikatów można zwiększyć, w szczególności w przypadku aplikacji, które korzystają z powiązań klienta, oraz aplikacji, które umieszczają dużą liczbę małych komunikatów w kolejce, wybierając zamiast asynchronicznie umieszczania komunikatów. Gdy aplikacja umieszcza komunikat asynchronicznie, menedżer kolejek nie zwraca powodzenia lub niepowodzenia każdego wywołania, ale można okresowo sprawdzać, czy nie wystąpiły błędy.

Aby umieścić komunikat w kolejce asynchronicznie, należy użyć opcji MQPMO_ASYNC_RESPONSE w polu *Options* w strukturze MQPMO.

Jeśli komunikat nie jest zakwalifikowany do asynchronicznego umieszczania, jest on umieszczany w kolejce synchronicznie.

W przypadku żądania asynchronicznej odpowiedzi put dla operacji MQPUT lub MQPUT1, CompCode i przyczyna MQCC_OK i MQRC_NONE nie muszą oznaczać, że komunikat został pomyślnie umieszczony w kolejce. Mimo że powodzenie lub niepowodzenie każdej pojedynczej wywołania MQPUT lub MQPUT1 może nie zostać zwrócone natychmiast, pierwszy błąd, który wystąpił w wywołaniu asynchronicznym, można określić później za pomocą wywołania MQSTAT.

Więcej informacji na temat opcji MQPMO_ASYNC_RESPONSE zawiera sekcja Opcje MQPMO.

Przykładowy program Asynchroniczny Put demonstruje niektóre z dostępnych funkcji. Szczegółowe informacje na temat funkcji i projektu programu oraz sposobu jego uruchamiania zawiera sekcja “Przykładowy program do umieszczania w pamięci asynchronicznej” na stronie 119.

Korzystanie z współużytkowania konwersacji w aplikacji klienckiej

W środowisku, w którym dozwolone jest współużytkowanie konwersacji, konwersacje mogą współużytkować instancję kanału MQI.

Współużytkowanie konwersacji jest kontrolowane przez dwa pola o nazwie SharingConversations, z których jeden jest częścią struktury definicji kanału (MQCD), a jeden z nich jest częścią struktury parametru wyjścia kanału (MQCXP). Pole SharingConversations w tabeli MQCD jest liczbą całkowitą, określającą maksymalną liczbę konwersacji, które mogą współużytkować instancję kanału powiązaną z kanałem. Pole SharingConversations w tabeli MQCXP jest wartością boolowką wskazującą, czy instancja kanału jest obecnie współużytkowana.

W środowisku, w którym współużytkowanie konwersacji nie jest dozwolone, nowe połączenia klienckie określające identyczne tabele MQCD nie będą współużytkować instancji kanału.

Nowe połączenie aplikacji klienckiej będzie współużytkować instancję kanału, gdy spełnione są następujące warunki:

- Zarówno połączenie klienta, jak i połączenia z serwerem kończą się instancją kanału, są skonfigurowane do współużytkowania konwersacji, a te wartości nie są nadpisywane przez wyjścia kanału.
- Wartość MQCD połączenia klienckiego (podana w wywołaniu klienta MQCONNX lub z tabeli definicji kanału klienta (CCDT)) jest dokładnie zgodna z wartością MQCD połączenia klienta podaną w wywołaniu MQCONNX klienta lub z tabeli definicji kanału klienta, gdy istniejąca instancja kanału została najpierw utworzona. Należy zauważyć, że oryginalna tabela MQCD mogła zostać później zmieniona przez wyjścia lub negocjację kanału, ale zgodność ta jest porównana z wartością dostarczonej do systemu klienta, zanim zmiany te zostały wprowadzone.
- Limit współużytkowania konwersacji po stronie serwera nie został przekroczony.

Jeśli nowe połączenie aplikacji klienckiej jest zgodne z kryteriami umożliwiania współużytkowania instancji kanału z innymi konwersacjami, ta decyzja jest podejmowana przed wywołaniem jakichkolwiek wyjść w tej konwersacji. Wyjścia w takiej konwersacji nie mogą zmienić faktu, że współużytkuje instancję kanału z innymi konwersacjami. Jeśli nie ma istniejących instancji kanałów zgodnych z nową definicją kanału, nawiąże połączenie z nową instancją kanału.

Negocjowanie kanału jest wykonywane tylko dla pierwszej konwersacji w instancji kanału; wynegocjowane wartości dla instancji kanału są ustalone na tym etapie i nie można ich zmieniać podczas kolejnych konwersacji. Uwierzytelnianie TLS/SSL występuje również tylko w przypadku pierwszej konwersacji.

Jeśli wartość parametru MQCD SharingConversations zostanie zmieniona podczas inicjowania dowolnego zabezpieczenia, wysłania lub odebrania wyjścia dla pierwszej konwersacji w gnieździe przy użyciu połączenia klienckiego lub końca połączenia z serwerem instancji kanału, nowa wartość, która ma po zainicjowaniu wszystkich tych wyjść, jest używana do określenia wartości konwersacji współużytkowania dla instancji kanału (pierwszeństwo ma najniższa wartość).

Jeśli wynegocjowana wartość współużytkowania konwersacji wynosi zero, instancja kanału nigdy nie jest współużytkowana. Dalsze programy obsługi wyjścia, które ustawiają to pole na zero w podobny sposób, działają w swojej własnej instancji kanału.

Jeśli wynegocjowana wartość współużytkowania konwersacji jest większa niż zero, wówczas parametr MQCXP SharingConversations ma wartość TRUE w przypadku kolejnych wywołań wyjść, wskazując, że inne programy obsługi wyjścia w tej instancji kanału mogą być wprowadzane równocześnie z tym programem.

Podczas pisania programu obsługi wyjścia kanału należy rozważyć, czy będzie on uruchamiany w instancji kanału, która może obejmować współużytkowanie konwersacji. Jeśli instancja kanału może obejmować współużytkowanie konwersacji, należy wziąć pod uwagę wpływ na inne instancje wyjścia kanału zmiany pól MQCD. Wszystkie pola MQCD mają wspólne wartości we wszystkich konwersacjach współużytkowania. Po nawiązaniu instancji kanału, jeśli programy obsługi wyjścia próbują zmienić zawartość pól MQCD, mogą napotkać problemy, ponieważ inne instancje programów obsługi wyjścia działające w instancji kanału mogą próbować zmienić te same pola w tym samym czasie. Jeśli taka sytuacja może wystąpić w przypadku programów obsługi wyjścia, należy przekształcić do postaci szeregowej dostęp do zmaterializowanej tabeli MQCD w kodzie wyjścia.

W przypadku pracy z kanałem zdefiniowanym w celu współużytkowania konwersacji, ale nie ma potrzeby współużytkowania dla konkretnej instancji kanału, należy ustawić wartość parametru MQCD SharingConversations na wartość 1 lub 0 po zainicjowaniu wyjścia kanału w pierwszej konwersacji w instancji kanału. Informacje na temat wartości SharingConversationszawiera sekcja SharingConversations .

Przykład

Współużytkowanie konwersacji jest włączone.

Korzystasz z definicji kanału połączenia klienckiego, która określa program obsługi wyjścia.

Przy pierwszym uruchomieniu tego kanału program obsługi wyjścia zmienia niektóre parametry MQCD po jego zainicjowaniu. Są one wykonywane przez kanał, więc definicja, z którą jest uruchomiony kanał, różni się od tej, która została pierwotnie dostarczona. Parametr MQCXP SharingConversations jest ustawiony na wartość TRUE.

Następnym razem, gdy aplikacja nawiąże połączenie przy użyciu tego kanału, konwersacja zostanie uruchomiona w instancji kanału, która została wcześniej uruchomiona, ponieważ ma ona tę samą definicję kanału oryginalnego. Instancja kanału, z którą aplikacja nawiązuje połączenie po raz drugi, jest tą samą instancją, co po raz pierwszy, z którą jest połączona. W związku z tym korzysta z definicji, które zostały zmienione przez program obsługi wyjścia. Gdy program obsługi wyjścia jest inicjowany dla drugiej konwersacji, mimo że może zmieniać pola MQCD, *nie* są one zachowane przez kanał. Te same charakterystyki mają zastosowanie do wszystkich kolejnych konwersacji, które współużytkują instancję kanału.

Korzystanie z tabeli MQCONNX

Wywołania MQCONNX można użyć do określenia struktury definicji kanału (MQCD) w strukturze MQCNO.

Umożliwia to wywoływanie aplikacji klienckiej w celu określenia definicji kanału połączenia klienckiego w czasie wykonywania. Więcej informacji na ten temat zawiera sekcja Korzystanie z struktury MQCNO w wywołaniu MQCONNX. Jeśli używany jest produkt MQCONNX, wywołanie wysłane na serwerze zależy od konfiguracji poziomu serwera i procesu nasłuchującego.

W przypadku korzystania z klienta MQCONNX z poziomu klienta następujące opcje są ignorowane:

- MQCNO_STANDARD_BINDING

- MQCNO_FASTPATH_BINDING

Struktura MQCD, której można użyć, zależy od numeru wersji produktu MQCD, który jest używany. Informacje na temat wersji MQCD (MQCD_VERSION) zawiera sekcja MQCD Version(Wersja MQCD). Struktury MQCD można użyć na przykład do przekazywania programów obsługi wyjścia kanału na serwer. Jeśli używany jest produkt MQCD w wersji 3 lub nowszej, można użyć struktury do przekazania tablicy wyjść do serwera. Tej funkcji można użyć do wykonania więcej niż jednej operacji w tym samym komunikacie, na przykład do szyfrowania i kompresji, przez dodanie wyjścia dla każdej operacji, zamiast modyfikowania istniejącego wyjścia. Jeśli w strukturze MQCD nie zostanie określona tablica, sprawdzane będą pojedyncze pola wyjścia. Aby uzyskać więcej informacji na temat programów obsługi wyjścia kanału, patrz "Programy obsługi wyjścia kanału dla kanałów przesyłania komunikatów" na stronie 406.

Uchwyty połączeń współużytkowanych na tabeli MQCONNX

Istnieje możliwość współużytkowania uchwytów między różnymi wątkami w ramach tego samego procesu, przy użyciu uchwytów połączeń współużytkowanych.

W przypadku określenia uchwytu połączenia współużytkowanego uchwyt połączenia zwrócony z wywołania MQCONNX może być przekazywany w kolejnych wywołaniach MQI w dowolnym wątku w procesie.

Uwaga: Za pomocą uchwytu połączenia współużytkowanego na kliencie MQI produktu WebSphere MQ można nawiązać połączenie z menedżerem kolejek serwera, który nie obsługuje uchwytów połączeń współużytkowanych.

Więcej informacji na ten temat zawiera sekcja "Korzystanie z tabeli MQCONNX" na stronie 365.

Budowanie aplikacji dla klientów MQI produktu WebSphere MQ

Aplikacje mogą być budowane i uruchamiane w środowisku klienta MQI produktu WebSphere MQ . Aplikacja musi zostać zbudowana i dowiązana do używanego klienta MQI produktu WebSphere MQ . Sposób budowania i tworzenia powiązanych aplikacji różni się w zależności od używanej platformy i języka programowania.

Jeśli aplikacja ma być uruchamiana w środowisku klienta, można ją zapisać w językach przedstawionych w poniższej tabeli:

Tabela 47. Języki programowania obsługiwane w środowiskach klienckich

Platforma klienta	C	C++	COBOL	pTAL	RPG	Visual Basic
AIX	Tak	Tak	Tak			
HP Integrity NonStop Server	Tak		Tak	Tak		
HP-UX	Tak	Tak	Tak			
Linux	Tak	Tak	Tak			
Solaris	Tak	Tak	Tak			
Windows	Tak	Tak	Tak			Tak

Instrukcje dotyczące łączenia lub budowania aplikacji klienckich w tych językach można znaleźć w tematach pokrewnych.

Łączenie aplikacji C z kodem klienta MQI produktu WebSphere MQ

Po zapisaniu aplikacji WebSphere MQ , która ma być uruchamiana na kliencie MQI produktu WebSphere MQ , należy połączyć ją z menedżerem kolejek.

Istnieje możliwość połączenia aplikacji z menedżerem kolejek na dwa sposoby:

1. Bezpośrednio, w takim przypadku menedżer kolejek musi znajdować się na tej samej stacji roboczej, co aplikacja użytkownika.
2. Do pliku biblioteki klienta, który zapewnia dostęp do menedżerów kolejek na tej samej lub na innej stacji roboczej

Produkt WebSphere MQ udostępnia plik biblioteki klienta dla każdego środowiska:

AIX
Biblioteka libmqic.a dla aplikacji niewielowątkowych lub biblioteka libmqic_r.a dla aplikacji wielowątkowych.

HP-UX
Biblioteka libmqic.sl dla aplikacji niewielowątkowych lub biblioteka libmqic_r.sl dla aplikacji wielowątkowych.

Linux
Biblioteka libmqic.so dla aplikacji niewielowątkowych lub biblioteka libmqic_r.so dla aplikacji wielowątkowych.

Solaris
libmqic.so.

Aby użyć programów na stacji roboczej z zainstalowanym tylko klientem MQI produktu WebSphere MQ dla systemu Solaris, należy zrekompilować programy w celu połączenia ich z biblioteką klienta:

```
$ /opt/SUNWspro/bin/cc -o <prog> <prog> c -mt -lmqic \
-lsocket -lc -lnsl -ldl
```

Parametry muszą zostać wprowadzone w poprawnej kolejności, tak jak pokazano poniżej.

Windows
MQIC32.LIB.

Łączenie aplikacji C + + z kodem klienta MQI produktu WebSphere MQ

Można pisać aplikacje, które mają być uruchamiane na kliencie w języku C + +. Metody budowania różnią się w zależności od środowiska.

Informacje na temat łączenia aplikacji C++ można znaleźć w sekcji Budowanie programów WebSphere MQ C++.

Szczegółowe informacje na temat wszystkich aspektów używania języka C + + zawiera sekcja Używanie języka C++ .

Łączenie aplikacji w języku COBOL z kodem klienta MQI produktu IBM WebSphere MQ

Po napisanie aplikacji w języku COBOL, która ma być uruchamiana na kliencie MQI produktu IBM WebSphere MQ , należy połączyć ją z odpowiednią biblioteką.

Produkt IBM WebSphere MQ udostępnia plik biblioteki klienta dla każdego środowiska:

AIX
Powiąż niewielowątkową aplikację COBOL z biblioteką libmqicb.a lub wielowątkową aplikacją COBOL z biblioteką libmqicb_r.a.

HP-UX
Powiąż niewielowątkową aplikację COBOL z biblioteką libmqicb.sl lub wielowątkową aplikacją COBOL z biblioteką libmqicb_r.sl.

Linux
Powiąż niewielowątkową aplikację w języku COBOL z biblioteką libmqicb.so lub z wątkiem aplikacji COBOL z biblioteką libmqicb_r.so.

Solaris
Powiąż niewielowątkową aplikację w języku COBOL z biblioteką libmqicb.so lub z wątkiem aplikacji COBOL z biblioteką libmqicb_r.so.

Windows
Powiąż kod aplikacji z biblioteką MQICCBB dla 32-bitowego języka COBOL. Klient MQI produktu IBM WebSphere MQ dla produktu Windows nie obsługuje 16-bitowego języka COBOL.

Łączenie aplikacji Visual Basic z kodem klienta MQI produktu WebSphere MQ

Aplikacje Visual Basic można łączyć z kodem klienta MQI produktu WebSphere MQ w systemie Windows.

Powiąż aplikację Visual Basic z następującymi plikami włączenie:

CMQB.bas
MQI

CMQBB.bas
MQAI

CMQCFB.bas
Komendy PCF

CMQXB.bas
Kanały

Ustaw wartość `mqtype=2` dla klienta w kompilatorze Visual Basic, aby upewnić się, że automatyczny wybór biblioteki dll klienta jest poprawny:

MQIC32.dll
Windows 2000, Windows XP i Windows 2003

Uruchamianie aplikacji w środowisku klienta MQI produktu IBM WebSphere MQ

Aplikację IBM WebSphere MQ można uruchomić zarówno w pełnym środowisku IBM WebSphere MQ , jak i w środowisku klienta MQI produktu IBM WebSphere MQ bez zmiany kodu pod warunkiem, że spełnione są określone warunki.

Warunki te są następujące:

- Aplikacja nie musi łączyć się jednocześnie z więcej niż jednym menedżerem kolejek.
- Nazwa menedżera kolejek nie jest poprzedzona gwiazdką (*) w wywołaniu `MQCONN` lub `MQCONNX` .
- Aplikacja nie musi korzystać z żadnego z wyjątków wymienionych w sekcji Jakie aplikacje działają na kliencie MQI produktu IBM WebSphere MQ ?

Uwaga: Biblioteki używane w czasie edycji połączenia określają środowisko, w którym musi działać aplikacja.

Podczas pracy w środowisku klienta MQI produktu IBM WebSphere MQ należy pamiętać, że:

- Każda aplikacja działająca w środowisku klienta MQI produktu IBM WebSphere MQ ma własne połączenia z serwerami. Aplikacja tworzy jedno połączenie z serwerem za każdym razem, gdy wywołuje wywołanie `MQCONN` lub `MQCONNX` .
- Aplikacja wysyła i pobiera komunikaty synchronicznie. Oznacza to oczekiwanie od momentu, w którym wywołanie jest wydawane na kliencie, a także zwrot kodu zakończenia i kodu przyczyny w sieci.
- Wszystkie konwersję danych są wykonywane przez serwer, ale patrz także MQCCSID , aby uzyskać informacje na temat przesłaniania identyfikatora CCSID skonfigurowanego komputera.

Łączenie aplikacji klienta MQI produktu IBM WebSphere MQ z menedżerami kolejek

Aplikacja działająca w środowisku klienta MQI produktu IBM WebSphere MQ może łączyć się z menedżerem kolejek na różne sposoby. Można użyć zmiennych środowiskowych, struktury MQCNO lub tabeli definicji klienta.

Gdy aplikacja działająca w środowisku klienta IBM WebSphere MQ wysyła wywołanie MQCONN lub MQCONNX, klient identyfikuje sposób, w jaki ma być nawiązywać połączenie. Gdy wywołanie MQCONNX jest wysyłane przez aplikację na kliencie IBM WebSphere MQ , biblioteka klienta MQI wyszukuje informacje o kanale klienta w następującej kolejności:

1. Korzystanie z zawartości pól *ClientConnOffset* lub *ClientConnPtr* w strukturze MQCNO (jeśli jest podana). Te pola identyfikują strukturę definicji kanału (MQCD), która ma być używana jako definicja kanału połączenia klienta. Szczegóły połączenia mogą zostać przesłonięte przy użyciu wyjścia wstępnego połączenia. Więcej informacji na ten tematzawiera sekcja “Odwołanie do definicji połączenia przy użyciu wyjścia wstępnego z połączeniem z repozytorium” na stronie 433.
2. Jeśli ustawiona jest zmienna środowiskowa MQSERVER, używany jest kanał, który definiuje.
3. Jeśli plik `mqclient.ini` jest zdefiniowany i zawiera parametry ServerConnection, używany jest kanał, który definiuje. Więcej informacji na ten temat zawiera sekcja Konfigurowanie klienta przy użyciu pliku konfiguracyjnego oraz sekcja Sekcja CHANNELS pliku konfiguracyjnego klienta.
4. Jeśli ustawione są zmienne środowiskowe MQCHLLIB i MQCHLTAB, używana jest tabela definicji kanału klienta, do której mają być używane.
5. Jeśli plik `mqclient.ini` jest zdefiniowany i zawiera atrybuty ChannelDefinitionDirectory i ChannelDefinitionFile, atrybuty te są używane do zlokalizowania tabeli definicji kanału klienta. Więcej informacji na ten temat zawiera sekcja Konfigurowanie klienta przy użyciu pliku konfiguracyjnego oraz sekcja Sekcja CHANNELS pliku konfiguracyjnego klienta.
6. Na koniec, jeśli zmienne środowiskowe nie są ustawione, klient wyszuka tabelę definicji kanału klienta ze ścieżką i nazwą, które są ustanowione z `DefaultPrefix` w pliku `mqs.ini` . Jeśli wyszukiwanie w tabeli definicji klienta nie powiedzie się, klient używa następujących ścieżek:
   - UNIX and Linux : `/var/mqm/AMQCLCHL.TAB`
   - Windows: `C:\Program Files\IBM\Websphere MQ\amqclchl.tab`

Pierwsza z opcji opisanych na poprzedniej liście (przy użyciu pól *ClientConnOffset* lub *ClientConnPtr* komendy MQCNO) jest obsługiwana tylko przez wywołanie MQCONNX. Jeśli aplikacja używa komendy MQCONN, a nie MQCONNX, informacje o kanale są wyszukiwane w pozostałych pięciu sposobach w kolejności podanej na liście. Jeśli znalezienie informacji o kanale nie powiedzie się, wywołanie MQCONN lub MQCONNX nie powiedzie się.

Nazwa kanału (dla połączenia klienckiego) musi być zgodna z nazwą kanału połączenia z serwerem zdefiniowanym na serwerze dla wywołania MQCONN lub MQCONNX, aby powiodło się.

Jeśli z aplikacji zostanie wyświetlony kod powrotu MQRC_Q_MGR_NOT_AVAILABLE z komunikatem o błędzie w pliku dziennika błędów `AMQ9517 -uszkodzony plik`, należy zapoznać się z tabelami migracji i tabel definicji kanału klienta (CCDT).

Pojęcia pokrewne

Tabela definicji kanału klienta

Zadania pokrewne

Konfigurowanie połączeń między serwerem a klientem

Odsyłacze pokrewne

SERWER MQ

MQCHLLIB

MQCHLTAB

Łączenie aplikacji klienckich z menedżerami kolejek przy użyciu zmiennych środowiskowych

Informacje o kanale klienta mogą być dostarczane do aplikacji działającej w środowisku klienta za pomocą zmiennych środowiskowych MQSERVER, MQCHLLIB i MQCHLTAB.

Szczegółowe informacje na temat tych zmiennych można znaleźć w sekcji MQSERVER, MQCHLLIB i MQCHLTAB .

Łączenie aplikacji klienckich z menedżerami kolejek przy użyciu struktury MQCNO

Definicję kanału można określić w strukturze definicji kanału (MQCD), która jest dostarczana z użyciem struktury MQCNO wywołania MQCONNX.

Więcej informacji na ten temat zawiera sekcja Korzystanie ze struktury MQCNO w wywołaniu MQCONNX .

Łączenie aplikacji klienckich z menedżerami kolejek przy użyciu tabeli definicji kanału klienta

Jeśli komenda MQSC DEFINE CHANNEL zostanie użyta, szczegóły wprowadzone przez użytkownika zostaną umieszczone w tabeli definicji kanału klienta (ccdt). Zawartość parametru *QMgrName* wywołania MQCONN lub MQCONNX określa menedżera kolejek, z którym łączy się klient.

Dostęp do tego pliku jest uzyskiwany przez klienta w celu określenia kanału, który będzie używany przez aplikację. Jeśli istnieje więcej niż jedna odpowiednia definicja kanału, wybór kanału ma wpływ na atrybuty kanału klienta wagi kanału (CLNTWGHT) i powinowactwa połączenia (AFFINITY).

Rola tabeli definicji kanału klienta

Tabela definicji kanału klienta (CCDT) zawiera definicje kanałów połączenia klienta. Jest to szczególnie przydatne, gdy aplikacje klienckie mogą wymagać połączenia z wieloma alternatywnymi menedżerami kolejek.

Tabela definicji kanału klienta jest tworzona podczas definiowania menedżera kolejek.

Uwaga: Ten sam plik może być używany przez więcej niż jednego klienta IBM WebSphere MQ . Dostęp do różnych wersji tego pliku można uzyskać za pomocą zmiennych środowiskowych MQCHLLIB i MQCHLTAB IBM WebSphere MQ . Informacje na temat zmiennych środowiskowych można znaleźć w sekcji Korzystanie ze zmiennych środowiskowych programu WebSphere MQ .

Grupy menedżerów kolejek w tabeli definicji kanału klienta

Istnieje możliwość zdefiniowania zestawu połączeń w tabeli definicji kanału klienta (CCDT) jako *grupy menedżerów kolejek*. Istnieje możliwość połączenia aplikacji z menedżerem kolejek, który jest częścią grupy menedżerów kolejek. Można to zrobić, dodając przedrostek nazwy menedżera kolejek w wywołaniu programu `MQCONN` lub `MQCONNX` z gwiazdką.

Można wybrać opcję definiowania połączeń z więcej niż jednym serwerem, ponieważ:

- Klient ma zostać podłączony do dowolnego zestawu menedżerów kolejek, który jest uruchomiony, w celu zwiększenia dostępności.
- Użytkownik chce ponownie połączyć klienta z tym samym menedżerem kolejek, który został pomyślnie połączony z tym samym menedżerem kolejek, ale nawiąże połączenie z innym menedżerem kolejek, jeśli połączenie nie powiedzie się.
- Jeśli połączenie nie powiedzie się, należy ponowić próbę nawiązania połączenia klienta z innym menedżerem kolejek. W tym celu należy ponownie wydać komendę `MQCONN` w programie klienckim.
- Jeśli połączenie nie powiedzie się, użytkownik chce automatycznie ponownie połączyć połączenie klienta z innym menedżerem kolejek bez zapisywania kodu klienta.
- Użytkownik chce automatycznie ponownie połączyć połączenie klienta z inną instancją menedżera kolejek z wieloma instancjami, jeśli instancja rezerwowa przejmuje tę instancję, bez zapisywania kodu klienta.
- Użytkownik chce zrównoważyć połączenia klientów w wielu menedżerach kolejek, przy czym więcej klientów łączy się z niektórymi menedżerami kolejek niż inne.

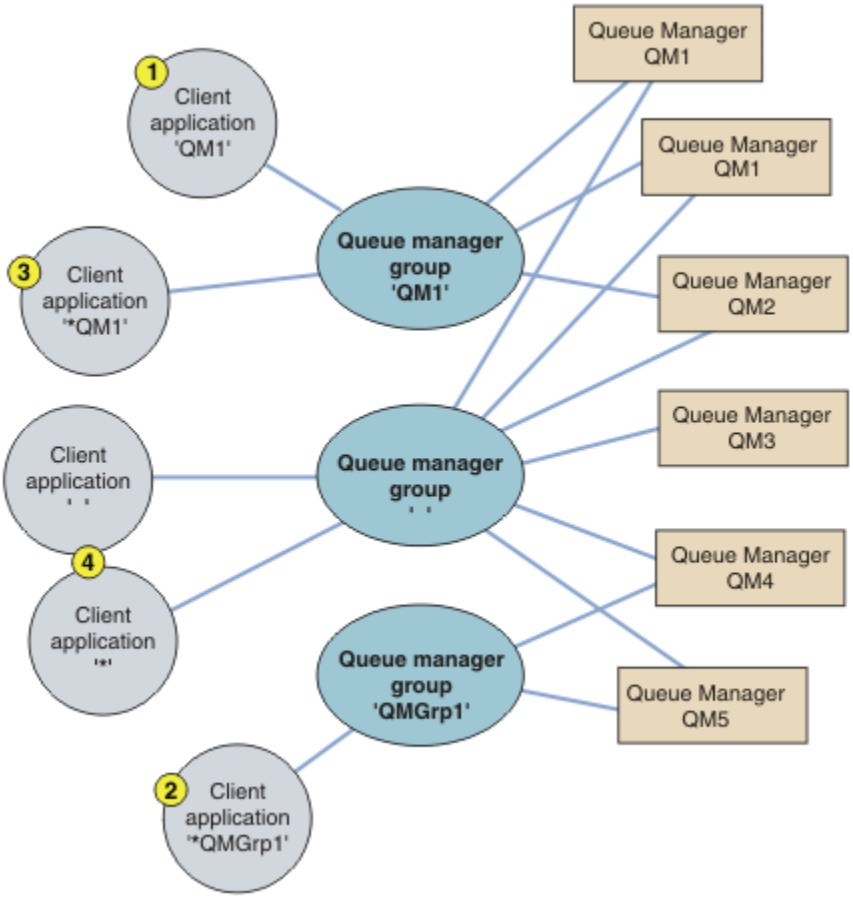
- Użytkownik chce rozpowszechnić ponowne połączenie wielu połączeń klienckich w wielu menedżerach kolejek i w czasie, w przypadku gdy duża liczba połączeń powoduje niepowodzenie.
- Użytkownik chce mieć możliwość przeniesienia menedżerów kolejek bez zmiany kodu aplikacji klienckiej.
- Użytkownik chce napisać programy użytkowe klienta, które nie muszą znać nazw menedżerów kolejek.

Łączenie się z różnymi menedżerami kolejek nie zawsze jest odpowiednie. Na przykład rozszerzony klient transakcyjny lub klient Java na serwerze WebSphere Application Servermoże wymagać nawiązania połączenia z przewidywalną instancją menedżera kolejek. Klasy WebSphere MQ classes for Java nie obsługują automatycznego nawiązywania ponownego połączenia przez klient.

Grupa menedżerów kolejek to zestaw połączeń zdefiniowanych w tabeli definicji kanału klienta (CCDT). Zestaw jest zdefiniowany przez jego członków o tej samej wartości atrybutu **QMNAME** w definicjach kanałów.

Rysunek 70 na stronie 371 jest graficzną reprezentacją tabeli połączeń klienta, przedstawiając trzy grupy menedżerów kolejek, dwie nazwane grupy menedżerów kolejek zapisane w tabeli CCDT jako **QMNAME**(QM1) i **QMNAME**(QMGrp1), a także jedną pustą lub domyślną grupę napisaną jako **QMNAME**(' ').

1. Grupa menedżerów kolejek QM1 ma trzy kanały połączenia klienckiego, łącząc je z menedżerami kolejek QM1 i QM2. QM1 może być menedżerem kolejek z wieloma instancjami, który znajduje się na dwóch różnych serwerach.
2. Domyślna grupa menedżerów kolejek ma sześć kanałów połączenia klienckiego łączących go ze wszystkimi menedżerami kolejek.
3. QMGrp1 ma kanały połączenia klienta z dwoma menedżerami kolejek, QM4 i QM5.



Rysunek 70. Grupy menedżerów kolejek

Cztery przykłady użycia tej tabeli połączeń klienta są opisane przy pomocy numerowanych aplikacji klienckich w produkcie Rysunek 70 na stronie 371.

1. W pierwszym przykładzie aplikacja kliencka przekazuje nazwę menedżera kolejek, QM1, jako parametr **QmgrName** do jej wywołania MQI produktu MQCONN lub MQCONNX . Kod klienta WebSphere MQ wybiera zgodną grupę menedżerów kolejek, QM1. Grupa zawiera trzy kanały połączenia, a klient MQI produktu WebSphere MQ próbuje połączyć się z serwerem QM1 za pomocą każdego z tych kanałów, dopóki nie znajdzie programu nasłuchującego WebSphere MQ dla połączenia przyłączonego do działającego menedżera kolejek o nazwie QM1.

   Kolejność prób połączenia zależy od wartości atrybutu AFFINITY połączenia klienckiego i współczynników korygujący kanału klienta. W ramach tych ograniczeń kolejność prób połączeń jest losowa, zarówno w przypadku trzech możliwych połączeń, jak i w czasie, w celu rozłożenia obciążenia nawiązając połączenia.

   Wywołanie MQCONN lub MQCONNX wydane przez aplikację kliencką powiedzie się, gdy połączenie zostanie nawiązane z działającą instancją serwera QM1.

2. W drugim przykładzie aplikacja kliencka przekazuje nazwę menedżera kolejek poprzedzoną gwiazdką (*QMGrp1 jako parametr **QmgrName** do jej wywołania MQI produktu MQCONN lub MQCONNX). Klient WebSphere MQ wybiera zgodną grupę menedżerów kolejek QMGrp1. Ta grupa zawiera dwa kanały połączenia klienckiego, a klient MQI produktu WebSphere MQ próbuje połączyć się z *dowolnym* menedżerem kolejek przy użyciu każdego kanału z kolei. W tym przykładzie klient MQI produktu WebSphere MQ musi nawiązać pomyślne połączenie. Nazwa menedżera kolejek, z którym łączy się połączenie, nie ma znaczenia.

   Reguła dla kolejności prób nawiązania połączenia jest taka sama jak poprzednio. Jedyna różnica polega na tym, że poprzedzając nazwę menedżera kolejek gwiazdką, klient wskazuje, że nazwa menedżera kolejek nie jest istotna.

   Wywołanie MQCONN lub MQCONNX wystawione przez aplikację kliencką powiedzie się, gdy połączenie zostanie nawiązane z działającą instancją dowolnego menedżera kolejek połączonego z kanałami w grupie menedżerów kolejek QMGrp1 .

3. Trzeci przykład jest zasadniczo taki sam, jak drugi, ponieważ parametr **QmgrName** jest poprzedzony gwiazdką (*QM1). Przykład ten ilustruje, że nie można określić, który menedżer kolejek połączenia kanału klienta ma nawiązać połączenie, sprawdzając atrybut QMNAME w jednej definicji kanału przez siebie. Fakt, że atrybut **QMNAME** definicji kanału to QM1, nie jest wystarczający do tego, aby było możliwe nawiązanie połączenia z menedżerem kolejek o nazwie QM1. Jeśli aplikacja kliencka prefikuje swój parametr **QmgrName** z gwiazdką, to dowolny menedżer kolejek może być elementem docelowym połączenia.

   W tym przypadku wywołania MQCONN lub MQCONNX wydane przez aplikację kliencką powiodą się, gdy połączenie zostanie nawiązane z działającą instancją klasy QM1 lub QM2.

4. W czwartym przykładzie przedstawiono użycie grupy domyślnej. W takim przypadku aplikacja kliencka przekazuje gwiazdkę, '*' lub pustą ' ', jako parametr **QmgrName** do jej wywołania MQI produktu MQCONN lub MQCONNX . Zgodnie z konwencją w definicji kanału klienta, pusty atrybut **QMNAME** oznacza domyślną grupę menedżerów kolejek, a pusty lub gwiazdka parametru **QmgrName** jest zgodny z pustym atrybutem **QMNAME** .

   W tym przykładzie domyślna grupa menedżerów kolejek ma połączenia kanału klienta ze wszystkimi menedżerami kolejek. Wybierając domyślną grupę menedżerów kolejek, aplikacja może być połączona z dowolnym menedżerem kolejek w grupie.

   Wywołanie MQCONN lub MQCONNX wystawione przez aplikację kliencką powiedzie się, gdy połączenie zostanie nawiązane z działającą instancją dowolnego menedżera kolejek.

   Uwaga: Grupa domyślna różni się od domyślnego menedżera kolejek, mimo że aplikacja używa pustego parametru **QmgrName** do łączenia się z domyślną grupą menedżerów kolejek lub do domyślnego menedżera kolejek. Pojęcie domyślnej grupy menedżerów kolejek ma znaczenie tylko w przypadku aplikacji klienckiej, a domyślny menedżer kolejek jest odpowiedni dla aplikacji serwera.

Zdefiniuj kanały połączenia klienckiego tylko w jednym menedżerze kolejek, w tym te kanały, które łączą się z drugim lub trzecim menedżerem kolejek. *Nie* zdefiniuj ich w dwóch menedżerach kolejek,

a następnie spróbuj scalić te dwie tabele definicji kanału klienta. Klient może uzyskać dostęp tylko do jednej tabeli definicji kanału klienta.

Przykłady

Zapoznaj się ponownie z listą przyczyn używania grup menedżerów kolejek na początku tego tematu. W jaki sposób korzystanie z grupy menedżerów kolejek udostępnia te możliwości?

Nawiąże połączenie z dowolnym zestawem menedżerów kolejek.
Zdefiniuj grupę menedżerów kolejek z połączeniami do wszystkich menedżerów kolejek w zestawie i nawiąże połączenie z grupą za pomocą parametru **QmgrName** z przedrostkiem gwiazdki.

Ponownie nawiąże połączenie z tym samym menedżerem kolejek, ale nawiąże połączenie z innym menedżerem kolejek, jeśli menedżer kolejek połączony z ostatnim czasem jest niedostępny.
Zdefiniuj grupę menedżerów kolejek tak, jak wcześniej, ale ustaw atrybut **AFFINITY**(PREFEROWANE) dla każdej definicji kanału klienta.

Jeśli połączenie nie powiedzie się, ponów próbę nawiązania połączenia z innym menedżerem kolejek.
Nawiąże połączenie z grupą menedżerów kolejek i ponownie wydaj wywołanie MQI produktu MQCONN lub MQCONNX , jeśli połączenie zostało zerwane lub menedżer kolejek nie powiedzie się.

Jeśli połączenie nie powiedzie się, automatycznie ponownie nawiąże połączenie z innym menedżerem kolejek.
Połącz się z grupą menedżerów kolejek za pomocą opcji MQCONNX **MQCNO** MQCNO_RECONNECT.

Automatycznie ponownie nawiąże połączenie z inną instancją menedżera kolejek z wieloma instancjami.
Wykonaj to samo, co w poprzednim przykładzie. W takim przypadku, aby ograniczyć grupę menedżerów kolejek do łączenia się z instancjami konkretnego menedżera kolejek z wieloma instancjami, należy zdefiniować grupę z połączeniami tylko z instancjami menedżera kolejek z wieloma instancjami.

Można również poprosić aplikację kliencką o wywołanie jej wywołania MQI produktu MQCONN lub MQCONNX bez znaku gwiazdki poprzedzonej przedrostkiem w parametrze **QmgrName** . W ten sposób aplikacja kliencka może połączyć się tylko z nazwanym menedżerem kolejek. Na koniec można ustawić opcję **MQCNO** na wartość MQCNO_RECONNECT_Q_MGR. Ta opcja akceptuje ponowne połączenia z tym samym menedżerem kolejek, który był wcześniej połączony. Tej wartości można również użyć do ograniczenia ponownego połączenia z tą samą instancją zwykłego menedżera kolejek.

Równoważenie połączeń klientów między menedżerami kolejek, z większą liczbę klientów połączonych z niektórymi menedżerami kolejek niż inne.
Zdefiniuj grupę menedżerów kolejek i ustaw atrybut **CLNTWGHT** dla każdej definicji kanału klienta, aby nierównomiernie rozprowadzać połączenia.

Rozłożyć ponownie obciążenie ponownie połączenia klienta i rozłożyć je z upływem czasu, po awarii połączenia lub menedżera kolejek.
Wykonaj to samo, co w poprzednim przykładzie. Klient MQI produktu WebSphere MQ zrandomizuje ponowne połączenia między menedżerami kolejek i rozprzestrzenia ponowne połączenia w czasie.

Przenieś menedżery kolejek bez zmiany kodu klienta.
Pakiet CCDT izoluje aplikację kliencką z miejsca, w którym znajduje się menedżer kolejek.

Użytkownik może wybrać dystrybucję tabeli połączeń klienta do każdego klienta lub umieścić tabelę CCDT w systemie plików współużytkowanych dla każdego klienta, do którego ma się odwoływać. Alternatywnie można użyć programowej wersji pakietu CCDT obsługiwanej w wywołaniu interfejsu MQI produktu MQCONNX i wywołać usługę, aby przekazać tabelę CCDT do aplikacji klienckiej.

Napisz aplikację kliencką, która nie zna nazw menedżerów kolejek.
Należy użyć nazw grup menedżerów kolejek i ustanowić konwencję nazewnictwa dla nazw grup menedżerów kolejek, która jest odpowiednia dla aplikacji klienckich w organizacji, a także odzwierciedlać architekturę rozwiązań zamiast nadawania nazw menedżerom kolejek.

Nawiązanie połączenia z grupami współużytkowania kolejek
Istnieje możliwość połączenia aplikacji z menedżerem kolejek, który jest częścią grupy współużytkowania kolejki. Można to zrobić, korzystając z nazwy grupy współużytkowania kolejki zamiast nazwy menedżera kolejek w wywołaniu MQCONN lub MQCONNX.

Nazwy grup współużytkujących kolejkę składają się z maksymalnie czterech znaków. Nazwa taka musi być unikalna w danej sieci i nie może być identyczna z nazwą menedżera kolejek.

Definicja kanału klienta powinna używać ogólnego interfejsu grupy współużytkowania kolejki do łączenia się z dostępnym menedżerem kolejek w grupie. Więcej informacji na ten temat zawiera sekcja Podłączanie klienta do grupy współużytkowania kolejek. Należy sprawdzić, czy menedżer kolejek, z którym łączy się program nasłuchujący, jest elementem grupy współużytkowania kolejek.

Przykłady ważenia kanałów i powinowactwa

Poniższe przykłady ilustrują sposób, w jaki kanały połączenia klienckiego są wybierane, gdy używane są niezerowe wartości `ClientChannelWeights` .

Atrybuty kanału `ClientChannelWeight` (Waga) i `ConnectionAffinity` (ConnectionAffinity) sterują sposobem wyboru kanałów połączenia klienckiego, gdy dla połączenia jest dostępny więcej niż jeden odpowiedni kanał. Kanały te są skonfigurowane do łączenia się z różnymi menedżerami kolejek w celu zapewnienia wyższej dostępności, równoważenia obciążenia lub obu tych funkcji. Wywołania MQCONN, które mogą być wynikiem połączenia z jednym z kilku menedżerów kolejek, muszą poprzedzić nazwę menedżera kolejek gwiazdką zgodnie z opisem w:Przykłady wywołań MQCONN: przykład 1. Nazwa menedżera kolejek zawiera znak gwiazdki (*).

Odpowiednie kanały kandydujące dla połączenia to te, w których atrybut QMNAME jest zgodny z nazwą menedżera kolejek określoną w wywołaniu MQCONN. Jeśli wszystkie mające zastosowanie kanały dla połączenia mają wartość `ClientChannelWaga` równą zeru (wartość domyślna), to są one wybierane w kolejności alfabetycznej, tak jak w przykładzie: Przykłady wywołań MQCONN: Przykład 1. Nazwa menedżera kolejek zawiera znak gwiazdki (*).

Poniższe przykłady ilustrują, co się dzieje, gdy używane są niezerowe wartości `ClientChannelWeights` . Należy zauważyć, że ponieważ ta funkcja obejmuje pseudo-losowy wybór kanału, przykłady pokazują sekwencję działań, które mogą się zdarzyć, a nie to, co na pewno się da.

Przykład 1. Wybieranie kanałów, gdy właściwość ConnectionAffinity jest ustawiona na wartość PREFERRED.
W tym przykładzie pokazano, w jaki sposób klient MQI produktu WebSphere MQ wybiera kanał z tabeli definicji kanału klienta, gdzie właściwość ConnectionAffinity jest ustawiona na wartość PREFERRED.

W tym przykładzie kilka maszyn klienckich korzysta z tabeli definicji kanału klienta (CCDT) udostępnianej przez menedżer kolejek. Pakiet CCDT zawiera kanały połączeń klientów z następującymi atrybutami (pokazywane przy użyciu składni komendy DEFINE CHANNEL):

```
CHANNEL(A) QMNAME(DEV) CONNAME(devqm.it.company.example)
CHANNEL(B) QMNAME(CORE) CONNAME(core1.ops.company.example) CLNTWGHT(5) +
AFFINITY(PREFERRED)
CHANNEL(C) QMNAME(CORE) CONNAME(core2.ops.company.example) CLNTWGHT(3) +
AFFINITY(PREFERRED)
CHANNEL(D) QMNAME(CORE) CONNAME(core3.ops.company.example) CLNTWGHT(2) +
AFFINITY(PREFERRED)
```

Problemy z aplikacją MQCONN (*CORE)

Kanał A nie jest kandydatem do tego połączenia, ponieważ atrybut QMNAME nie jest zgodny. Kanały B, C i D są identyfikowane jako kandydaci i są umieszczane w kolejności preferencji w oparciu o ich wagę. W tym przykładzie zamówienie może mieć wartość C, B, D. Klient próbuje połączyć się z menedżerem kolejek w pliku core2.ops.company.example. Nazwa menedżera kolejek pod tym adresem nie jest sprawdzona, ponieważ wywołanie MQCONN zawierło gwiazdkę w nazwie menedżera kolejek.

Należy pamiętać, że przy użyciu produktu `AFFINITY(PREFERRED)` za każdym razem, gdy ten konkretny klient łączy się z tym klientem, kanały będą umieszczać w tej samej kolejności początkowej preferencji. Ma to zastosowanie nawet wtedy, gdy połączenia pochodzą z różnych procesów lub w różnych momentach.

W tym przykładzie nie można uzyskać dostępu do menedżera kolejek na poziomie core.2.ops.company.example . Klient próbuje połączyć się z plikiem core1.ops.company.example , ponieważ kanał B jest następny w kolejności preferencji. Ponadto kanał C został zdegrazowany, aby stać się najmniej preferowanym.

Drugie wywołanie MQCONN (*CORE) jest wydawane przez tę samą aplikację. Kanał C został zdegrazowany przez poprzednie połączenie, więc najbardziej preferowanym kanałem jest teraz B. To połączenie jest nawiązane z plikiem core1.ops.company.example.

Druga maszyna współużytkuje tę samą tabelę definicji kanału klienta, co umożliwia umieszczenie kanałów w innej kolejności początkowej. Na przykład D, B, C. W normalnych okolicznościach, ze wszystkimi kanałami pracujące, aplikacje na tym komputerze są połączone z plikiem core3.ops.company.example , podczas gdy te na pierwszym komputerze są połączone z plikiem core2.ops.company.example. Pozwala to na równoważenie obciążenia dużej liczby klientów w wielu menedżerach kolejek, pozwalając każdemu klientowi na nawiązanie połączenia z tym samym menedżerem kolejek, jeśli jest on dostępny.

Przykład 2. Wybieranie kanałów, gdy parametr ConnectionAffinity jest ustawiony na wartość NONE.
W tym przykładzie pokazano, w jaki sposób klient MQI produktu WebSphere MQ wybiera kanał z tabeli definicji kanału klienta, gdzie wartość ConnectionAffinity jest ustawiona na wartość NONE.

W tym przykładzie pewna liczba klientów korzysta z tabeli definicji kanału klienta (CCDT) udostępnianej przez menedżera kolejek. Pakiet CCDT zawiera kanały połączeń klientów z następującymi atrybutami (pokazywane przy użyciu składni komendy DEFINE CHANNEL):

```
CHANNEL(A) QMNAME(DEV) CONNAME(devqm.it.company.example)
CHANNEL(B) QMNAME(CORE) CONNAME(core1.ops.company.example) CLNTWGHT(5) +
AFFINITY(NONE)
CHANNEL(C) QMNAME(CORE) CONNAME(core2.ops.company.example) CLNTWGHT(3) +
AFFINITY(NONE)
CHANNEL(D) QMNAME(CORE) CONNAME(core3.ops.company.example) CLNTWGHT(2) +
AFFINITY(NONE)
```

Aplikacja wydaje MQCONN (*CORE). Podobnie jak w poprzednim przykładzie, kanał A nie jest uwzględniany, ponieważ nazwa QMNAME nie jest zgodna. Kanał B, C lub D są wybierane na podstawie ich wagi, z prawdopodobieństwem 50%, 30% lub 20%. W tym przykładzie może zostać wybrany kanał B. Nie utworzono trwałej kolejności preferencji.

Zostanie wykonane drugie wywołanie MQCONN (*CORE). Ponownie wybierany jest jeden z trzech odpowiednich kanałów, z takimi samymi prawdopodobieństwami. W tym przykładzie wybierany jest kanał C. Jednak plik core2.ops.company.example nie odpowiada, dlatego między pozostałymi kanałami kandydującymi jest dokonany inny wybór. Wybrany jest kanał B, a aplikacja jest połączona z plikiem core1.ops.company.example.

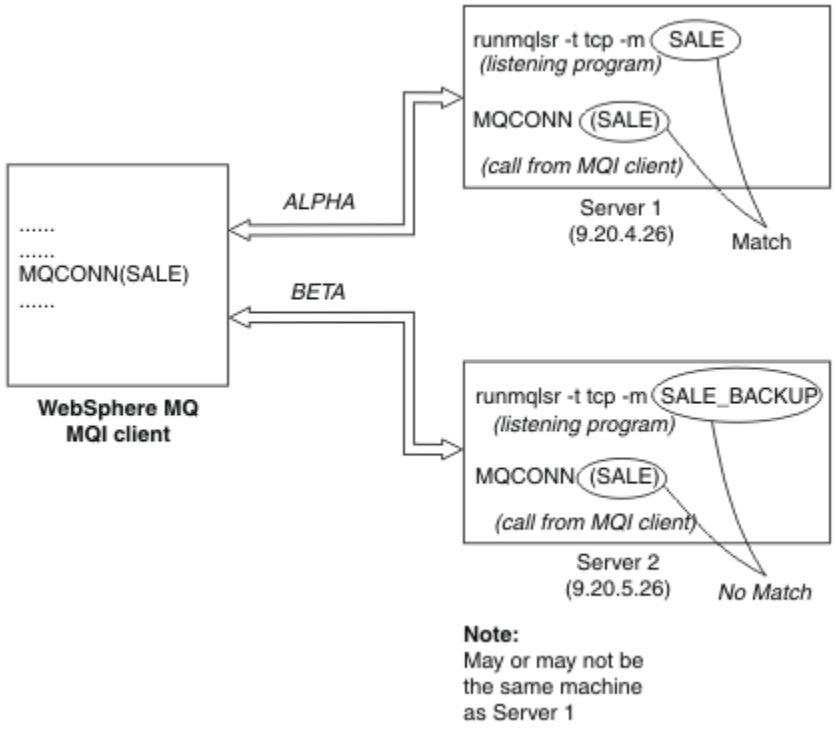
W przypadku wartości AFFINITY (NONE) każde wywołanie MQCONN jest niezależne od innych. Z tego powodu, gdy ta przykładowa aplikacja tworzy trzecią wartość MQCONN (*CORE), może to raz jeszcze podjąć próbę nawiązania połączenia przez zerwany kanał C, a następnie wybrać jedną z wartości B lub D.

Przykłady wywołań MQCONN

Przykłady użycia komendy MQCONN w celu nawiązania połączenia z konkretnym menedżerem kolejek lub do jednej z grup menedżerów kolejek.

W każdym z poniższych przykładów sieć jest taka sama; istnieje połączenie zdefiniowane dla dwóch serwerów z tego samego klienta MQI produktu WebSphere MQ . (W tych przykładach można użyć wywołania MQCONNX zamiast wywołania MQCONN).

Na komputerach serwera działają dwa menedżery kolejek: jedna o nazwie SALE , a druga o nazwie SALE_BACKUP.



Rysunek 71. Przykład MQCONN

Definicje kanałów w tych przykładach są następujące:

Definicje interfejsu ALE:

```
DEFINE CHANNEL(ALPHA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to WebSphere MQ MQI client')

DEFINE CHANNEL(ALPHA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) DESCR('WebSphere MQ MQI client connection to server 1') +
QMNAME(SALE)

DEFINE CHANNEL(BETA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.5.26) DESCR('WebSphere MQ MQI client connection to server 2') +
QMNAME(SALE)
```

Definicja SALE_BACKUP:

```
DEFINE CHANNEL(BETA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to WebSphere MQ MQI client')
```

Definicje kanałów klienta można podsumować w następujący sposób:

Nazwa	CHLTYPE	TRPTYPE	CONNAME	QMNAME
ALPHA	CLNTCONN	TCP	9.20.4.26	Sprz.
BETA	CLNTCONN	TCP	9.20.5.26	Sprz.

Co ilustrują przykłady MQCONN
Przykłady demonstrują użycie wielu menedżerów kolejek jako systemu zapasowego.

Przypuśćmy, że połączenie komunikacyjne z serwerem 1 jest tymczasowo zerwane. Demonstruje się użycie wielu menedżerów kolejek jako systemu zapasowego.

Każdy przykład obejmuje różne wywołania MQCONN i wyjaśnia, co dzieje się w konkretnym przykładzie, stosując następujące reguły:

1. Tabela definicji kanału klienta (CCDT) jest skanowana w kolejności alfabetycznej dla nazwy menedżera kolejek (pole QMNAME) odpowiadającej temu, który jest podany w wywołaniu MQCONN.
2. Jeśli zostanie znaleziony zgodny element, zostanie użyta definicja kanału.
3. Podejmowana jest próba uruchomienia kanału na komputerze identyfikowany przez nazwę połączenia (CONNAME). Jeśli operacja zakończy się pomyślnie, aplikacja będzie kontynuowana. Wymaga:
   - Program nasłuchujący, który ma być uruchomiony na serwerze.
   - Obiekt nasłuchiwania, który ma być połączony z tym samym menedżerem kolejek, co klient, z którym ma zostać nawiązane połączenie (o ile zostało określone).
4. Jeśli próba uruchomienia kanału nie powiedzie się i w tabeli definicji kanału klienta znajduje się więcej niż jedna pozycja (w tym przykładzie istnieją dwie pozycje), plik będzie wyszukiwany w celu uzyskania kolejnego dopasowania. Jeśli dopasowanie zostanie znalezione, przetwarzanie będzie kontynuowane w kroku 1.
5. Jeśli nie zostanie znaleziony żaden zgodny element lub nie ma więcej wpisów w tabeli definicji kanału klienta, a uruchomienie kanału nie powiodło się, aplikacja nie będzie mogła nawiązać połączenia. W wywołaniu MQCONN zwracany jest odpowiedni kod przyczyny i kod zakończenia. Aplikacja może podjąć działanie w oparciu o zwrócone kody przyczyny i zakończenia.

Przykład 1. Nazwa menedżera kolejek zawiera gwiazdkę (*)
W tym przykładzie aplikacja nie dotyczy menedżera kolejek, z którym łączy się dany menedżer kolejek. Aplikacja wysyła wywołanie MQCONN dla nazwy menedżera kolejek, w tym gwiazdkę. Wybrany jest odpowiedni kanał.

Problemy z aplikacją:

```
MQCONN (*SALE)
```

Postępując zgodnie z zasadami, dzieje się tak w tym przypadku:

1. Tabela definicji kanału klienta (CCDT) jest skanowana pod kątem nazwy menedżera kolejek SALE, która jest zgodna z wywołaniem MQCONN aplikacji.
2. Znaleziono definicje kanałów dla produktów ALPHA i BETA .
3. Jeśli jeden kanał ma wartość CLNTWGHT równą 0, ten kanał jest wybierany. Jeśli obie wartości mają wartość CLNTWGHT równą 0, wybierany jest kanał ALPHA , ponieważ jest on pierwszy w kolejności alfabetycznej. Jeśli oba kanały mają niezerową wartość CLNTWGHT, jeden kanał jest wybierany losowo, w oparciu o jego wagę.
4. Podjęto próbę uruchomienia kanału.
5. Jeśli wybrano kanał BETA , próba uruchomienia tego kanału powiodła się.
6. Jeśli wybrano kanał ALPHA , próba uruchomienia go NIE powiedzie się, ponieważ łącze komunikacyjne jest zerwane. Następnie należy wykonać następujące kroki:
   a. Jedynym innym kanałem dla nazwy menedżera kolejek SALE jest BETA.
   b. Podejmowana jest próba uruchomienia tego kanału-operacja ta powiodła się.
7. Sprawdzenie, czy nasłuchiwanie jest uruchomione, pokazuje, że istnieje jeden uruchomiony program nasłuchujący. Nie jest on połączony z menedżerem kolejek produktu SALE , ale dlatego, że parametr wywołania MQI zawiera gwiazdkę (*), nie jest wykonywane żadne sprawdzenie. Aplikacja jest połączona z menedżerem kolejek produktu SALE_BACKUP i kontynuuje przetwarzanie.

Przykład 2. Podana nazwa menedżera kolejek
W tym przykładzie aplikacja musi łączyć się z określonym menedżerem kolejek. Aplikacja wysyła wywołanie MQCONN dla tej nazwy menedżera kolejek. Wybrany jest odpowiedni kanał.

Aplikacja wymaga połączenia z określonym menedżerem kolejek o nazwie SALE, jak widać w wywołaniu MQI:

```
MQCONN (SALE)
```

Postępując zgodnie z zasadami, dzieje się tak w tym przypadku:

1. Tabela definicji kanału klienta (CCDT) jest skanowana w kolejności alfabetycznej, dla nazwy menedżera kolejek SALE, która jest zgodna z wywołaniem MQCONN aplikacji.
2. Pierwsza znaleziona definicja kanału to ALPHA.
3. Podjęto próbę uruchomienia kanału- *nie* powiodło się, ponieważ łącze komunikacyjne jest zerwane.
4. Tabela definicji kanału klienta zostanie ponownie zeskanowana dla nazwy menedżera kolejek SALE , a nazwa kanału BETA zostanie znaleziona.
5. Podjęto próbę uruchomienia kanału-operacja ta powiodła się.
6. Sprawdzenie, czy nasłuchiwanie jest uruchomione, wskazuje, że jest uruchomiony, ale nie jest on połączony z menedżerem kolejek produktu SALE .
7. W tabeli definicji kanału klienta nie ma kolejnych wpisów. Aplikacja nie może kontynuować działania i otrzymuje kod powrotu MQRC_Q_MGR_NOT_AVAILABLE.

Przykład 3. Nazwa menedżera kolejek jest pusta lub gwiazdka (*)
W tym przykładzie aplikacja nie dotyczy menedżera kolejek, z którym łączy się dany menedżer kolejek. Aplikacja wysyła komendę MQCONN, określając pustą nazwę menedżera kolejek lub gwiazdkę. Wybrany jest odpowiedni kanał.

Jest on traktowany w taki sam sposób, jak produkt “Przykład 1. Nazwa menedżera kolejek zawiera gwiazdkę (*)” na stronie 377.

Uwaga: Jeśli ta aplikacja była uruchomiona w środowisku innym niż klient MQI produktu WebSphere MQ , a nazwa była pusta, podejmowana jest próba nawiązania połączenia z domyślnym menedżerem kolejek. *Nie* jest to przypadek, gdy jest uruchamiany z poziomu środowiska klienta. Dostęp do menedżera kolejek jest powiązany z programem nasłuchującym, z którym łączy się kanał.

Problemy z aplikacją:

```
MQCONN ("")
```

lub wersji

```
MQCONN (*)
```

Postępując zgodnie z zasadami, dzieje się tak w tym przypadku:

1. Tabela definicji kanału klienta (CCDT) jest skanowana w kolejności alfabetycznej, dla nazwy menedżera kolejek, która jest pusta i jest zgodna z wywołaniem wywołania MQCONN aplikacji.
2. Pozycja dla kanału o nazwie ALPHA ma nazwę menedżera kolejek w definicji SALE. To *nie* jest zgodne z parametrem wywołania MQCONN, który wymaga, aby nazwa menedżera kolejek była pusta.
3. Następny wpis dotyczy nazwy kanału BETA.
4. `queue manager name` w definicji to SALE. Po raz kolejny *nie* jest to zgodne z parametrem wywołania MQCONN, który wymaga, aby nazwa menedżera kolejek była pusta.
5. W tabeli definicji kanału klienta nie ma kolejnych wpisów. Aplikacja nie może kontynuować działania i otrzymuje kod powrotu MQRC_Q_MGR_NOT_AVAILABLE.

Wyzwalanie w środowisku klienta

Komunikaty wysyłane przez aplikacje WebSphere MQ działające na klientach MQI produktu WebSphere MQ przyczyniają się do wyzwalania dokładnie w taki sam sposób, jak wszystkie inne komunikaty i mogą być używane do wyzwalania programów zarówno na serwerze, jak i na kliencie.

Wyzwalanie jest szczegółowo wyjaśnione w sekcji Kanały wyzwalane.

Monitor wyzwalacza i aplikacja, która ma być uruchomiona, muszą znajdować się w tym samym systemie.

Domyślne parametry kolejki wyzwalanej są takie same jak w środowisku serwera. W szczególności, jeśli w aplikacji klienckiej nie są określone opcje sterujące punktu synchronizacji MQPMO, umieszczanie komunikatów w kolejce wyzwalanej, która jest lokalna względem menedżera kolejek systemu z/OS , komunikaty są umieszczane w obrębie jednostki pracy. Jeśli warunek wyzwalający zostanie spełniony, komunikat wyzwalacza jest umieszczany w kolejce inicjujący w tej samej jednostce pracy i nie może zostać pobrany przez monitor wyzwalacza do momentu zakończenia jednostki pracy. Proces, który ma zostać wyzwolony, nie zostanie uruchomiony, dopóki jednostka pracy nie zostanie zakończona.

Definicja procesu

Należy zdefiniować definicję procesu na serwerze, ponieważ jest ona powiązana z kolejką, w której ustawiono wyzwalanie.

Obiekt procesu definiuje, co ma zostać wyzwolone. Jeśli klient i serwer nie są uruchomione na tej samej platformie, wszystkie procesy uruchomione przez monitor wyzwalacza muszą definiować *ApplType*, w przeciwnym razie serwer przyjmuje jego definicje domyślne (czyli typ aplikacji, która jest zwykle powiązana z maszyną serwera) i powoduje niepowodzenie.

Jeśli na przykład monitor wyzwalacza działa na kliencie z systemem Windows i chce wysłać żądanie do serwera w innym systemie operacyjnym, należy zdefiniować parametr MQAT_WINDOWS_NT. W przeciwnym razie w innym systemie operacyjnym używane są definicje domyślne, a proces nie powiedzie się.

monitor wyzwalacza

Monitor wyzwalacza udostępniany przez produkty WebSphere MQ innych niżz/OS działa w środowiskach klienckich dla systemów UNIX, Linux i Windows .

Aby uruchomić monitor wyzwalacza, wydaj jedną z następujących komend:

- Windows Linux UNIX Na platformach Windows, UNIX i Linux :

```
runmqtmc [-m QMgrName] [-q InitQ]
```

Domyślną kolejką inicjujową jest SYSTEM.DEFAULT.INITIATION.QUEUE w domyślnym menedżerze kolejek. Kolejka inicjujący jest w miejscu, w którym monitor wyzwalacza wyszukuje komunikaty wyzwalacza. Następnie wywołuje on programy dla odpowiednich komunikatów wyzwalacza. Ten monitor wyzwalacza obsługuje domyślny typ aplikacji i jest taki sam, jak produkt `runmqtrm` , z wyjątkiem tego, że łączy biblioteki klienckie.

Łańcuch komendy, zbudowany przez monitor wyzwalacza, jest następujący:

1. *ApplicId* z odpowiedniej definicji procesu. *ApplicId* to nazwa programu, który ma zostać uruchomiony, ponieważ zostanie on wprowadzony w wierszu komend.
2. Struktura MQTMC2 ujęta w cudzysłów, która została uzyskana z kolejki inicjuj. Łańcuch komendy jest uruchamiany, który ma ten łańcuch, dokładnie tak, jak jest to podany, w cudzysłowie, aby komenda systemowa zaakceptowała ją jako jeden parametr.
3. *EnvrData* z odpowiedniej definicji procesu.

Monitor wyzwalacza nie widzi, czy istnieje inny komunikat w kolejce inicjujący do momentu zakończenia aplikacji, która została uruchomiona. Jeśli aplikacja ma do wykonania wiele operacji, monitor wyzwalacza może nie nadążać za liczbą przybywających komunikatów wyzwalacza. Istnieją dwa sposoby radzenia sobie z tą sytuacją:

1. Czy uruchomiono więcej monitorów wyzwalaczy

W przypadku wybrania większej liczby monitorów wyzwalacza można sterować maksymalną liczbą aplikacji, które mogą być uruchamiane w dowolnym momencie.

2. Uruchom uruchomione aplikacje w tle

   Jeśli aplikacje mają być uruchamiane w tle, produkt WebSphere MQ nie ma żadnych ograniczeń dotyczących liczby aplikacji, które mogą być uruchamiane.

Aby uruchomić uruchomioną aplikację w tle w systemach UNIX and Linux , należy umieścić znak & (ampersand) na końcu *EnvrData* definicji procesu.

Aplikacje CICS (inne niżz/OS)

Program użytkowy innej niżz/OS CICS , który wysyła wywołanie MQCONN lub MQCONNX , musi być zdefiniowany jako CEDA jako RESIDENT. Jeśli aplikacja serwera CICS zostanie ponownie dowiązana jako klient, ryzyko utraty synchronizacji punktów synchronizacji jest ryzyko.

Program użytkowy innej niżz/OS CICS , który wysyła wywołanie MQCONN lub MQCONNX , musi być zdefiniowany jako CEDA jako RESIDENT. Aby kod rezydentny był jak najmały, można utworzyć dowiązanie do osobnego programu w celu wydania wywołania MQCONN lub MQCONNX .

Jeśli do zdefiniowania połączenia klienta używana jest zmienna środowiskowa MQSERVER, musi ona zostać określona w polu CICSENV.CMD .

Aplikacje produktu WebSphere MQ mogą być uruchamiane w środowisku serwera WebSphere MQ lub na kliencie WebSphere MQ bez zmiany kodu. Jednak w środowisku serwera WebSphere MQ program CICS może działać jako koordynator punktu synchronizacji, a użytkownik korzysta z usług EXEC CICS SYNCPOINT i EXEC CICS SYNCPOINT ROLLBACK, a nie MQCMIT i MQBACK. Jeśli aplikacja CICS jest po prostu relinked as a client, sync point support is lost. Produkty MQCMIT i MQBACK muszą być używane w przypadku aplikacji działającej na kliencie MQI produktu WebSphere MQ .

Przygotowywanie i uruchamianie aplikacji CICS i Tuxedo

Aby uruchamiać aplikacje CICS i Tuxedo jako aplikacje klienckie, należy używać różnych bibliotek z tych aplikacji, które są używane z aplikacjami serwera. Identyfikator użytkownika, pod którym działa aplikacja, również jest inny.

Aby przygotować aplikacje CICS i Tuxedo do uruchamiania jako aplikacje klienckie MQI produktu WebSphere MQ , należy postępować zgodnie z instrukcjami znajdującymi się w sekcji Konfigurowanie rozszerzonego klienta transakcyjnego.

Należy jednak pamiętać, że informacje, które dotyczą przygotowywania aplikacji CICS i Tuxedo, w tym przykładowych programów dostarczanych z produktem WebSphere MQ, zakłada, że użytkownik przygotowuje aplikacje do działania w systemie serwera WebSphere MQ . W wyniku tego informacje odnoszą się wyłącznie do bibliotek produktu WebSphere MQ , które są przeznaczone do użycia w systemie serwera. Przygotowując aplikacje klienckie, należy wykonać następujące czynności:

- Użyj odpowiedniej biblioteki systemu klienta dla powiązań języka, z których korzysta aplikacja. Na przykład w przypadku aplikacji napisanych w języku C w systemach AIX, HP-UXlub Solaris należy użyć biblioteki libmqic biblioteki zamiast biblioteki libmqm. W systemach Windows należy użyć biblioteki mqic.lib zamiast katalogu mqm.lib.
- Zamiast bibliotek systemowych serwera wyświetlanych w systemie Tabela 48 na stronie 380, dla systemów AIX, HP-UXi Solaris oraz Tabela 49 na stronie 381w systemach Windows należy użyć równoważnych bibliotek systemowych klienta. Jeśli biblioteka systemowa serwera nie znajduje się na liście w tych tabelach, należy użyć tej samej biblioteki w systemie klienckim.

Tabela 48. Biblioteki systemowe klienta w systemach AIX, HP-UXi Solaris

Biblioteka dla systemu serwera WebSphere MQ	**Równoważna biblioteka do użycia w systemie klienckim WebSphere MQ**
libmqmxa	libmqcxa

Tabela 49. Biblioteki systemowe klienta w systemach Windows	
Biblioteka dla systemu serwera WebSphere MQ	**Równoważna biblioteka do użycia w systemie klienckim WebSphere MQ**
mqmxa.lib	mqcxa.lib
mqmtux.lib	mqcxa.lib
mqmenc.lib	mqcxa.lib
mqmcics4.lib	mqccics4.lib

Identyfikator użytkownika używany przez aplikację kliencką

Gdy aplikacja serwera WebSphere MQ jest uruchamiana w systemie CICS, zwykle jest on przełączany z poziomu użytkownika CICS na identyfikator użytkownika transakcji. Jednak w przypadku uruchamiania aplikacji klienckiej MQI produktu WebSphere MQ w systemie CICS zachowuje uprawnienia uprzywilejowanego systemu CICS.

Programy przykładowe CICS i Tuxedo

Programy przykładowe CICS i Tuxedo przeznaczone do użycia w systemach AIX, HP-UX, Solaris i Windows.

Tabela 50 na stronie 381 zawiera przykładowe programy CICS i Tuxedo, które są dostarczane w celu użycia w systemach klienckich AIX, HP-UXi Solaris. Tabela 51 na stronie 381 zawiera listę równoważnych informacji dla systemów klienckich Windows . Tabele zawierają również listę plików używanych do przygotowania i uruchamiania programów. Opis przykładowych programów można znaleźć w sekcji “Przykład transakcji CICS” na stronie 122 i “Próbki TUXEDO” na stronie 160.

Tabela 50. Przykładowe programy dla systemów klienckich AIX, HP-UXi Solaris		
Opis	**Źródło**	**Moduł wykonywalny**
Program CICS	amqscic0.ccs	amqscic
Plik nagłówkowy dla programu CICS	amqscih0.h	-
Program kliencki Tuxedo do umieszczania komunikatów	amqstxpx.c	-
Program kliencki Tuxedo do pobierania wiadomości	amqstxgx.c	-
Program serwerowy Tuxedo dla dwóch programów klienckich	amqstxsx.c	-
Plik UBBCONFIG dla programów Tuxedo	ubbstxcx.cfg	-
Plik tabeli pól dla programów Tuxedo	amqstxvx.flds	-
Wyświetl plik opisu dla programów Tuxedo	amqstxvx.v	-

Tabela 51. Przykładowe programy dla systemów klienckich Windows		
Opis	**Źródło**	**Moduł wykonywalny**
Transakcja CICS	amqscic0.ccs	amqscic
Plik nagłówkowy dla transakcji CICS	amqscih0.h	-
Program kliencki Tuxedo do umieszczania komunikatów	amqstxpx.c	-
Program kliencki Tuxedo do pobierania wiadomości	amqstxgx.c	-
Program serwerowy Tuxedo dla dwóch programów klienckich	amqstxsx.c	-

Tabela 51. Przykładowe programy dla systemów klienckich Windows (kontynuacja)		
Opis	**Źródło**	**Moduł wykonywalny**
Plik UBBCONFIG dla programów Tuxedo	ubbstxcx.cfg	-
Plik tabeli pól dla programów Tuxedo	amqstxvx.fld	-
Wyświetl plik opisu dla programów Tuxedo	amqstxvx.v	-
Plik makefile dla programów Tuxedo	amqstxmc.mak	-
Plik ENVFILE dla programów Tuxedo	amqstxen.env	-

Komunikat o błędzie AMQ5203, jako zmodyfikowany dla aplikacji CICS i Tuxedo

W przypadku uruchamiania aplikacji CICS lub Tuxedo, które korzystają z rozszerzonego klienta transakcyjnego, mogą być wyświetlane standardowe komunikaty diagnostyczne. Jeden z nich został zmodyfikowany do użycia z rozszerzonym klientem transakcyjnym

Komunikaty, które mogą zostać wyświetlone w plikach dziennika błędów produktu WebSphere MQ , są opisane w sekcji Komunikaty diagnostyczne: AMQ4000-9999. Komunikat AMQ5203 został zmodyfikowany do użycia z rozszerzonym klientem transakcyjnym. Poniżej znajduje się tekst zmodyfikowanego komunikatu:

AMQ5203: Wystąpił błąd podczas wywoływania interfejsu XA.

Wyjaśnienie

Numer błędu to & 2, gdzie wartość 1 wskazuje, że podana wartość flag dla & 1 była niepoprawna, 2 wskazuje, że wystąpiła próba użycia bibliotek wielowątkowych i niewielowątkowych w tym samym procesie, 3 wskazuje, że wystąpił błąd z podaną nazwą menedżera kolejek '& 3', 4 wskazuje, że identyfikator menedżera zasobów & 1 jest niepoprawny, 5 wskazuje na to, że podjęto próbę użycia drugiego menedżera kolejek o nazwie '& 3 'gdy inny menedżer kolejek był już połączony, wartość 6 wskazuje, że menedżer transakcji został wywołany w momencie, gdy aplikacja nie jest połączona z menedżerem kolejek, 7 wskazuje, że wywołanie XA zostało wykonane w trakcie innego wywołania, 8 wskazuje, że łańcuch xa_info' & 4 'w wywołaniu xa_open zawierał niepoprawną wartość parametru o nazwie' & 5 ', a 9 wskazuje, że łańcuch xa_info' & 4 'w wywołaniu xa_open nie ma wymaganego parametru, nazwa parametru' & 5 '.

Odpowiedź użytkownika

Popraw błąd i spróbuj ponownie wykonać operację.

Przygotowywanie i uruchamianie aplikacji serwera Microsoft Transaction Server

Aby przygotować aplikację MTS do uruchamiania jako aplikacja kliencka MQI produktu WebSphere MQ , należy postępować zgodnie z poniższa instrukcjami, aby uzyskać informacje o środowisku.

Ogólne informacje na temat tworzenia aplikacji Microsoft Transaction Server (MTS), które uzyskują dostęp do zasobów produktu WebSphere MQ , można znaleźć w sekcji dotyczącej MTS w Centrum pomocy produktu WebSphere MQ .

Aby przygotować aplikację MTS do uruchamiania jako aplikacja kliencka MQI produktu WebSphere MQ , wykonaj jedną z następujących czynności dla każdego komponentu aplikacji:

- Jeśli komponent używa powiązań języka C dla interfejsu MQI, należy postępować zgodnie z instrukcjami w sekcji “Przygotowywanie programów w języku C w systemie Windows” na stronie 471 , ale należy połączyć komponent z biblioteką mqicxa.lib zamiast z biblioteką mqic.lib.

- Jeśli komponent korzysta z klas języka C++ WebSphere MQ , należy postępować zgodnie z instrukcjami w sekcji “Budowanie programów C++ w systemie Windows” na stronie 669 , ale należy połączyć ten komponent z biblioteką imqx23vn.lib zamiast imqc23vn.lib.
- Jeśli komponent używa powiązań języka Visual Basic dla interfejsu MQI, należy postępować zgodnie z instrukcjami w “Przygotowywanie programów Visual Basic w systemie Windows” na stronie 475 , ale po zdefiniowaniu projektu Visual Basic należy wpisać `MqType=3` w polu **Warunkowe argumenty kompilacji** .
- Jeśli komponent korzysta z klas automatyzacji produktu WebSphere MQ dla ActiveX (MQAX), należy zdefiniować zmienną środowiskową GMQ_MQ_LIB o wartości `mqic32xa.dll` .

  Można zdefiniować zmienną środowiskową z poziomu aplikacji lub zdefiniować ją w taki sposób, aby jej zasięg był szeroki. Jednak zdefiniowanie go jako całego systemu może spowodować, że dowolna istniejąca aplikacja MQAX, która nie definiuje zmiennej środowiskowej z aplikacji, zachowuje się niepoprawnie.

Przygotowywanie i uruchamianie aplikacji JMS produktu WebSphere MQ

Aplikacje JMS produktu WebSphere MQ można uruchamiać w trybie klienta z serwerem WebSphere Application Server jako menedżerem transakcji. Mogą zostać wyświetlone pewne komunikaty ostrzegawcze.

Aby przygotować i uruchomić aplikacje JMS produktu WebSphere MQ w trybie klienta z serwerem WebSphere Application Server jako menedżerem transakcji, należy postępować zgodnie z instrukcjami w sekcji “Korzystanie z klas produktu WebSphere MQ dla usługi JMS” na stronie 734.

Po uruchomieniu aplikacji klienckiej WebSphere MQ JMS mogą być wyświetlane następujące komunikaty ostrzegawcze:

MQJE080
Niewystarczające jednostki licencji-uruchom komendę setmqcap

MQJE081
Plik zawierający informacje o jednostce licencji znajduje się w niepoprawnym formacie-uruchom komendę setmqcap

MQJE082
Nie można znaleźć pliku zawierającego informacje o jednostce licencji-uruchom komendę setmqcap

Procedury zewnętrzne, wyjścia interfejsu API i usługi instalacyjne produktu WebSphere MQ

Obiekty menedżera kolejek można rozszerzać, korzystając z wyjść użytkownika, wyjść funkcji API lub usług instalowalnych. Ten temat zawiera odsyłacze do informacji na temat używania i programowania tych programów.

Aby uzyskać informacje na temat korzystania z wyjść użytkownika, wyjść funkcji API i usług instalowalnych w celu rozszerzenia infrastruktury menedżera kolejek, należy zapoznać się z Rozszerzanie obiektów menedżera kolejek.

Informacje na temat pisania i kompilowania wyjść i instalowalnych usług zawiera sekcja “Pisanie i kompilowanie wyjść i usług instalowalnych” na stronie 384.

Pojęcia pokrewne
Programy obsługi wyjścia kanału dla kanałów MQI
Odsyłacze pokrewne
Odwołanie do wyjścia funkcji API
Informacje uzupełniające o interfejsie usług instalowalnych

Pisanie i kompilowanie wyjść i usług instalowalnych

Istnieje możliwość tworzenia i kompilowania wyjść bez łączenia się z żadną biblioteką produktu IBM WebSphere MQ w systemach UNIX, Linux i Windows.

O tym zadaniu

Ten temat dotyczy tylko systemów Windowsi UNIX and Linux . Szczegółowe informacje na temat pisania wyjść i instalowalnych usług dla innych platform można znaleźć w odpowiednich tematach dotyczących poszczególnych platform.

Jeśli produkt IBM WebSphere MQ jest zainstalowany w położeniu innym niż domyślne, należy napisać i skompilować wyjścia bez łączenia się z żadną biblioteką produktu IBM WebSphere MQ .

Wyjścia można zapisywać i kompilować w systemach Windows, UNIX and Linux bez łączenia żadnej z tych bibliotek produktu IBM WebSphere MQ :

- mqmzf
- MQM
- mqmvx
- mqmvxd
- mqic
- mqutl

Istniejące wyjścia, które są powiązane z tymi bibliotekami, nadal działają, pod warunkiem, że w systemach UNIX and Linux IBM WebSphere MQ jest zainstalowane w domyślnym położeniu.

Procedura

1. Dołącz plik nagłówkowy cmqec.h .

   Dołączenie tego pliku nagłówkowego powoduje automatyczne dołączanie plików nagłówkowych cmqc.h, cmqxc.h i cmqzc.h .
2. Zapisz wyjście tak, aby wywołania MQI i DCI były wykonywane za pomocą struktury MQIEP. Więcej informacji na temat struktury MQIEP można znaleźć w sekcji Struktura MQIEP.
   - Usługi instalowalne
     - Aby wskazać wywołanie MQZEP, należy użyć parametru **Hconfig** .
     - Przed użyciem parametru **Hconfig** należy sprawdzić, czy pierwsze 4 bajty produktu **Hconfig** są zgodne z **StrucId** struktury MQIEP.
     - Więcej informacji na temat pisania instalowalnych komponentów usług znajduje się w sekcji MQIEP.
   - Wyjścia funkcji API
     - Aby wskazać wywołanie MQXEP, należy użyć parametru **Hconfig** .
     - Przed użyciem parametru **Hconfig** należy sprawdzić, czy pierwsze 4 bajty produktu **Hconfig** są zgodne z **StrucId** struktury MQIEP.
     - Więcej informacji na temat pisania wyjść funkcji API zawiera sekcja “Pisanie wyjść funkcji API” na stronie 398.
   - Wyjścia kanału
     - Należy użyć parametru **pEntryPoints** struktury MQCXP, aby wskazać wywołania MQI i DCI.
     - Przed użyciem produktu **pEntryPoints**należy sprawdzić, czy numer wersji produktu MQCXP jest w wersji 8 lub nowszej.
     - Aby uzyskać więcej informacji na temat pisania wyjść kanału, zobacz: “Pisanie programów obsługi wyjścia kanału” na stronie 409.

- Wyjścia konwersji danych
  - Należy użyć parametru **pEntryPoints** struktury MQDXP, aby wskazać wywołania MQI i DCI.
  - Przed użyciem produktu **pEntryPoints**należy sprawdzić, czy numer wersji produktu MQDXP jest w wersji 2 lub nowszej.
  - Do utworzenia kodu konwersji danych, który korzysta z parametru **pEntryPoints** , można użyć komendy **crtmqcvx** oraz pliku źródłowego amqsvfc0.c . Patrz “Zapisywanie wyjścia konwersji danych dla produktu WebSphere MQ for Windows” na stronie 431 oraz “Zapisywanie wyjścia konwersji danych dla produktu WebSphere MQ w systemach UNIX and Linux” na stronie 427.
  - Jeśli istnieją wyjścia konwersji danych, które zostały wygenerowane za pomocą komendy **crtmqcvx** , należy ponownie wygenerować wyjście przy użyciu zaktualizowanej komendy.
  - Aby uzyskać więcej informacji na temat pisania wyjść konwersji danych, zobacz: “Pisanie wyjść konwersji danych” na stronie 426.
- Wyjścia przed podłączniem
  - Należy użyć parametru **pEntryPoints** struktury MQNXP, aby wskazać wywołania MQI i DCI.
  - Przed użyciem produktu **pEntryPoints**należy sprawdzić, czy numer wersji produktu MQNXP jest w wersji 2 lub nowszej.
  - Aby uzyskać więcej informacji na temat pisania wyjść przed nawiązaniem połączenia, patrz “Odwołanie do definicji połączenia przy użyciu wyjścia wstępnego z połączeniem z repozytorium” na stronie 433.
- Publikuj wyjścia
  - Aby wskazać wywołania MQI i DCI, należy użyć parametru **pEntryPoints** struktury MQPSXP.
  - Przed użyciem produktu **pEntryPoints**należy sprawdzić, czy numer wersji produktu MQPSXP jest w wersji 2 lub nowszej.
  - Aby uzyskać więcej informacji na temat pisania wyjść publikowania, zobacz: “Pisanie i kompilowanie wyjść publikowania” na stronie 435.
- Wyjścia obciążenia klastra
  - Aby wskazać wywołania MQXCLWLN, należy użyć parametru **pEntryPoints** struktury MQWXP.
  - Przed użyciem produktu **pEntryPoints**należy sprawdzić, czy numer wersji produktu MQWXP jest w wersji 4 lub nowszej.
  - Więcej informacji na temat pisania wyjść obciążenia klastra zawiera sekcja “Pisanie i kompilowanie wyjść obciążenia klastra” na stronie 437.

Na przykład w przypadku wyjścia kanału wywołującej wywołanie MQPUT:

```
pChannelExitParms -> pEntryPoints -> MQPUT_Call(pChannelExitParms -> Hconn,
                                                Hobj,
                                                &md,
                                                &pmo,
                                                messlen,
                                                buffer,
                                                &CompCode,
                                                &Reason);
```

Dodatkowe przykłady można znaleźć w publikacji “Przykładowe programy produktu WebSphere MQ” na stronie 98.

3. Skompiluj wyjście:
   - Nie należy łączyć się z bibliotekami produktu IBM WebSphere MQ .
   - Nie należy dołączać wbudowanej ścieżki RPath do żadnych bibliotek produktu IBM WebSphere MQ w wyjściu.
   - Więcej informacji na temat kompilowania programu obsługi wyjścia można znaleźć w jednym z następujących tematów:
     - Wyjścia funkcji API: “Kompilowanie wyjść funkcji API” na stronie 400.

- Wyjścia kanału, wyjścia publikowania, wyjścia obciążenia klastra: “Kompilowanie programów obsługi wyjścia kanału w systemach Windows, UNIX and Linux” na stronie 424.
- Wyjścia konwersji danych: “Pisanie wyjść konwersji danych” na stronie 426.

4. Umieść wyjście w jednym z następujących miejsc:
   - Ścieżka wyboru, którą można w pełni kwalifikować podczas konfigurowania wyjścia
   - Domyślna ścieżka wyjścia w określonym katalogu instalacyjnym. Na przykład: *MQ_DATA_PATH*/exits/installation2.
   - Domyślna ścieżka wyjścia

     Domyślna ścieżka wyjścia to *MQ_DATA_PATH*/exits dla wyjść 32-bitowych, a *MQ_DATA_PATH*/exits64 dla 64-bitowych wyjść. Ścieżki te można zmienić w pliku qm.ini lub mqclient.ini . Więcej informacji na ten temat zawiera sekcja Ścieżka wyjścia. W systemach Windows i Linuxdo zmiany ścieżki można użyć programu WebSphere MQ Explorer:

     a. Kliknij prawym przyciskiem myszy nazwę menedżera kolejek
     b. Kliknij opcję **Właściwości ...**
     c. Kliknij opcję **Exits** .
     d. W domyślnym polu ścieżki wyjścia podaj nazwę ścieżki do katalogu, w którym znajduje się program obsługi wyjścia.

   Jeśli wyjście zostanie umieszczone zarówno w określonym katalogu instalacyjnym, jak i w domyślnym katalogu ścieżki, to instalacja programu WebSphere MQ o nazwie określonej w ścieżce jest używana przez program obsługi wyjścia określonego katalogu instalacyjnego. Na przykład wyjście jest umieszczane w /exits/installation2 i w /exits, ale nie w /exits/installation1. Instalacja produktu WebSphere MQ installation2 korzysta z wyjścia z produktu /exits/installation2. Instalacja produktu WebSphere MQ installation1 korzysta z wyjścia z katalogu /exits .
5. Jeśli to konieczne, skonfiguruj wyjście:
   - Instalowalne usługi: “Konfigurowanie usług i komponentów” na stronie 394.
   - Wyjścia funkcji API: “Konfigurowanie wyjść funkcji API” na stronie 404.
   - Wyjścia kanału: “Konfigurowanie wyjść kanału” na stronie 425.
   - Publikowanie wyjść: “Konfigurowanie wyjść publikowania” na stronie 437.
   - Wyjścia przed nawiązaniem połączenia: “Sekcja PreConnect w pliku konfiguracyjnym klienta” na stronie 435.

Instalowalne usługi i komponenty dla systemów UNIX, Linux i Windows

Ta sekcja zawiera wprowadzenie do instalowalnych usług oraz powiązanych z nimi funkcji i komponentów. Interfejs do tych funkcji jest udokumentowany w taki sposób, że użytkownik lub dostawcy oprogramowania mogą dostarczać komponenty.

Główne przyczyny udostępniania usług instalacyjnych produktu WebSphere MQ to:

- Aby zapewnić elastyczność wyboru, czy używać komponentów udostępnianych przez produkty WebSphere MQ , czy też zastąpić je lub rozszerzać innymi produktami.
- Aby umożliwić dostawcom uczestnictwo, udostępniając komponenty, które mogą korzystać z nowych technologii, bez wprowadzania wewnętrznych zmian w produktach WebSphere MQ .
- Aby umożliwić produktowi WebSphere MQ korzystanie z nowych technologii szybciej i taniej, a także produkty wcześniejsze i niższe.

Instalowane usługi i *komponenty usług* są częścią struktury produktu WebSphere MQ . W centrum tej struktury znajduje się część menedżera kolejek, która implementuje funkcję i reguły powiązane z interfejsem kolejki komunikatów (Message Queue Interface-MQI). Ta część centralna wymaga wielu funkcji serwisowych, nazywanych *usługami instalowalnymi*, aby móc wykonywać swoją pracę. Możliwe do zainstalowania usługi to:

- Usługa autoryzacji
- usługa nazw

Każda możliwa do zainstalowania usługa jest pokrewny zestawem funkcji zaimplementowanych przy użyciu co najmniej jednego *komponentów usług*. Każdy komponent jest wywoływany przy użyciu poprawnie zaprojektowanego, publicznie dostępnego interfejsu. Umożliwia to niezależnym dostawcom oprogramowania i innym osobom trzecim udostępnianie instalowalnych komponentów w celu rozszerzenia lub zastąpienia produktów dostarczanych przez produkty WebSphere MQ . Tabela 52 na stronie 387 podsumowuje usługi i komponenty, które mogą być używane.

Tabela 52. Podsumowanie instalowalnych komponentów usług

usługa instalowalna	**Dostarczony komponent**	**Funkcja**	**Wymagania**
Usługa autoryzacji	menedżer uprawnień obiektu (OAM)	Udostępnia sprawdzanie autoryzacji komend i wywołań MQI. Użytkownicy mogą napisać własny komponent w celu rozszerzenia lub zastąpienia OAM. Na przykład, aby sprawdzić, czy identyfikator użytkownika ma uprawnienia do otwierania kolejki.	(Przyjęto odpowiednie narzędzia autoryzacji platformy)
usługa nazw	Brak	Zapewnia obsługę menedżera kolejek w celu wyszukiwania nazwy menedżera kolejek, do którego należy określona kolejka. • Zdefiniowany przez użytkownika **Uwaga:** W kolejkach współużytkowanych musi być ustawiony atrybut *Scope* CELL.	• Menedżer nazw innej firmy lub napisanej przez użytkownika

Interfejs usług instalowalnych jest opisany w sekcji Informacje uzupełniające dotyczące interfejsu usług instalowalnych.

Pisanie komponentu usługi

W tej sekcji opisano relacje między usługami, komponentami, punktami wejścia i kodami powrotu.

Funkcje i komponenty

Każda usługa składa się z zestawu powiązanych funkcji. Na przykład usługa nazw zawiera funkcję dla:

- Wyszukiwanie nazwy kolejki i zwracanie nazwy menedżera kolejek, w którym zdefiniowana jest kolejka
- Wstawianie nazwy kolejki do katalogu usługi
- Usuwanie nazwy kolejki z katalogu usługi

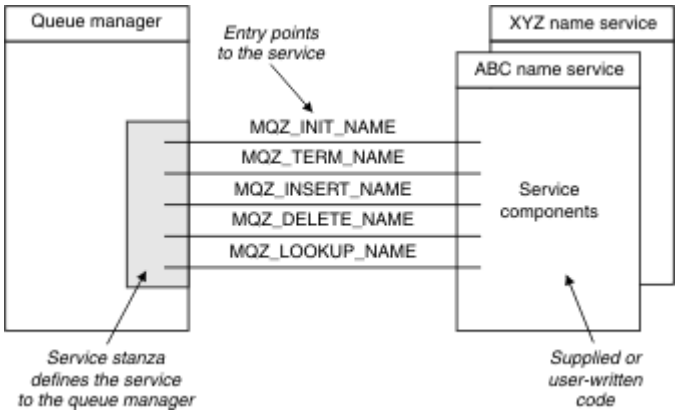
Zawiera on także funkcje inicjowania i zakończenia.

Instalowalna usługa jest udostępniana przez jeden lub więcej komponentów usług. Każdy komponent może wykonywać niektóre lub wszystkie funkcje, które są zdefiniowane dla tej usługi. Na przykład w produkcie WebSphere MQ for AIXdostarczany komponent usługi autoryzacji, OAM, wykonuje wszystkie dostępne funkcje. Więcej informacji na ten temat zawiera sekcja “Interfejs usługi autoryzacji” na stronie

391 . Komponent jest również odpowiedzialny za zarządzanie dowolnymi zasobami lub oprogramowaniem (na przykład katalogiem LDAP), które musi implementować usługę. Pliki konfiguracyjne udostępniają standardowy sposób ładowania komponentu i określania adresów podprogramów funkcjonalnych, które udostępnia.

Rysunek 72 na stronie 388 przedstawia sposób, w jaki usługi i komponenty są powiązane:

- Usługa jest zdefiniowana w menedżerze kolejek przez sekcje w pliku konfiguracyjnym.
- Każda usługa jest obsługiwana przez podany kod w menedżerze kolejek. Użytkownicy nie mogą zmieniać tego kodu i w związku z tym nie mogą tworzyć własnych usług.
- Każda usługa jest implementowana przez jeden lub kilka komponentów. Te komponenty mogą być dostarczane razem z produktem lub użytkownikami. Można wywołać wiele komponentów dla usługi, z których każda obsługuje różne urządzenia w ramach usługi.
- Punkty wejścia łączą komponenty usługi z kodem pomocowym w menedżerze kolejek.



Rysunek 72. Zrozumienie usług, komponentów i punktów wejścia

Punkty wejścia

Każdy komponent usługi jest reprezentowany przez listę adresów punktów wejścia procedur, które obsługują określoną usługę instalowalną. Usługa instalowalna definiuje funkcję, która ma być wykonywana przez każdą procedurę.

Porządkowanie komponentów usługi po ich skonfigurowaniu definiuje kolejność, w jakiej punkty wejścia są wywoływane w próbie spełnienia żądania dla usługi.

W dostarczonym pliku nagłówkowego `cmqzc.h`podane punkty wejścia dla każdej usługi mają przedrostek MQZID_.

Jeśli usługi są obecne, usługi są ładowane w predefiniowanym porządku. Poniższa lista przedstawia usługi oraz kolejność, w jakiej zostały zainicjowane.

1. `NameService`
2. `AuthorizationService`
3. `UserIdentifierService`

`AuthorizationService` jest jedyną usługą, która jest skonfigurowana domyślnie. Skonfiguruj `NameService` i `UserIdentifierService` ręcznie, jeśli mają być używane.

Usługi i komponenty usług mają odwzorowanie jeden-do-jednego lub jeden do wielu. Dla każdej usługi można zdefiniować wiele komponentów usług. W systemach UNIX and Linux wartość Service w sekcji `ServiceComponent` musi być zgodna z wartością nazwy sekcji Service w pliku qm.ini . W systemie Windowswartość klucza rejestru usług produktu `ServiceComponent`musi być zgodna z wartością klucza rejestru nazw i jest zdefiniowana jako: `HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\`*`qmname`*`\` , gdzie *nazwa_menedżera_kolejek* to nazwa menedżera kolejek.

W przypadku systemów UNIX and Linux komponenty usług są uruchamiane w kolejności, w jakiej są one zdefiniowane w pliku qm.ini . W systemie Windows, ponieważ używany jest rejestr systemu Windows, produkt WebSphere MQ wysyła wywołanie **RegEnumKey** , które zwraca wartości w kolejności alfabetycznej. Dlatego w systemie Windows usługi są wywoływane w kolejności alfabetycznej, ponieważ są one zdefiniowane w rejestrze.

Kolejność definicji `ServiceComponent` jest istotna. Ta kolejność narzuca kolejność, w jakiej komponenty są uruchamiane dla danej usługi. Na przykład, `AuthorizationService` w systemie Windows jest skonfigurowany z domyślnym komponentem OAM o nazwie `MQSeries.WindowsNT.auth.service`. Dla tej usługi można zdefiniować dodatkowe komponenty w celu przesłonięcia domyślnego OAM. Jeśli nie określono wartości `MQCACF_SERVICE_COMPONENT` , do przetwarzania żądania używany jest pierwszy komponent napotkany w kolejności alfabetycznej, a nazwa tego komponentu jest używana.

Kody powrotu

Komponenty usług udostępniają menedżerowi kolejek kody powrotu do raportowania w różnych warunkach. Informują one o powodzeniu lub niepowodzeniu operacji, a także wskazują, czy menedżer kolejek ma przejść do następnego komponentu usługi. Ten wskaźnik zawiera osobny parametr *Continuation* .

Dane komponentu

Pojedynczy komponent usługi może wymagać, aby dane były współużytkowane przez różne funkcje. Usługi instalowalne udostępniają opcjonalny obszar danych, który ma być przekazywany w każdym wywołaniu komponentu usługi. Ten obszar danych jest przeznaczony do wyłącznego korzystania z komponentu usługi. Jest on współużytkowany przez wszystkie wywołania danej funkcji, nawet jeśli są one wykonywane z różnych przestrzeni adresowych lub procesów. Gwarantowany jest adresowalny z komponentu usługi za każdym razem, gdy jest wywoływany. Wielkość tego obszaru musi być zadeklarowane w sekcji *`ServiceComponent`* .

Inicjowanie i kończenia komponentów
Użycie opcji inicjowania i zakończenia komponentu.

Gdy wywoływana jest procedura inicjowania komponentu, musi ona wywołać funkcję menedżera kolejek **MQZEP** dla każdego punktu wejścia obsługiwanego przez komponent. **MQZEP** definiuje punkt wejścia do usługi. Przyjmuje się, że wszystkie niezdefiniowane punkty wyjścia mają wartość NULL.

Komponent jest zawsze wywoływany raz przy użyciu podstawowej opcji inicjowania, zanim zostanie wywołany w inny sposób.

Komponent może być wywoływany z dodatkową opcją inicjowania na niektórych platformach. Na przykład może być wywoływany jeden raz dla każdego procesu systemu operacyjnego, wątku lub czynności, do której dostęp jest uzyskiwany przez usługę.

Jeśli używana jest inicjalizacja drugorzędna:

- Komponent może być wywoływany więcej niż jeden raz dla inicjowania dodatkowego. W przypadku każdego takiego wywołania, gdy usługa nie jest już potrzebna, zostanie wysłane zgodne wywołanie dla zakończenia drugorzędnego.

  W przypadku usług nazewnictwa jest to wywołanie MQZ_TERM_NAME.

  W przypadku usług autoryzacji jest to wywołanie MQZ_TERM_AUTHORITY.
- Punkty wejścia muszą być ponownie określone (wywołując MQZEP) za każdym razem, gdy wywoływany jest komponent dla inicjowania podstawowego i dodatkowego.
- Dla komponentu jest używana tylko jedna kopia danych komponentu. Dla każdego inicjowania dodatkowego nie ma innej kopii.
- Komponent nie jest wywoływany w przypadku innych wywołań usługi (z procesu systemu operacyjnego, wątku lub zadania, w zależności od przypadku) przed przeprowadzonym wtórnym zainicjowaniem.

- Komponent musi ustawić parametr *Version* na tę samą wartość dla inicjowania podstawowego i dodatkowego.

Komponent jest zawsze wywoływany po raz pierwszy z podstawową opcją kończenia, gdy nie jest już wymagany. Do tego komponentu nie są wykonywane żadne dodatkowe wywołania.

Komponent jest wywoływany z opcją zakończenia drugorzędnego, jeśli został wywołany w celu zainicjowania dodatkowego.

menedżer uprawnień obiektu (OAM)
Komponent usługi autoryzacji dostarczany wraz z produktami WebSphere MQ jest nazywany menedżerem uprawnień do obiektów (Object Authority Manager-OAM).

Domyślnie program OAM jest aktywny i działa z komendami sterującą **dspmqaut** (uprawnienie do wyświetlania),**dmpmqaut** (uprawnienie do zrzutu) i **setmqaut** (uprawnienie do ustawiania lub resetowania).

Składnia tych komend oraz sposób ich użycia są opisane w sekcji Komendy sterujące.

OAM współpracuje z *jednostką* jednostki głównej lub grupy.

- W systemach UNIX and Linux :
  - Nazwa użytkownika jest identyfikatorem użytkownika lub identyfikatorem powiązanym z programem użytkowym uruchomionym w imieniu użytkownika.
  - Grupa jest kolekcją nazw użytkowników zdefiniowaną w systemie UNIX lub Linux .
  - Autoryzacje mogą być nadawane lub odbierane tylko na poziomie grupy. Żądanie nadania lub unieważnienia uprawnień użytkownika aktualizuje grupę podstawową dla tego użytkownika.
- W systemach Windows:
  - Nazwa użytkownika to identyfikator użytkownika systemu Windows lub identyfikator powiązany z programem użytkowym uruchomionym w imieniu użytkownika.
  - grupa jest grupą Windows.
  - Autoryzacje mogą być nadawane lub odbierane na poziomie głównym lub grupowym.

Podczas wykonywania żądania MQI lub wydania komendy OAM sprawdza autoryzację jednostki powiązanej z operacją, aby sprawdzić, czy może ona:

- Wykonaj żądaną operację.
- Uzyskaj dostęp do określonych zasobów menedżera kolejek.

Usługa autoryzacji umożliwia rozszerzanie lub zastępowanie sprawdzanych uprawnień dla menedżerów kolejek poprzez napisanie własnego komponentu usługi autoryzacji.

usługa nazw
Usługa nazw to instalowalna usługa, która udostępnia obsługę menedżera kolejek w celu wyszukiwania nazwy menedżera kolejek, do którego należy określona kolejka. Z usługi nazw nie można pobrać żadnych innych atrybutów kolejki.

Usługa nazw umożliwia aplikacji otwieranie zdalnych kolejek na potrzeby danych wyjściowych, tak jak w przypadku kolejek lokalnych. Usługa nazw nie jest wywoływana dla obiektów innych niż kolejki.

Uwaga: Dla kolejek zdalnych ***musi*** ich atrybut *Scope* jest ustawiony na wartość CELL.

Gdy aplikacja otwiera kolejkę, wyszukuje nazwę kolejki najpierw w katalogu menedżera kolejek. Jeśli nie znajdzie go tam, wyszukuje tak wiele usług nazw, które zostały skonfigurowane, dopóki nie znajdzie takiego, który rozpoznaje nazwę kolejki. Jeśli żadna nazwa nie rozpozna nazwy, otwarcie nie powiedzie się.

Usługa nazw zwraca menedżera kolejek, który jest właścicielem dla tej kolejki. Menedżer kolejek kontynuuje działanie z żądaniem MQOPEN tak, jakby komenda określiła nazwę kolejki i menedżera kolejek w oryginalnym żądaniu.

Interfejs usługi nazw (NSI) jest częścią środowiska produktu WebSphere MQ .

Jak działa usługa nazw

Jeśli definicja kolejki określa atrybut *Scope* jako menedżer kolejek, to znaczy wartość SCOPE (QMGR) w MQSC, definicja kolejki (wraz ze wszystkimi atrybutami kolejki) jest przechowywana tylko w katalogu menedżera kolejek. Nie może to być zastąpione przez usługę instalowalną.

Jeśli definicja kolejki określa atrybut *Scope* jako komórkę, to znaczy wartość SCOPE (CELL) w MQSC, to definicja kolejki jest ponownie zapisywana w katalogu menedżera kolejek wraz ze wszystkimi atrybutami kolejki. Jednak nazwa kolejki i menedżera kolejek są również przechowywane w usłudze nazw. Jeśli żadna usługa nie jest dostępna, która może przechowywać te informacje, nie można zdefiniować kolejki z komórką produktu *Scope* .

Katalog, w którym przechowywane są informacje, może być zarządzany przez usługę lub usługa może korzystać z usługi bazowej, na przykład katalogu LDAP, w tym celu. W obu przypadkach definicje zapisane w katalogu muszą być trwałe, nawet po zakończeniu działania komponentu i menedżera kolejek, dopóki nie zostaną jawnie usunięte.

Uwaga:

1. Aby wysłać komunikat do lokalnej definicji kolejki hosta zdalnego (z zasięgiem komórki) w innym menedżerze kolejek w komórce katalogu nazw, należy zdefiniować kanał.
2. Nie można pobrać komunikatów bezpośrednio z kolejki zdalnej, nawet jeśli ma on zasięg komórki.
3. Podczas wysyłania do kolejki z zasięgiem komórki CELL nie jest wymagana żadna definicja kolejki zdalnej.
4. Usługa nadawania nazw centralnie definiuje kolejkę docelową, mimo że do docelowego menedżera kolejek i pary definicji kanału potrzebna jest kolejka transmisji. Ponadto kolejka transmisji w systemie lokalnym musi mieć taką samą nazwę, jak menedżer kolejek, do którego należy kolejka docelowa, z zasięgiem komórki w systemie zdalnym.

   Na przykład, jeśli zdalny menedżer kolejek ma nazwę QM01, to kolejka transmisji w systemie lokalnym musi mieć również nazwę QM01.

Interfejs usługi autoryzacji
Usługa autoryzacji udostępnia punkty wejścia do użycia przez menedżer kolejek.

Punkty wejścia są następujące:

MQZ_AUTHENTICATE_USER
Uwierzytelnia identyfikator użytkownika i hasło oraz może ustawiać pola kontekstu tożsamości.

MQZ_CHECK_AUTHORITY
Sprawdza, czy jednostka ma uprawnienia do wykonywania jednej lub większej liczby operacji na określonym obiekcie.

MQZ_CHECK_PRIVILEGED
Sprawdza, czy określony użytkownik jest użytkownikiem uprzywilejowanym.

MQZ_COPY_ALL_AUTHORITY,
Kopiuje wszystkie bieżące autoryzacje, które istnieją dla obiektu, do którego istnieje odwołanie, do innego obiektu.

MQZ_DELETE_AUTHORITY,
Usuwa wszystkie autoryzacje powiązane z określonym obiektem.

MQZ_ENUMERATE_AUTHORITY_DATA
Pobiera wszystkie dane uprawnień, które są zgodne z podanymi kryteriami wyboru.

MQZ_FREE_USER,
Zwalnia przydzielone przydzielone zasoby.

MQZ_GET_AUTHORITY,
Pobiera uprawnienia, które jednostka musi uzyskać, aby uzyskać dostęp do określonego obiektu.

Uprawnienie MQZ_GET_EXPLICIT_AUTHORITY
Uzyskuje uprawnienie do dostępu do określonego obiektu przez nazwaną grupę (ale bez dodatkowego uprawnienia grupy **nobody**) lub uprawnienie, do którego ma dostęp grupa podstawowa nazwanego użytkownika, aby uzyskać dostęp do określonego obiektu.

MQZ_INIT_AUTHORITY,
Inicjuje komponent usługi autoryzacji.

MQZ_INQUIRE
Wysyła zapytania do obsługiwanych funkcji usługi autoryzacji.

MQZ_REFRESH_CACHE
Odśwież wszystkie autoryzacje.

MQZ_SET_AUTHORITY,
Ustawia uprawnienia, które jednostka ma do określonego obiektu.

MQZ_TERM_AUTHORITY,
Przerywa komponent usługi autoryzacji.

Dodatkowo w produkcie WebSphere MQ for Windowsusługa autoryzacji udostępnia następujące punkty wejścia do użycia przez menedżer kolejek:

- **MQZ_CHECK_AUTHORITY_2**
- **MQZ_GET_AUTHORITY_2**
- **MQZ_GET_EXPLICIT_AUTHORITY_2**
- **MQZ_SET_AUTHORITY_2**

Te punkty wejścia obsługują korzystanie z identyfikatora zabezpieczeń systemu Windows (NT SID).

Nazwy te są definiowane jako **typedef**w pliku nagłówkowego `cmqzc.h`, który może być używany do tworzenia prototypów funkcji komponentu.

Funkcja inicjowania (**MQZ_INIT_AUTHORITY**) musi być głównym punktem wejścia dla komponentu. Pozostałe funkcje są wywoływane przez adres punktu wejścia, który funkcja inicjowania dodała do wektora punktu wejścia komponentu.

Interfejs usługi nazw
Usługa nazw udostępnia punkty wejścia do użycia przez menedżer kolejek.

Dostępne są następujące punkty wejścia:

MQZ_INIT_NAME
Zainicjuj komponent usługi nazw.

MQZ_TERM_NAME
Przerwij komponent usługi nazw.

MQZ_LOOKUP_NAME
Wyszukaj nazwę menedżera kolejek dla podanej kolejki.

MQZ_INSERT_NAME
Wstaw wpis zawierający nazwę menedżera kolejek będącego właścicielem podanej kolejki do katalogu używanego przez usługę.

MQZ_DELETE_NAME
Usuń pozycję dla podanej kolejki z katalogu używanego przez usługę.

Jeśli skonfigurowana jest więcej niż jedna usługa nazw:

- W przypadku wyszukiwania funkcja MQZ_LOOKUP_NAME jest wywoływana dla każdej usługi na liście, dopóki nie zostanie rozstrzygnięta nazwa kolejki (chyba że jakikolwiek komponent wskazuje, że wyszukiwanie powinno zostać zatrzymane).
- W przypadku wstawiania funkcja MQZ_INSERT_NAME jest wywoływana dla pierwszej usługi na liście, która obsługuje tę funkcję.
- W przypadku usuwania funkcja MQZ_DELETE_NAME jest wywoływana dla pierwszej usługi na liście, która obsługuje tę funkcję.

Nie istnieje więcej niż jeden komponent, który obsługuje funkcje wstawiania i usuwania. Jednak komponent, który obsługuje wyszukiwanie, jest możliwy do wykonania i może być używany na przykład jako ostatni komponent na liście w celu rozstrzygnięcia dowolnej nazwy, która nie jest znana przez żaden inny komponent usługi nazw, do menedżera kolejek, w którym można zdefiniować nazwę.

W języku programowania C nazwy są definiowane jako typy danych funkcji przy użyciu instrukcji typedef. Można ich użyć do prototypowania funkcji serwisowych, aby upewnić się, że parametry są poprawne.

Plik nagłówkowy, który zawiera wszystkie materiały specyficzne dla usług instalowalnych, to `cmqzc.h` dla języka C.

Oprócz funkcji inicjowania (MQZ_INIT_NAME), która musi być głównym punktem wejścia komponentu, funkcje są wywoływane przez adres punktu wejścia, który został dodany przez funkcję inicjowania, przy użyciu wywołania MQZEP.

Korzystanie z wielu komponentów usług
Dla usługi można zainstalować więcej niż jeden komponent. Umożliwia to komponentom udostępnianie tylko częściowych implementacji usługi, a także oparcie się na innych komponentach w celu udostępnienia pozostałych funkcji.

Przykład użycia wielu komponentów

Załóżmy, że tworzysz dwa komponenty usług nazw o nazwach `ABC_name_serv` i `XYZ_name_serv`.

`ABC_name_serv`
Ten komponent obsługuje wstawienie nazwy lub usunięcie nazwy z katalogu usług, ale nie obsługuje wyszukiwania nazwy kolejki.

`XYZ_name_serv`
Ten komponent obsługuje wyszukiwanie nazwy kolejki, ale nie obsługuje wstawiania nazwy do katalogu usług lub usuwania jej z katalogu usług.

Komponent `ABC_name_serv` przechowuje bazę danych nazw kolejek i korzysta z dwóch prostych algorytmów do wstawiania lub usuwania nazwy z katalogu usług.

Komponent `XYZ_name_serv` korzysta z prostego algorytmu, który zwraca stałą nazwę menedżera kolejek dla każdej nazwy kolejki, z którą jest wywoływany. Nie posiada on bazy danych nazw kolejek i dlatego nie obsługuje funkcji wstawiania i usuwania.

Komponenty są instalowane w tym samym menedżerze kolejek. Sekcje *`ServiceComponent`* są zamawiane w taki sposób, że komponent `ABC_name_serv` jest wywoływany jako pierwszy. Wszystkie wywołania wstawiania lub usuwania kolejki w katalogu komponentu są obsługiwane przez komponent `ABC_name_serv`. Jest to jedyny sposób implementowania tych funkcji. Jednak wywołanie wyszukiwania, którego komponent `ABC_name_serv` nie może rozstrzygnąć, jest przekazywane do komponentu wyszukiwania tylko w komponencie `XYZ_name_serv`. Ten komponent dostarcza nazwę menedżera kolejek z prostego algorytmu.

Pomijanie punktów wejścia podczas korzystania z wielu komponentów

Jeśli użytkownik zdecyduje się na użycie wielu komponentów w celu udostępnienia usługi, może zaprojektować komponent usługi, który nie implementuje określonych funkcji. Środowisko usług instalowalnych nie zawiera żadnych ograniczeń, które można pominąć. Jednak w przypadku konkretnych usług instalowalnych pominięcie jednej lub większej liczby funkcji może być logicznie niespójne z celem usługi.

Przykład punktów wejścia używanych z wieloma komponentami

Tabela 53 na stronie 394 przedstawia przykład usługi nazw możliwych do zainstalowania, dla której zostały zainstalowane dwa komponenty. Każdy z nich obsługuje inny zestaw funkcji powiązanych z daną usługą instalowalną. W przypadku funkcji wstawiania, najpierw wywoływana jest punkt wejścia komponentu ABC. Zakłada się, że punkty wejścia, które nie zostały zdefiniowane dla usługi (za pomocą

komendy **MQZEP**), mają wartość NULL. Punkt wejścia dla inicjowania znajduje się w tabeli, ale nie jest to wymagane, ponieważ inicjowanie jest wykonywane przez główny punkt wejścia komponentu.

Gdy menedżer kolejek musi korzystać z usługi instalowalnej, używa ona punktów wejścia zdefiniowanych dla tej usługi (kolumny w programie Tabela 53 na stronie 394). Po włączeniu każdego z komponentów menedżer kolejek określa adres procedury, która implementuje wymaganą funkcję. Następnie wywołuje on procedurę, jeśli istnieje. Jeśli operacja zakończy się pomyślnie, menedżer kolejek używa dowolnych wyników i informacji o statusie.

Tabela 53. Przykład punktów wejścia dla instalowalnej usługi

Numer funkcji	**Komponent usługi nazw ABC**	**Komponent usługi nazw XYZ**
MQZID_INIT_NAME (inicjowanie)	ABC_initialize ()	XYZ_initialize ()
MQZID_TERM_NAME (Przerwij)	ABC_terminate ()	XYZ_terminate ()
MQZID_INSERT_NAME (Wstaw)	ABC_Insert ()	NULL
MQZID_DELETE_NAME (Usunięcie)	ABC_Delete ()	NULL
MQZID_LOOKUP_NAME (Wyszukiwanie)	NULL	XYZ_Lookup ()

Jeśli procedura nie istnieje, menedżer kolejek będzie powtarzał ten proces dla następnego komponentu na liście. Dodatkowo, jeśli procedura istnieje, ale zwraca kod wskazujący, że nie może wykonać operacji, próba jest kontynuowana z następnym dostępnym komponentem. Podprogramy w komponentach usług mogą zwracać kod wskazujący, że nie należy wykonywać żadnych dalszych prób wykonania operacji.

Konfigurowanie usług i komponentów

Skonfiguruj komponenty usług przy użyciu plików konfiguracyjnych menedżera kolejek, z wyjątkiem systemów Windows , w których każdy menedżer kolejek ma własną sekcję w rejestrze.

1. Dodaj sekcje do pliku konfiguracyjnego menedżera kolejek, aby zdefiniować usługę dla menedżera kolejek i określ położenie modułu.

   Każda używana usługa musi mieć sekcję *Service* , która definiuje usługę dla menedżera kolejek.

   Dla każdego komponentu w ramach usługi musi istnieć sekcja *ServiceComponent* . Identyfikuje ona nazwę i ścieżkę modułu zawierającego kod dla tego komponentu.

   Aby uzyskać więcej informacji, zobacz: “Format sekcji usługi” na stronie 394 i “Format sekcji komponentu usługi” na stronie 395

   Komponent usługi autoryzacji, znany jako Object Authority Manager (OAM), jest dostarczany razem z produktem. Podczas tworzenia menedżera kolejek plik konfiguracyjny menedżera kolejek (lub rejestr w systemach Windows) jest automatycznie aktualizowany w taki sposób, aby obejmował odpowiednie sekcje dla usługi autoryzacji i dla komponentu domyślnego (OAM). W przypadku pozostałych komponentów należy ręcznie skonfigurować plik konfiguracyjny menedżera kolejek.

   Kod dla każdego komponentu usługi jest ładowany do menedżera kolejek po uruchomieniu menedżera kolejek przy użyciu powiązania dynamicznego, w którym jest to obsługiwane na platformie.
2. Zatrzymaj i zrestartuj menedżer kolejek, aby aktywować komponent.

Format sekcji usługi

Sekcja Service zawiera nazwę usługi i liczbę punktów wejścia zdefiniowanych dla usługi.

Format sekcji jest następujący:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
```

gdzie:

<service_name>
Nazwa danej usługi. Jest to zdefiniowane przez usługę.

<entries>
Liczba punktów wejścia zdefiniowanych dla usługi. Dotyczy to punktów wejścia inicjowania i zakończenia.

Format sekcji usług dla systemów Windows
W systemach Windows sekcja *Service* zawiera atrybut SecurityPolicy .

Format sekcji to:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
  SecurityPolicy=<policy>
```

gdzie:

<service_name>
Nazwa danej usługi. Jest to zdefiniowane przez usługę.

<entries>
Liczba punktów wejścia zdefiniowanych dla usługi. Dotyczy to punktów wejścia inicjowania i zakończenia.

<policy>
NTSIDsRequired (identyfikator zabezpieczeń systemu Windows) lub Default. Jeśli wartość NTSIDsRequirednie zostanie określona, zostanie użyta wartość Default . Ten atrybut jest poprawny tylko wtedy, gdy Name ma wartość AuthorizationService.

Patrz także "Konfigurowanie sekcji usług autoryzacji: systemy Windows" na stronie 396.

Format sekcji komponentu usługi

Format sekcji komponentu usługi jest następujący:

```
ServiceComponent:
  Service=<service_name>
  Name=<component_name>
  Module=<module_name>
  ComponentDataSize=<size>
```

gdzie:

<service_name>
Nazwa danej usługi. Musi to być zgodne z Name określonymi w sekcji usługi.

<component_name>
Nazwa opisowa komponentu usługi. Ta wartość musi być unikalna i zawierać tylko znaki, które są poprawne dla nazw obiektów WebSphere MQ (na przykład nazw kolejek). Ta nazwa występuje w komunikatach operatora wygenerowanych przez usługę. Zalecamy stosowanie nazwy zaczynającej się od znaku towarowego firmy lub podobnego łańcucha wyróżniającego.

<module_name>
Nazwa modułu, który ma zawierać kod dla tego komponentu.

<size>
Wielkość (w bajtach) obszaru danych komponentu przekazana do komponentu w każdym wywołaniu. Podaj wartość zero, jeśli dane komponentu nie są wymagane.

Te dwie sekcje mogą występować w dowolnej kolejności, a klucze sekcji pod nimi mogą być również wykonywane w dowolnej kolejności. W przypadku jednej z tych sekcji muszą być obecne wszystkie klucze sekcji. Jeśli klucz sekcji jest zduplikowany, używany jest ostatni klucz sekcji.

Podczas uruchamiania menedżer kolejek przetwarza każdą pozycję komponentu usługi z kolei w pliku konfiguracyjnym. Następnie ładuje określony moduł komponentu, wywołując punkt wejścia komponentu (który musi być pozycją punktu wejścia dla inicjowania komponentu), przekazując mu uchwyt konfiguracji.

Konfigurowanie sekcji usług autoryzacji: systemy UNIX and Linux
W systemach UNIX and Linux każdy menedżer kolejek ma własny plik konfiguracyjny menedżera kolejek.

Na przykład domyślna ścieżka i nazwa pliku konfiguracyjnego menedżera kolejek dla menedżera kolejek QMNAME to `/var/mqm/qmgrs/QMNAME/qm.ini`.

Sekcja *Service* i sekcja *ServiceComponent* dla domyślnego komponentu autoryzacji są automatycznie dodawane do programu `qm.ini` , ale mogą zostać nadpisane przez program `mqsnoaut`. Wszystkie pozostałe sekcje *ServiceComponent* należy dodać ręcznie.

Na przykład następujące sekcje w pliku konfiguracyjnym menedżera kolejek definiują dwa komponenty usługi autoryzacji w produkcie WebSphere MQ dla systemu AIX. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .

```
Service:
   Name=AuthorizationService
   EntryPoints=13

ServiceComponent:
   Service=AuthorizationService
   Name=MQSeries.UNIX.auth.service
   Module=MQ_INSTALLATION_PATH/lib/amqzfu
   ComponentDataSize=0

ServiceComponent:
   Service=AuthorizationService
   Name=user.defined.authorization.service
   Module=/usr/bin/udas01
   ComponentDataSize=96
```

Rysunek 73. Sekcje usługi autoryzacji produktu UNIX and Linux w pliku qm.ini

Sekcja komponentu usługi (`MQSeries.UNIX.auth.service`) definiuje domyślny komponent usługi autoryzacji, czyli OAM. Jeśli ta sekcja zostanie usunięta i zrestartowany zostanie menedżer kolejek, OAM jest wyłączony i nie zostaną przeprowadzone żadne sprawdzenia autoryzacji.

Konfigurowanie sekcji usług autoryzacji: systemy Windows
W produkcie WebSphere MQ for Windows każdy menedżer kolejek ma własną sekcję w rejestrze.

Sekcja *Service* i sekcja *ServiceComponent* dla domyślnego komponentu autoryzacji są automatycznie dodawane do rejestru, ale mogą zostać przesłonięte przy użyciu programu `mqsnoaut`. Wszystkie pozostałe sekcje *ServiceComponent* należy dodać ręcznie.

Atrybut `SecurityPolicy` można również dodać za pomocą usług WebSphere MQ . Atrybut `SsecurityPolicy` ma zastosowanie tylko wtedy, gdy usługa określona w sekcji *Service* jest usługą autoryzacji, tj. domyślną wartością OAM. Atrybut `SecurityPolicy` umożliwia określenie strategii bezpieczeństwa dla każdego menedżera kolejek. Możliwe wartości:

Default
Określ `Default` , jeśli domyślna strategia bezpieczeństwa ma być używana. Jeśli identyfikator zabezpieczeń systemu Windows (NT SID) nie jest przekazywany do OAM dla konkretnego identyfikatora użytkownika, podejmowana jest próba uzyskania odpowiedniego identyfikatora SID przez przeszukanie odpowiednich baz danych zabezpieczeń.

NTSIDsRequired
Wymaga, aby identyfikator SID NT był przekazywany do OAM podczas sprawdzania zabezpieczeń.

Więcej informacji na temat formatu sekcji Service zawiera sekcja “Format sekcji usług dla systemów Windows” na stronie 395. Więcej ogólnych informacji na temat bezpieczeństwa zawiera sekcja Konfigurowanie zabezpieczeń w systemach Windowsi UNIX and Linux.

Sekcja komponentu usługi `MQSeries.WindowsNT.auth.service` definiuje domyślny komponent usługi autoryzacji (OAM). Jeśli ta sekcja zostanie usunięta i zrestartowany zostanie menedżer kolejek, OAM jest wyłączony i nie zostaną przeprowadzone żadne sprawdzenia autoryzacji.

Konfigurowanie sekcji usług nazw: systemy Unix i Linux
W tym miejscu należy umieścić krótki opis; używany w pierwszym akapicie i streszczeniu.

Poniższe przykłady sekcji pliku konfiguracyjnego produktu UNIX and Linux dla usługi nazw określają komponent usługi nazw dostarczony przez (fikcyjne) przedsiębiorstwo ABC.

```
# Stanza for name service
 Service:
    Name=NameService
    EntryPoints=5

# Stanza for name service component, provided by ABC
 ServiceComponent:
    Service=NameService
    Name=ABC.Name.Service
    Module=/usr/lib/abcname
    ComponentDataSize=1024
```

Rysunek 74. Sekcje usługi nazw w pliku qm.ini (dla systemów UNIX and Linux)

Uwaga: W systemach Windows informacje z sekcji usługi nazw są przechowywane w rejestrze.

Odświeżanie OAM po zmianie autoryzacji użytkownika
W produkcie WebSphere MQinformacje o grupie autoryzacji OAM można odświeżać natychmiast po zmianie przypisania do grupy autoryzacji użytkownika, odzwierciedlając zmiany wprowadzone na poziomie systemu operacyjnego bez konieczności zatrzymywania i restartowania menedżera kolejek. Aby wykonać tę komendę, wydaj komendę **REFRESH SECURITY** .

Uwaga: W przypadku zmiany autoryzacji za pomocą komendy `setmqaut` , OAM wprowadza takie zmiany natychmiast.

Menedżery kolejek przechowują dane autoryzacji w kolejce lokalnej o nazwie SYSTEM.AUTH.DATA.QUEUE. Te dane są zarządzane przez `amqzfuma.exe` .

Odsyłacze pokrewne
REFRESH SECURITY

Pisanie i kompilowanie wyjść funkcji API

Wyjścia funkcji API umożliwiają zapisywanie kodu, który zmienia zachowanie wywołań funkcji API produktu WebSphere MQ , takich jak MQPUT i MQGET, a następnie wstawianie tego kodu bezpośrednio przed lub bezpośrednio po tych wywołaniach.

Uwaga: Nieobsługiwane w produkcie WebSphere MQ for z/OS.

Dlaczego należy używać wyjść funkcji API?

Każda z aplikacji ma do wykonania konkretne zadanie, a jego kod powinien wykonać to zadanie tak efektywnie, jak to możliwe. Na wyższym poziomie może być konieczne zastosowanie standardów lub procesów biznesowych do określonego menedżera kolejek dla **wszystkich** aplikacji, które używają tego menedżera kolejek. Bardziej efektywne jest to, aby zrobić to powyżej poziomu poszczególnych aplikacji, a tym samym bez konieczności zmiany kodu każdej aplikacji, której dotyczy problem.

Poniżej przedstawiono kilka sugestii dotyczących obszarów, w których wyjścia funkcji API mogą być przydatne:

- W przypadku *zabezpieczeń*można zapewnić uwierzytelnianie, sprawdzanie, czy aplikacje są autoryzowane do uzyskiwania dostępu do kolejki lub menedżera kolejek. Można również policyjne aplikacje używały interfejsu API, uwierzytelniać poszczególne wywołania API, a nawet parametry, których używają.

- W przypadku *elastyczności*można reagować na szybkie zmiany w środowisku biznesowym bez konieczności zmiany aplikacji polegających na danych w tym środowisku. Można na przykład użyć wyjść funkcji API, które odpowiadają na zmiany stóp procentowych, kursów wymiany walut lub ceny komponentów w środowisku produkcyjnym.
- W przypadku *monitorowania* użycia kolejki lub menedżera kolejek można śledzić przepływ aplikacji i komunikatów, rejestrować błędy w wywołaniach interfejsu API, skonfigurować zapisy kontrolne dla celów rozliczania lub gromadzić statystyki użycia dla celów planowania.

Co się dzieje, gdy kończy się wyjście interfejsu API?

Po zapisaniu programu obsługi wyjścia i zidentyfikowaniu go w produkcie WebSphere MQ, menedżer kolejek automatycznie wywołuje kod wyjścia w zarejestrowanych punktach.

Procedury wyjścia funkcji API do uruchomienia są identyfikowane w sekcjach w systemach IBM i, Windowsi UNIX and Linux . Ten temat obejmuje sekcje w plikach konfiguracyjnych mqs.ini i qm.ini.

Definicja procedur może wystąpić w trzech miejscach:

1. ApiExitCommon, w pliku mqs.ini , identyfikuje podprogramy dla całego WebSphere MQ, które są stosowane podczas uruchamiania menedżerów kolejek. Mogą one zostać przesłonięte przez procedury zdefiniowane dla poszczególnych menedżerów kolejek (patrz pozycja “3” na stronie 398 na tej liście).
2. Szablon ApiExitw pliku mqs.ini identyfikuje podprogramy dla całego produktu WebSphere MQ, kopiowane do zestawu lokalnego ApiExit(patrz pozycja “3” na stronie 398 na tej liście) podczas tworzenia nowego menedżera kolejek.
3. ApiExitLokalne, w pliku qm.ini , identyfikuje podprogramy, które mają zastosowanie do określonego menedżera kolejek.

Po utworzeniu nowego menedżera kolejek definicje szablonów ApiExitw pliku mqs.ini są kopiowane do lokalnych definicji ApiExitw pliku qm.ini dla nowego menedżera kolejek. Po uruchomieniu menedżera kolejek używane są zarówno definicje lokalne ApiExit, jak i ApiExit. Definicje lokalne ApiExitzastępują wspólne definicje ApiExit, jeśli obie identyfikują podprogram o tej samej nazwie. Atrybut Sequence opisany w sekcji “Konfigurowanie wyjść funkcji API” na stronie 404 określa kolejność, w jakiej procedury zdefiniowane w sekcjach są uruchamiane.

Korzystanie z wyjść funkcji API w wielu instalacjach produktu WebSphere MQ

Należy upewnić się, że wyjścia funkcji API napisane dla wcześniejszej wersji produktu WebSphere MQ są używane do pracy ze wszystkimi wersjami, ponieważ zmiany wprowadzone do wyjść w wersji 7.1 mogą nie działać z wcześniejszą wersją. Więcej informacji na temat zmian wprowadzonych do wyjść zawiera sekcja “Pisanie i kompilowanie wyjść i usług instalowalnych” na stronie 384.

Przykłady udostępnione dla wyjść funkcji API amqsaem i amqsaxe odzwierciedlają zmiany wymagane podczas pisania wyjść. Aplikacja kliencka musi upewnić się, że poprawne biblioteki produktu WebSphere MQ , które odpowiadają instalacji menedżera kolejek, z którym powiązana jest aplikacja, są z nim powiązane przed uruchomieniem aplikacji.

Pisanie wyjść funkcji API

Istnieje możliwość pisania wyjść dla każdego wywołania interfejsu API przy użyciu języka programowania C.

Wyjścia są dostępne dla każdego wywołania interfejsu API w następujący sposób:

- MQCB, w celu ponownego zarejestrowania wywołania zwrotnego dla określonego uchwytu obiektu i aktywacji sterowania oraz zmiany w wywołaniu zwrotnym
- MQCTL do wykonywania działań sterujących na uchwytach obiektów otwartych dla połączenia
- MQCONN/MQCONNX służy do udostępniania uchwytu połączenia menedżera kolejek w celu użycia w kolejnych wywołaniach API
- MQDISC, aby odłączyć się od menedżera kolejek
- MQBEGIN, aby rozpocząć globalną jednostkę pracy (UOW)

- MQBACK, aby wycofać UOW
- MQCMIT, aby zatwierdzić jednostkę pracy
- MQOPEN, aby otworzyć zasób WebSphere MQ dla późniejszego dostępu
- MQCLOSE, aby zamknąć zasób WebSphere MQ , który wcześniej został otwarty dla dostępu
- MQGET-umożliwia pobranie komunikatu z kolejki, która została wcześniej otwarta na potrzeby dostępu.
- MQPUT1, aby umieścić komunikat w kolejce
- Wywołanie MQPUT w celu umieszczenia komunikatu w kolejce, która została wcześniej otwarta dla dostępu
- MQINQ, aby uzyskać informacje na temat atrybutów zasobu WebSphere MQ , który został wcześniej otwarty na potrzeby dostępu
- MQSET służy do ustawiania atrybutów kolejki, która została wcześniej otwarta na potrzeby dostępu.
- MQSTAT, w celu pobrania informacji o statusie
- MQSUB, aby zarejestrować subskrypcję aplikacji w określonym temacie
- MQSUBRQ, aby złożyć wniosek o subskrypcję

Funkcja MQ_CALLBACK_EXIT udostępnia funkcję wyjścia, która ma zostać wykonana przed i po przetworzeniu wywołania zwrotnego. Więcej informacji na ten temat zawiera sekcja Callback-MQ_CALLBACK_EXIT.

W przypadku wyjść funkcji API wywołania przyjmują postać ogólną:

```
MQ_call_EXIT (parameters, context, ApiCallParameters)
```

gdzie *call* jest nazwą wywołania MQI bez przedrostka MQ ; na przykład PUT, GET. *parameters* sterują funkcją wyjścia, zapewniając przede wszystkim komunikację między wyjściem a zewnętrznymi blokami sterującą MQAXP (struktura parametru wyjścia funkcji API) i MQAXC (struktura kontekstu wyjścia funkcji API). *context* opisuje kontekst, w którym wywołano wyjście funkcji API, a program *ApiCallParameters* reprezentuje parametry wywołania MQI.

Aby ułatwić zapisanie wyjścia funkcji API, udostępniono przykładowe wyjście amqsaxe0.c; to wyjście generuje pozycje śledzenia do określonego pliku. Tego przykładu można użyć jako punktu początkowego podczas zapisywania wyjść. Więcej informacji na temat korzystania z wyjścia przykładowego zawiera sekcja “Przykładowy program obsługi wyjścia funkcji API” na stronie 117.

Więcej informacji na temat wywołań wyjścia funkcji API, zewnętrznych bloków sterujących i powiązanych tematów zawiera sekcja Informacje dodatkowe o wyjściu interfejsu API.

Ogólne informacje na temat pisania, kompilowania i konfigurowania wyjścia zawiera sekcja “Pisanie i kompilowanie wyjść i usług instalowalnych” na stronie 384.

Korzystanie z uchwytów komunikatów w wyjściach API

Użytkownik może sterować dostępem do właściwości komunikatu, do których ma dostęp wyjście funkcji API. Właściwości są powiązane z uchwytem ExitMsg. Właściwości ustawione w wyjściu umieszczonym są ustawiane w umieszczonym komunikacie, ale właściwości pobrane w wyjściu pobierania nie są zwracane do aplikacji.

Podczas rejestrowania funkcji wyjścia MQ_INIT_EXIT za pomocą wywołania MQI MQXEP z parametrem **Function** ustawionym na wartość MQXF_INIT i **ExitReason** ustawionym na wartość MQXR_CONNECTION, należy przejść do struktury MQXEPO jako parametru **ExitOpts** . Struktura MQXEPO zawiera pole ExitProperties , które określa zestaw właściwości, które mają zostać udostępnione dla wyjścia. Jest on określany jako łańcuch znaków reprezentujący przedrostek właściwości, który odpowiada nazwie folderu MQRFH2 .

Każde wyjście funkcji API odbiera strukturę MQAXP, która zawiera pole ExitMsgHandle. To pole jest ustawiane na wartość wygenerowaną przez produkt WebSphere MQ i jest specyficzna dla połączenia.

W związku z tym uchwyt nie zmienia się między wyjściami API tego samego lub innego typu w tym samym połączeniu.

W przypadku wartości MQ_PUT_EXIT lub MQ_PUT1_EXIT z produktem **ExitReason** z tabeli MQXR_BEFORE, czyli wyjścia interfejsu API wykonywanego przed umieszczniem komunikatu, wszystkie właściwości (inne niż właściwości deskryptora komunikatu) powiązane z uchwytem ExitMsgpo zakończeniu wyjścia są ustawiane w umieszczonym komunikacie. Aby temu zapobiec, należy ustawić parametr Uchwyt ExitMsgna wartość MQHM_NONE. Można także podać inny uchwyt komunikatu.

W przypadku operacji MQ_GET_EXIT uchwyt ExitMsgjest czyszczony z właściwości i zapełniany właściwościami określonymi w polu ExitProperties , gdy zarejestrowano wartość MQ_INIT_EXIT, inne niż właściwości deskryptora komunikatu. Właściwości te nie są dostępne dla aplikacji pobierających. Jeśli aplikacja pobierający określiła uchwyt komunikatu w polu MQGMO (Get message options), to wszystkie właściwości powiązane z tym uchwytem, w tym właściwości deskryptora komunikatu, są dostępne dla wyjścia funkcji API. Aby zapobiec zapełnieniu uchwytu ExitMsgwłaściwościami, należy ustawić go na wartość MQHM_NONE.

Dostępny jest przykładowy program amqsaem0.cilustrujący użycie uchwytów komunikatów w wyjściach API.

Kompilowanie wyjść funkcji API

Po zapisaniu wyjścia należy skompilować go i połączyć w następujący sposób.

W poniższych przykładach przedstawiono komendy używane dla przykładowego programu opisanego w sekcji “Przykładowy program obsługi wyjścia funkcji API” na stronie 117. W przypadku platform innych niż Windows można znaleźć przykładowy kod wyjścia funkcji API w programie *MQ_INSTALLATION_PATH*/samp oraz skompilowaną i dowiązaną bibliotekę współużytkowaną w programie *MQ_INSTALLATION_PATH*/samp/bin. W systemach Windows przykładowy kod wyjścia funkcji API można znaleźć w programie *MQ_INSTALLATION_PATH*\Tools\c\Samples. *MQ_INSTALLATION_PATH* Reprezentuje katalog, w którym zainstalowano produkt WebSphere MQ .

Uwaga dla użytkowników:

1. Wskazówki dotyczące programowania aplikacji 64-bitowych są wymienione w sekcji Standardy kodowania na platformach 64-bitowych

Dzięki wprowadzeniu klientów Multicast wyjścia funkcji API i wyjścia konwersji danych muszą być możliwe do uruchomienia po stronie klienta, ponieważ niektóre komunikaty mogą nie przechodzić przez menedżer kolejek. Następujące biblioteki są teraz częścią pakietów klienta, a także pakietów serwera:

Tabela 54. Biblioteki, które znajdują się teraz w pakietach klienta i serwera

System operacyjny	**Biblioteki**
Windows	32 bity & 64 bit: mqm.dll & mqm.pdb
Linux & HP-UX	32 bity & 64 bit: libmqm.so & libmqm_r.so
AIX	32 bity & 64 bit: libmqm.a & libmqm_r.a
Solaris	32 bity & 64 bit: libmqm.so

Kompilowanie wyjść funkcji API w systemach Unix i Linux

Przykłady sposobu kompilowania wyjść funkcji API w systemach UNIX i Linux .

Na wszystkich platformach punktem wejścia do modułu jest MQStart.

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

W systemie AIX

Skompiluj kod źródłowy wyjścia funkcji API, wydając jedną z następujących komend:

Aplikacje 32-bitowe

Niewątkowa

```
cc -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Wątkowy

```
xlc_r -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe_r \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Aplikacje 64-bitowe

Niewątkowa

```
cc -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Wątkowy

```
xlc_r -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe_r \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Na platformie HP-UX Itanium

Aplikacje 32-bitowe

Niewątkowa

Skompiluj kod źródłowy procedury zewnętrznej API:

```
c89 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Odsyłacz do kodu źródłowego wyjścia funkcji API

```
ld +b: -b amqsaxe.o +ee MQStart -o /var/mqm/exits/amqsaxe
rm amqsaxe.o
```

Wątkowy

Skompiluj kod źródłowy procedury zewnętrznej API:

```
c89 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Odsyłacz do kodu źródłowego wyjścia funkcji API

```
ld +b: -b amqsaxe.o +ee MQStart -o /var/mqm/exits/amqsaxe_r
rm amqsaxe.o
```

Aplikacje 64-bitowe

Niewątkowa

Skompiluj kod źródłowy procedury zewnętrznej API:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Odsyłacz do kodu źródłowego wyjścia funkcji API

```
ld -b amqsaxe.o +ee MQStart -o  /var/mqm/exits64/amqsaxe
rm amqsaxe.o
```

Wątkowy

Skompiluj kod źródłowy procedury zewnętrznej API:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Odsyłacz do kodu źródłowego wyjścia funkcji API

```
ld -b amqsaxe.o +ee MQStart -o  /var/mqm/exits64/amqsaxe_r
rm amqsaxe.o
```

wł.Linux

Skompiluj kod źródłowy wyjścia funkcji API, wydając jedną z następujących komend:

31-bitowe aplikacje

Niewątkowa

```
 gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Wątkowy

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -IMQ_INSTALLATION_PATH/inc
```

Aplikacje 32-bitowe

Niewątkowa

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Wątkowy

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -IMQ_INSTALLATION_PATH/inc
```

Aplikacje 64-bitowe

Niewątkowa

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Wątkowy

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe_r amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

W systemie Solaris

Skompiluj kod źródłowy wyjścia funkcji API, wydając jedną z następujących komend:

Aplikacje 32-bitowe

platforma SPARC

```
cc -xarch=v8plus -KPIC -mt -G -o /var/mqm/exits/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/32 -lsocket -lnsl -ldl
```

platformax86-64

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/32 -lsocket -lnsl -ldl
```

Aplikacje 64-bitowe

platforma SPARC

```
cc -xarch=v9 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/64 -lsocket -lnsl -ldl
```

platformax86-64

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/64 -lsocket -lnsl -ldl
```

W systemach Windows

Skompiluj i dowiąż przykładowy program obsługi wyjścia funkcji API `amqsaxe0.c`w systemie Windows .

Plik manifestu jest opcjonalnym dokumentem XML zawierającym wersję lub inne informacje, które mogą być osadzone w skompilowanej aplikacji lub bibliotece DLL.

Jeśli taki dokument nie jest taki sam, należy pominąć parametr -manifest *manifest.file* w komendzie **mt** .

Dostosuj komendy podane w przykładach w programie Rysunek 75 na stronie 403 lub Rysunek 76 na stronie 403 , aby skompilować i dowiązać `amqsaxe0.c` w systemie Windows . Komendy działają z programem Microsoft Visual Studio 2005, 2008 lub 2010. W przykładach założono, że katalog WebSphere MQ `C:\Program Files\IBM\WebSphere MQ\tools\c\samples` jest katalogiem bieżącym.

32-bitowa

```
cl /c /nologo /MD /Foamqsaxe0.obj amqsaxe0.c

link /nologo /dll /def:amqsaxe.def

amqsaxe0.obj \
     /manifest /out:amqsaxe.dll

mt -nologo -manifest amqsaxe.dll.manifest \
   -outputresource:amqsaxe.dll;2
```

Rysunek 75. Skompiluj i dowiąże `amqsaxe0.c` w 32-bitowym systemie Windows

64 bity

```
cl /c /nologo /MD /Foamqsaxe0.obj amqsaxe0.c

link /nologo /dll /def:amqsaxe.def \
     /libpath:..\..\lib64 \


amqsaxe0.obj /manifest /out:amqsaxe.dll

mt -nologo -manifest amqsaxe.dll.manifest \
   -outputresource:amqsaxe.dll;2
```

Rysunek 76. Skompiluj i dowiąże `amqsaxe0.c` w 64-bitowym systemie Windows

Pojęcia pokrewne

“Przykładowy program obsługi wyjścia funkcji API” na stronie 117

Przykładowe wyjście interfejsu API generuje dane śledzenia MQI w pliku określonym przez użytkownika z przedrostkiem zdefiniowanym w zmiennej środowiskowej MQAPI_TRACE_LOGFILE.

Konfigurowanie wyjść funkcji API

Aby włączyć wyjścia funkcji API, należy skonfigurować produkt IBM WebSphere MQ , zmieniając informacje o konfiguracji.

Aby zmienić informacje konfiguracyjne, należy zmienić sekcje definiujące procedury obsługi wyjścia i sekwencję, w której są one uruchamiane. Informacje te mogą zostać zmienione w następujący sposób:

- Korzystanie z produktu IBM WebSphere MQ Explorer (w systemach Windows i Linux (platformyx86 i x86-64))
- Korzystanie z komendy **amqmdain** (w systemie Windows)
- Korzystanie bezpośrednio z plików mqs.ini i qm.ini (w systemach Windows, UNIX and Linux).

  Plik mqs.ini zawiera informacje istotne dla wszystkich menedżerów kolejek w określonym węźle. Można go znaleźć w katalogu `/var/mqm` w katalogu UNIX and Linux oraz w katalogu WorkPath określonym w kluczu `HKLM\SOFTWARE\IBM\WebSphere MQ` w systemach Windows .

  Plik qm.ini zawiera informacje istotne dla konkretnego menedżera kolejek. Dla każdego menedżera kolejek znajduje się jeden plik konfiguracyjny menedżera kolejek, który znajduje się w katalogu głównym drzewa katalogów zajmowanego przez menedżer kolejek. Na przykład ścieżka i nazwa pliku konfiguracyjnego dla menedżera kolejek o nazwie QMNAME to:

  W systemach UNIX and Linux :

  ```
  /var/mqm/qmgrs/QMNAME/qm.ini
  ```

  W systemach Windows :

  ```
  C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\qm.ini
  ```

  Przed edytowaniem pliku konfiguracyjnego należy utworzyć kopię zapasową w taki sposób, aby możliwe było przywrócenie kopii, jeśli zajdziesz taka potrzeba.

  Pliki konfiguracyjne można edytować:

  – Automatycznie, przy użyciu komend, które zmieniają konfigurację menedżerów kolejek w węźle
  – Ręcznie, przy użyciu standardowego edytora tekstu

  Jeśli w atrybucie pliku konfiguracyjnego zostanie ustawiona niepoprawna wartość, wartość ta zostanie zignorowana i zostanie wyświetlony komunikat operatora wskazujący problem. (Efekt jest taki sam, jak brak w całości atrybutu.)

Sekcje do skonfigurowania

Sekcje, które muszą zostać zmienione, są następujące:

ApiExitCommon

Zdefiniowane w pliku mqs.ini i w IBM WebSphere MQ Explorer na stronie właściwości produktu IBM WebSphere MQ w sekcji Exits.

Po uruchomieniu menedżera kolejek atrybuty znajdujące się w tej sekcji są odczytyywane, a następnie nadpisywane przez wyjścia interfejsu API zdefiniowane w pliku qm.ini.

ApiExitTemplate

Zdefiniowane w pliku mqs.ini i w IBM WebSphere MQ Explorer na stronie właściwości produktu IBM WebSphere MQ w sekcji Exits.

Gdy tworzony jest dowolny menedżer kolejek, atrybuty w tej sekcji są kopiowane do nowo utworzonego pliku qm.ini w sekcji ApiExitLocal.

ApiExitLocal

Zdefiniowane w pliku qm.ini i w IBM WebSphere MQ Explorer na stronie właściwości menedżera kolejek, w sekcji Exits.

Po uruchomieniu menedżera kolejek wyjścia interfejsu API zdefiniowane w tym miejscu przesłonią wartości domyślne zdefiniowane w pliku mqs.ini.

Atrybuty sekcji

- Podaj nazwę wyjścia funkcji API, używając następującego atrybutu:

  Name=nazwa_ApiExit_name
  Opisowa nazwa wyjścia interfejsu API przekazana do niego w polu ExitInfow polu Nazwa struktury MQAXP.

  Ta nazwa musi być unikalna, nie dłuższa niż 48 znaków i zawierać tylko poprawne znaki dla nazw obiektów IBM WebSphere MQ (na przykład nazwy kolejek).

- Zidentyfikuj moduł i punkt wejścia kodu wyjścia API, który ma być uruchamiany przy użyciu następujących atrybutów:

  Function=nazwa_funkcji
  Nazwa punktu wejścia funkcji do modułu zawierającego kod wyjścia API. Ten punkt wejścia jest funkcją MQ_INIT_EXIT.

  Wielkość tego pola jest ograniczona do wartości MQ_EXIT_NAME_LENGTH.

  Module=nazwa_modułu
  Moduł zawierający kod wyjścia API.

  Jeśli w polu znajduje się pełna nazwa ścieżki do modułu, jest ona używana w takiej postaci.

  Jeśli to pole zawiera tylko nazwę modułu, to moduł znajduje się przy użyciu atrybutu `ExitsDefaultPath` w pliku `ExitPath` w pliku qm.ini.

  Na platformach obsługujących oddzielne biblioteki wielowątkowe należy udostępnić wersję modułu wyjścia API, która nie jest wielowątkowa i jest wielowątkowa. Wersja wielowątkowa musi mieć przyrostek `_r` . Wersja wielowątkowa kodu pośredniczącego aplikacji IBM WebSphere MQ niejawnie dołącza produkt `_r` do podanej nazwy modułu przed jego załadowaniu.

  Długość tego pola jest ograniczona do maksymalnej długości ścieżki, którą obsługuje platforma.

- Opcjonalnie przekaz dane za pomocą wyjścia za pomocą następującego atrybutu:

  Data=nazwa_danych
  Dane, które mają być przekazywane do wyjścia funkcji API w polu ExitData struktury MQAXP.

  Jeśli zostanie podany ten atrybut, początkowe i końcowe odstępy zostaną usunięte, pozostały łańcuch zostanie obcięty do 32 znaków, a wynik zostanie przekazany do wyjścia. Jeśli ten atrybut zostanie pominięty, do wyjścia zostanie przekazana domyślna wartość 32 odstępów.

  Maksymalna długość tego pola to 32 znaki.

- Zidentyfikuj sekwencję tego wyjścia w odniesieniu do innych wyjść, korzystając z następującego atrybutu:

  Sequence=numer_sekwencji_
  Sekwencja, w której to wyjście funkcji API jest wywoływane w stosunku do innych wyjść funkcji API. Wyjście z niskim numerem kolejnym jest wywoływane przed wyjściem z wyższym numerem kolejnym. Nie ma potrzeby, aby numeracja sekwencji wyjść była jednoznaczna. Sekwencja 1, 2, 3 ma ten sam wynik co sekwencja 7, 42, 1096. Jeśli dwa wyjścia mają ten sam numer kolejny, menedżer kolejek decyduje o tym, który z nich powinien najpierw zadzwonić. Można określić, który został wywołany po zdarzeniu, umieszczając czas lub znacznik w obszarze ExitChain(Łańcuch ExitChain) wskazany przez ExitChainAreaPtr w produkcie MQAXP lub przez zapisanie własnego pliku dziennika.

  Ten atrybut jest niepodpisaną wartością liczbową.

Sekcje przykładowe

Przykładowy plik mqs.ini zawiera następujące sekcje:

ApiExitTemplate
Ta sekcja definiuje wyjście z nazwą opisową `OurPayrollQueueAuditor`, nazwą modułu `auditor`i numerem kolejnym 2. Do wyjścia jest przekazywana wartość danych o wartości 123.

ApiExitCommon
Ta sekcja definiuje wyjście z nazwą opisową `MQPoliceman`, nazwą modułu `tmqp`i numerem kolejnym 1. Przekazywane dane są instrukcjami (`CheckEverything`).

```
mqs.ini

   ApiExitTemplate:
      Name=OurPayrollQueueAuditor
      Sequence=2
      Function=EntryPoint
      Module=/usr/ABC/auditor
      Data=123
   ApiExitCommon:
      Name=MQPoliceman
      Sequence=1
      Function=EntryPoint
      Module=/usr/MQPolice/tmqp
      Data=CheckEverything
```

Poniższy przykładowy plik qm.ini zawiera lokalną definicję wyjścia ApiExitwyjścia o nazwie opisowej `ClientApplicationAPIchecker`, nazwie modułu `ClientAppChecker`i numerze kolejnym 3.

```
qm.ini

   ApiExitLocal:
      Name=ClientApplicationAPIchecker
      Sequence=3
      Function=EntryPoint
      Module=/usr/Dev/ClientAppChecker
      Data=9.20.176.20
```

Programy obsługi wyjścia kanału dla kanałów przesyłania komunikatów

Ta kolekcja tematów zawiera informacje na temat programów obsługi wyjścia kanału WebSphere MQ dla kanałów przesyłania komunikatów.

Agenty kanałów komunikatów (MCAs) mogą również wywoływać wyjścia konwersji danych. Aby uzyskać więcej informacji na temat pisania wyjść konwersji danych, zobacz: “Pisanie wyjść konwersji danych” na stronie 426.

Niektóre z tych informacji mają zastosowanie również do wyjść z kanałów MQI, które łączą klienty MQI produktu WebSphere MQ z menedżerami kolejek. Więcej informacji na ten temat zawiera sekcja Programy obsługi wyjścia kanału dla kanałów MQI.

Programy obsługi wyjścia kanału są wywoływane w zdefiniowanych miejscach w przetwarzaniu wykonywanego przez programy MCA.

Niektóre z tych programów obsługi wyjścia użytkownika działają w pary komplementarne. Na przykład, jeśli program obsługi wyjścia użytkownika jest wywoływany przez wysyłający agent MCA w celu zaszyfrowania komunikatów do transmisji, proces komplementarny musi działać w odbiorczym końcu, aby odwrócić ten proces.

W programie Tabela 55 na stronie 406 wyświetlane są typy wyjść kanału, które są dostępne dla każdego typu kanału.

Tabela 55. Wyjścia kanału dostępne dla każdego typu kanału

Typ kanału	Wyjście komunikatu	Wyjście z ponowienia komunikatu	Wyjście odbierania	Wyjście zabezpieczeń	Wyjście wysyłania	Wyjście automatycznej definicji
Kanał nadawcy	Tak		Tak	Tak	Tak	

Tabela 55. Wyjścia kanału dostępne dla każdego typu kanału (kontynuacja)

Typ kanału	Wyjście komunikatu	Wyjście z ponowienia komunikatu	Wyjście odbierania	Wyjście zabezpieczeń	Wyjście wysyłania	Wyjście automatycznej definicji
Kanał serwera	Tak		Tak	Tak	Tak	
Kanał wysyłający klastry	Tak		Tak	Tak	Tak	Tak
Kanał odbiorcy	Tak	Tak	Tak	Tak	Tak	Tak
Kanał requestera	Tak	Tak	Tak	Tak	Tak	
Kanał odbierający klastry	Tak	Tak	Tak	Tak	Tak	Tak
Kanał połączenia klienckiego			Tak	Tak	Tak	
Kanał połączenia z serwerem			Tak	Tak	Tak	Tak

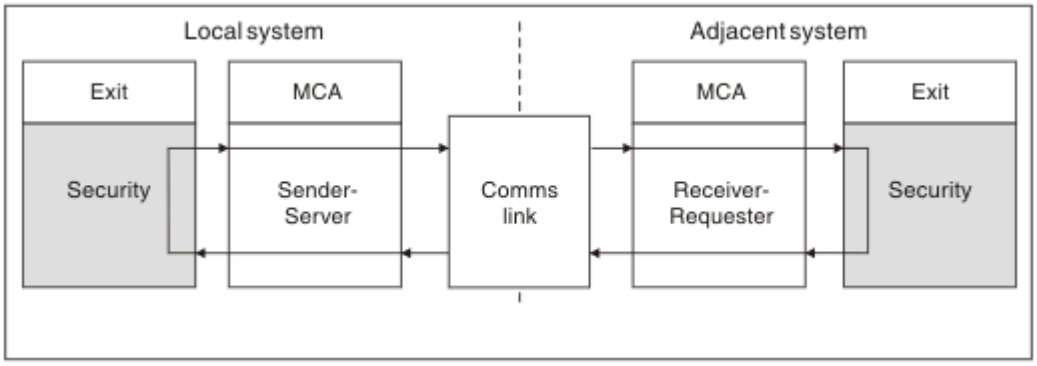
Jeśli użytkownik zamierza uruchomić wyjścia kanału na kliencie, nie można użyć zmiennej środowiskowej MQSERVER. Zamiast tego należy utworzyć i odwołać się do tabeli definicji kanału klienta (CCDT) zgodnie z opisem w sekcji Tabela definicji kanału klienta .

Przegląd przetwarzania

Przegląd informacji o tym, w jaki sposób MCAs korzysta z programów obsługi wyjścia.

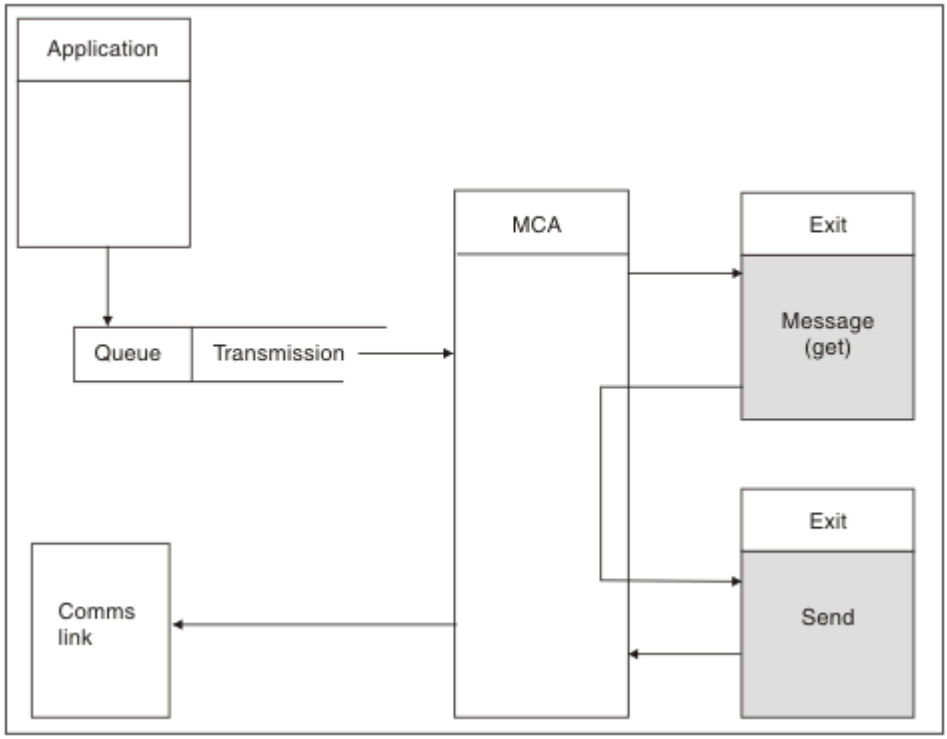
Podczas uruchamiania, MCAs wymieniają okno dialogowe uruchamiania w celu zsynchronizowania przetwarzania. Następnie przełączają się na wymianę danych, która obejmuje wyjścia zabezpieczeń. Te wyjścia muszą zakończyć się pomyślnie, aby faza uruchamiania zakończyła się pomyślnie i aby umożliwić przesyłanie komunikatów.

Faza sprawdzania zabezpieczeń jest pętlą, jak pokazano na rysunku Rysunek 77 na stronie 407.

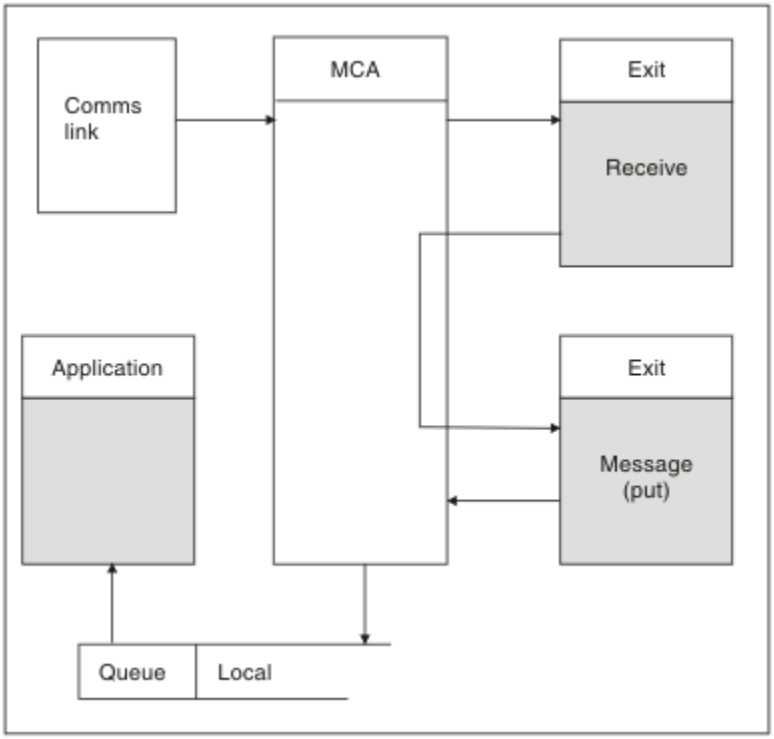


Rysunek 77. Pętla wyjścia zabezpieczeń

Podczas fazy przesyłania komunikatów wysyłający agent MCA pobiera komunikaty z kolejki transmisji, wywołuje wyjście komunikatu, wywołuje wyjście wysyłania, a następnie wysyła komunikat do odbierającego agenta MCA, jak to pokazano na rysunku Rysunek 78 na stronie 408.



Rysunek 78. Przykład wyjścia wysyłania na końcu kanału komunikatów nadawcy



Rysunek 79. Przykład wyjścia odbierania na końcu kanału komunikatów kanału komunikatów

Odbierający agent MCA odbierze komunikat z łącza komunikacyjnego, wywołuje wyjście odbierania, wywołuje wyjście komunikatu, a następnie umieszcza komunikat w kolejce lokalnej, jak to pokazano na Rysunek 79 na stronie 408. (Wyjście odbierania może być wywoływane więcej niż jeden raz przed wywołaniem wyjścia komunikatu).

Pisanie programów obsługi wyjścia kanału

W celu ułatwienia pisania programów obsługi wyjścia kanału można użyć następujących informacji.

Programy zewnętrzne i programy obsługi wyjścia kanału mogą korzystać ze wszystkich wywołań MQI, z wyjątkiem sytuacji, w których są one oznaczone w kolejnych sekcjach. W przypadku produktu MQ V7 i nowszego struktura MQCXP w wersji 7 i nowszej zawiera uchwyt połączenia hConn, który może zostać użyty zamiast wywołania MQCONN. W przypadku wcześniejszych wersji, aby uzyskać uchwyt połączenia, należy wydać komendę MQCONN, nawet jeśli zostanie zwrócone ostrzeżenie MQRC_ALREADY_CONNECTED, ponieważ sam kanał jest połączony z menedżerem kolejek.

Należy pamiętać, że wyjście kanału musi być wątkowo bezpieczne.

W przypadku wyjść z kanałów połączenia klienckiego menedżer kolejek, z którym program obsługi wyjścia próbuje nawiązać połączenie, zależy od sposobu, w jaki połączenie zostało nawiązane. Jeśli wyjście zostało połączone z programem MQM.LIB i nie określono nazwy menedżera kolejek w wywołaniu MQCONN, program zewnętrzny próbuje połączyć się z domyślnym menedżerem kolejek w systemie. Jeśli wyjście zostało połączone z programem MQM.LIB i określa nazwę menedżera kolejek, który został przekazany do wyjścia za pomocą pola QMgrName produktu MQCD, wyjście próbuje nawiązać połączenie z tym menedżerem kolejek. Jeśli wyjście było połączone z MQIC.LIB lub dowolna inna biblioteka, wywołanie MQCONN nie powiedzie się, niezależnie od tego, czy zostanie określona nazwa menedżera kolejek.

Należy unikać zmiany stanu transakcji powiązanej z przekazanej wartości hConn w wyjściu kanału. Nie wolno używać komend MQCMIT, MQBACK ani MQDISC z kanałem hConn, a komendy MQBEGIN nie można używać z określeniem kanału hConn.

Jeśli parametr MQCONNX jest używany z parametrem MQCNO_HANDLE_SHARE_BLOCK lub MQCNO_HANDLE_SHARE_NO_BLOCK w celu utworzenia nowego połączenia z produktem IBM WebSphere MQ , należy upewnić się, że połączenie jest poprawnie zarządzane i poprawnie rozłącza się z menedżerem kolejek. Na przykład: wyjście kanału, które tworzy nowe połączenie z menedżerem kolejek w każdym wywołaniu bez rozłączania, powoduje, że uchwyty połączeń są tworzone i zwiększane w liczbie wątków agenta.

Wyjście jest uruchamiane w tym samym wątku co sam agent MCA i korzysta z tego samego uchwytu połączenia. Dlatego jest on uruchamiany w obrębie tej samej jednostki pracy, co agent MCA, a wszystkie wywołania wykonane w punkcie synchronizacji są zatwierdzane lub wycofane przez kanał na końcu zadania wsadowego.

Oznacza to, że wyjście komunikatu kanału może wysyłać komunikaty powiadomienia, które są zatwierdzane tylko do tej kolejki po zatwierdzeniu zadania wsadowego zawierającego oryginalną wiadomość. Możliwe jest więc wystawianie wywołań MQI punktów synchronizacji z wyjścia komunikatów kanału.

Wyjście kanału może zmienić pola na zmaterializowanych tabelach MQCD. Zmiany te nie są jednak wykonywane, z wyjątkiem wymienionych okoliczności. Jeśli program obsługi wyjścia kanału zmienia pole w strukturze danych MQCD, nowa wartość jest ignorowana przez proces kanału IBM WebSphere MQ . Nowa wartość pozostaje jednak na zmaterializowanych tabelach MQCD i jest przekazywana do pozostałych wyjść w łańcuchu wyjścia i do dowolnej konwersacji współużytkującej instancję kanału. Więcej informacji na ten temat zawiera sekcja Zmiana pól MQCD w wyjściu kanału .

Ponadto w przypadku programów napisanych w języku C funkcja biblioteki C bez ponownego wejścia nie może być używana w programie obsługi wyjścia kanału.

Jeśli jednocześnie używane jest wiele bibliotek wyjścia kanału, mogą wystąpić problemy na niektórych platformach UNIX and Linux , jeśli kod dla dwóch różnych wyjść zawiera identycznie nazwane funkcje. Po załadowaniu wyjścia kanału program ładujący dynamiczny rozstrzywa nazwy funkcji w bibliotece wyjścia na adresy, na których załadowana jest biblioteka. Jeśli dwie biblioteki wyjścia definiują oddzielne

funkcje, które zdarzają się, że mają identyczne nazwy, ten proces rozstrzygania może niepoprawnie przetłumaczać nazwy funkcji jednej biblioteki w celu użycia funkcji innego. Jeśli ten problem wystąpi, należy określić dla konsolidatora, że musi on wyeksportować tylko wymagane funkcje wyjścia i wywołania MQStart, ponieważ te funkcje nie mają wpływu na działanie. Inne funkcje muszą być podane w lokalnej widoczności, tak aby nie były używane przez funkcje poza własną biblioteką wyjścia. Aby uzyskać więcej informacji, zapoznaj się z dokumentacją urządzenia konsolidatora.

Wszystkie wyjścia są wywoływane za pomocą struktury parametru wyjścia kanału (MQCXP), struktury definicji kanału (MQCD), przygotowanego buforu danych, parametru długości danych i parametru długości buforu. Długość buforu nie może być przekroczona:

- W przypadku wyjść komunikatów należy zezwolić na wysłanie największego komunikatu, który ma być wysyłany przez kanał, powiększony o długość struktury MQXQH.
- W przypadku wyjścia wysyłania i odbierania największy bufor musi być dostępny w następujący sposób:

  LU 6.2
  32 kB

  TCP:
  32 kB

  Uwaga: Maksymalna długość użyteczna może być mniejsza o 2 bajty niż długość tej długości. Sprawdź wartość zwróconej w polu MaxSegment, aby uzyskać szczegółowe informacje. Więcej informacji na temat długości MaxSegmentmożna znaleźć w sekcji MaxSegmentLength.

  NetBIOS:
  64 kB

  SPX:
  64 kB

  Uwaga: Wyjścia odbierania w kanałach nadawczych i wyjściach nadawczych w kanałach odbiorczych korzystają z buforów o wielkości 2 kB dla TCP.
- W przypadku wyjść zabezpieczeń rozproszona funkcja kolejkowania przydziela bufor o wielkości 4000 bajtów.

Dopuszczalne jest zwrócenie innego bufora, wraz z odpowiednimi parametrami. Więcej informacji na ten temat zawiera sekcja “Programy obsługi wyjścia kanału dla kanałów przesyłania komunikatów” na stronie 406 .

Pisanie programów obsługi wyjścia kanału w systemach Windows, UNIX and Linux
Poniższe informacje ułatwiają pisanie programów obsługi wyjścia kanału w systemach Windowsi UNIX and Linux .

Wykonaj instrukcje opisane w sekcji “Pisanie i kompilowanie wyjść i usług instalowalnych” na stronie 384. Użyj następujących informacji szczegółowych dotyczących wyjścia kanału, w stosownych przypadkach:

Wyjście musi być zapisane w języku C, a w systemie Windows jest to biblioteka DLL.

Zdefiniuj fikcyjną procedurę MQStart () w wyjściu i określ `MQStart` jako punkt wejścia w bibliotece. Rysunek 80 na stronie 410 pokazuje, jak skonfigurować wpis do programu:

```
#include <cmqec.h>

void MQStart() {;} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQCXP  pChannelExitParms,
                           PMQCD   pChannelDefinition,
                           PMQLONG pDataLength,
                           PMQLONG pAgentBufferLength,
                           PMQVOID pAgentBuffer,
                           PMQLONG pExitBufferLength,
                           PMQPTR  pExitBufferAddr)
{
... Insert code here
}
```

Rysunek 80. Przykładowy kod źródłowy dla wyjścia kanału

Podczas zapisywania wyjść kanału dla systemu Windows za pomocą programu Visual C + + należy napisać własny plik DEF . Przykład pokazany w programie Rysunek 81 na stronie 411. Więcej informacji na temat pisania programów obsługi wyjścia kanału znajduje się w sekcji “Pisanie programów obsługi wyjścia kanału” na stronie 409.

```
EXPORTS
ChannelExit
```

Rysunek 81. Przykładowy plik DEF dla systemu Windows

Programy obsługi wyjścia zabezpieczeń kanału
Za pomocą programów obsługi wyjścia zabezpieczeń można sprawdzić, czy partner na drugim końcu kanału jest autentyczny. Jest to znane jako uwierzytelnianie. Aby określić, że kanał musi używać wyjścia bezpieczeństwa, należy określić nazwę wyjścia w polu SCYEXIT definicji kanału.

Uwaga: Uwierzytelnianie można również uzyskać za pomocą rekordów uwierzytelniania kanału. Rekordy uwierzytelniania kanału zapewniają dużą elastyczność w zapobieganiu dostępowi do menedżerów kolejek od określonych użytkowników i kanałów oraz w odwzorowywanie zdalnych użytkowników na identyfikatory użytkowników produktu IBM WebSphere MQ . Obsługa protokołu SSL i TLS jest również udostępniana przez produkt IBM WebSphere MQ w celu uwierzytelniania użytkowników oraz zapewnienia szyfrowania i integralności danych w celu sprawdzenia danych. Więcej informacji na temat protokołu SSL i TLS można znaleźć w sekcji Obsługa protokołu SSL i TLS w programie WebSphere MQ. Jeśli jednak nadal wymagane jest bardziej wyrafinowane (lub różne) formy przetwarzania zabezpieczeń, a także inne typy kontroli i kontekstu zabezpieczeń, należy rozważyć zapisanie wyjść zabezpieczeń.

W przypadku wyjść zabezpieczeń zapisanych przed produktem IBM WebSphere MQ Version 7.1 warto zauważyć, że wcześniejsze wersje produktu IBM WebSphere MQ odpytywały bazowego dostawcę bezpiecznych gniazd (np. GSKit) w celu określenia nazwy wyróżniającej (SSLPEER) i wystawcy certyfikatu partnera zdalnego (SSLCERTI). W produkcie IBM WebSphere MQ Version 7.1 dodano obsługę dla zakresu nowych atrybutów zabezpieczeń. W celu uzyskania dostępu do tych atrybutów program IBM WebSphere MQ Version 7.1 uzyskuje kodowanie DER certyfikatu i używa go do określenia nazwy wyróżniającej podmiotu i wystawcy. Atrybuty nazwy wyróżniającej podmiotu i wystawcy są wyświetlane w następujących atrybutach statusu kanału:

- SSLPEER (selektor PCF MQCACH_SSL_SHORT_PEER_NAME)
- SSLCERTI (selektor PCF MQCACH_SSL_CERT_ISSUER_NAME)

Wartości te są zwracane przez komendy statusu kanału, jak również dane przekazywane do wyjść zabezpieczeń kanału na liście, jak pokazano poniżej:

- MQCD SSLPeerNamePtr
- MQCXP SSLRemCertIssNamePtr

W programie IBM WebSphere MQ Version 7.1atrybut SERIALNUMBER jest również dołączany do nazwy wyróżniającej podmiotu i zawiera numer seryjny certyfikatu partnera zdalnego. Niektóre atrybuty nazwy wyróżniającej są zwracane w innej kolejności niż w poprzednich wersjach. W rezultacie skład pól SSLPEER i SSLCERTI jest zmieniany w produkcie Version 7.1 z poprzednich wersji i dlatego zaleca się, aby wszystkie wyjścia zabezpieczeń lub aplikacje zależne od tych pól były badane i aktualizowane.

Istniejące filtry nazw węzłów produktu WebSphere MQ określone za pomocą pola SSLPEER definicji kanału nie będą miały wpływu i będą działały w taki sam sposób, jak we wcześniejszych wersjach. Wynika to z faktu, że algorytm uzgadniania nazw węzłów sieci WebSphere MQ został zaktualizowany w celu przetwarzania istniejących filtrów SSLPEER bez konieczności zmiany definicji kanału. Zmiana ta najprawdopodobniej ma wpływ na wyjścia zabezpieczeń i aplikacje zależne od wartości nazwy wyróżniającej podmiotu i nazwy wyróżniającej wystawcy zwracanych przez interfejs programistyczny PCF.

Wyjście zabezpieczeń może być zapisane w języku C lub Java.

Programy obsługi wyjścia zabezpieczeń kanału są wywoływane w następujących miejscach w cyklu przetwarzania agenta MCA:

- W momencie inicjowania i zakończenia MCA.

- Natychmiast po zakończeniu początkowego negocjowania danych przy uruchamianiu kanału. Odbiornik lub serwer końca kanału może inicjować wymianę komunikatów zabezpieczeń ze zdalnym końcem, udostępniając komunikat, który ma zostać dostarczony do wyjścia zabezpieczeń na zdalnym końcu. Może to również pogorszać się w tym zakresie. Program obsługi wyjścia jest ponownie uruchamiany w celu przetworzenia dowolnego komunikatu zabezpieczeń odebranego ze zdalnego zakończenia.
- Natychmiast po zakończeniu początkowego negocjowania danych przy uruchamianiu kanału. Koniec kanału wysyłającego lub requestera przetwarza komunikat bezpieczeństwa odebrany ze zdalnego punktu końcowego lub inicjuje wymianę zabezpieczeń, gdy zdalny koniec nie może zostać. Program obsługi wyjścia jest ponownie uruchamiany w celu przetworzenia wszystkich kolejnych komunikatów bezpieczeństwa, które mogą zostać odebrane.

Kanał requestera nigdy nie jest wywoływany za pomocą wywołania MQXR_INIT_SEC. Kanał powiadamia serwer o tym, że ma program obsługi wyjścia zabezpieczeń, a następnie serwer ma możliwość zainicjowania wyjścia zabezpieczeń. Jeśli nie ma on jednego, informuje o tym requester, a przepływ o zerowej długości jest zwracany do programu obsługi wyjścia.

Uwaga: Należy unikać wysyłania komunikatów bezpieczeństwa o zerowej długości.

Przykłady danych wymienianych przez programy obsługi wyjścia zabezpieczeń są ilustrowane w rysunkach Rysunek 82 na stronie 413 za pośrednictwem Rysunek 85 na stronie 415. W tych przykładach przedstawiono sekwencję zdarzeń, które występują w przypadku wyjścia zabezpieczeń odbiornika, a także wyjście zabezpieczeń nadawcy. Kolejne wiersze w liczbach przedstawiają upływ czasu. W niektórych przypadkach zdarzenia u odbiorcy i nadawcy nie są skorelowane, a więc mogą wystąpić w tym samym czasie lub w różnych momentach. W innych przypadkach zdarzenie w jednym programie obsługi wyjścia skutkuje zdarzeniem uzupełniającym występującym później w drugim programie obsługi wyjścia. Na przykład w produkcie Rysunek 82 na stronie 413:

1. Odbiornik i nadawca są wywoływane za pomocą wywołania MQXR_INIT, ale te wywołania nie są skorelowane i mogą być wykonywane w tym samym czasie lub w różnych momentach.
2. Odbiornik jest wywoływany przy użyciu wywołania MQXR_INIT_SEC, ale zwraca MQXCC_OK, które nie wymaga żadnych dodatkowych zdarzeń przy wyjściu nadawcy.
3. Nadawca jest wywoływany przy użyciu wywołania MQXR_INIT_SEC. Nie jest to skorelowane z wywołaniem odbiornika z MQXR_INIT_SEC. Nadawca zwraca wartość MQXCC_SEND_SEC_MSG, co powoduje, że zdarzenie uzupełniające jest w wyjściu odbiornika.
4. Odbiornik jest następnie wywoływany za pomocą MQXR_SEC_MSG i zwraca MQXCC_SEND_SEC_MSG, co powoduje, że zdarzenie uzupełniające jest wykonywane przy wyjściu nadawcy.
5. Nadawca jest następnie wywoływany za pomocą MQXR_SEC_MSG i zwraca MQXCC_OK, które nie wymaga żadnego dodatkowego zdarzenia w wyjściu odbiornika.

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_OK	
	Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_OK
Message transfer begins	

Rysunek 82. Wymiana zainicjowana przez nadawcę z umową

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_OK	
	Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_OK	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_SUPPRESS_FUNCTION
Channel closes	
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Rysunek 83. Wymiana zainicjowana przez nadawcę bez umowy

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_OK	
Message transfer begins	
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Rysunek 84. Wymiana inicjowana przez odbiorcę z umową

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_OK
Invoked with MQXR_SEC_MSG Responds with MQXCC_SUPPRESS_FUNCTION	
Channel closes	

Rysunek 85. Wymiana inicjowana przez odbiorcę bez zgody

Program obsługi wyjścia zabezpieczeń kanału jest przekazywany do buforu agenta zawierającego dane bezpieczeństwa, z wyjątkiem wszystkich nagłówków transmisji, generowanych przez wyjście zabezpieczeń. Dane te mogą być dowolnymi danymi, dzięki czemu albo zakończenie kanału będzie w stanie przeprowadzić sprawdzanie poprawności zabezpieczeń.

Program obsługi wyjścia zabezpieczeń zarówno w wysyłającym, jak i odbierającym końcu kanału komunikatów może zwrócić jeden z dwóch kodów odpowiedzi do dowolnego wywołania:

- Wymiana zabezpieczeń zakończyła się bez błędów
- Pomijaj kanał i zamknij

Uwaga:

1. Wyjścia zabezpieczeń kanału zwykle działają w parach. Po zdefiniowaniu odpowiednich kanałów należy się upewnić, że zgodne programy obsługi wyjścia są nazwane dla obu końców kanału.
2. W programie IBM iprogramy obsługi wyjścia zabezpieczeń, które zostały skompilowane z użyciem "Użyj uprawnień adoptowanych" (USEADPAUT = *YES), mogą adoptować uprawnienia QMQM lub QMQMADM. Należy zwrócić uwagę, że wyjście nie korzysta z tej funkcji, aby utworzyć zagrożenie bezpieczeństwa systemu.
3. W przypadku kanału SSL, na którym drugi koniec kanału udostępnia certyfikat, wyjście zabezpieczeń odbiera nazwę wyróżniającą tematu tego certyfikatu w polu MQCD, do którego dostęp jest uzyskiwany przez parametr SSLPeerNamePtr i nazwę wyróżniającą wystawcy w polu MQCXP, do którego dostęp uzyskuje się za pomocą komendy SSLRemCertIssNamePtr. Użycie, do którego można umieścić tę nazwę, to:
   - Aby ograniczyć dostęp przez kanał SSL.
   - Aby zmienić MQCD.MCAUserIdentifier oparty na nazwie.

Pojęcia pokrewne

Rekordy uwierzytelniania kanału

Pojęcia związane z protokołem SSL (Secure Sockets Layer) i TLS (Transport Layer Security)

Zapisywanie wyjścia zabezpieczeń

Wyjście zabezpieczeń można zapisać, korzystając z kodu szkieletu wyjścia zabezpieczeń.

Rysunek 86 na stronie 416 ilustruje sposób pisania wyjścia zabezpieczeń.

```
void MQENTRY MQStart() {;}
void MQENTRY EntryPoint (PMQVOID pChannelExitParms,
                         PMQVOID pChannelDefinition,
                         PMQLONG pDataLength,
                         PMQLONG pAgentBufferLength,
                         PMQVOID pAgentBuffer,
                         PMQLONG pExitBufferLength,
                         PMQPTR  pExitBufferAddr)
{
  PMQCXP pParms = (PMQCXP)pChannelExitParms;
  PMQCD  pChDef = (PMQCD)pChannelDefinition;
  /* TODO: Add Security Exit Code Here */

}
```

Rysunek 86. Kod szkieletu wyjścia zabezpieczeń

Musi istnieć standardowy punkt wejścia MQStart produktu WebSphere MQ , ale nie jest on wymagany do wykonywania żadnej funkcji. Nazwa funkcji (w tym przykładzieEntryPoint) może zostać zmieniona, ale funkcja musi zostać wyeksportowana, gdy biblioteka jest kompilowana i dowiązana. Podobnie jak w poprzednim przykładzie, wskaźniki pChannelExitParms muszą być rzutowane na wartość PMQCXP, a definicja pChannelmusi być rzutowana na wartość PMQCD. Ogólne informacje na temat wywoływania wyjść kanału oraz korzystania z parametrów zawiera sekcja MQ_CHANNEL_EXIT. Parametry te są używane w wyjściu zabezpieczeń w następujący sposób:

PMQVOID pChannelExitParms
Wejście/wyjście

Wskaźnik do struktury MQCXP-rzutowanie na wartość PMQCXP w celu uzyskania dostępu do pól. Ta struktura jest używana do komunikowania się między wyjściem a MCA. Następujące pola w MQCXP są szczególnie interesujące dla Exits Security:

ExitReason
Informuje o tym, że program Security Exit jest bieżącym stanem w wymianie zabezpieczeń i jest używany podczas podejmowania decyzji o działaniu, które ma zostać wykonane.

ExitResponse
Odpowiedź na MCA, która dyktuje kolejny etap w wymianie bezpieczeństwa.

ExitResponse2
Dodatkowe opcje sterujące umożliwiające zarządzanie sposobem interpretowania odpowiedzi przez agenta MCA w odpowiedzi na wyjście zabezpieczeń.

Obszar ExitUser
16 bajtów (maksimum) pamięci masowej, które mogą być używane przez wyjście zabezpieczeń do utrzymywania stanu między wywołaniami.

ExitData
Zawiera dane określone w polu SCYDATA definicji kanału (32 bajty dopełnione z prawej strony spacjami).

Definicja PMQVOID pChannel
Wejście/wyjście

Wskaźnik do struktury MQCD-rzutowanie na wartość PMQCD w celu uzyskania dostępu do pól. Ten parametr zawiera definicję kanału. Następujące pola na zmaterializowanych tabelach MQCD są szczególnie interesujące dla Exits Security:

ChannelName
Nazwa kanału (20 bajtów dopełniona z prawej strony znakami odstępu).

ChannelType
Kod definiujący typ kanału.

Identyfikator użytkownika MCA
Ta grupa trzech pól jest inicjowana do wartości pola MCAUSER określonego w definicji kanału. Każdy identyfikator użytkownika określony przez wyjście zabezpieczeń w tych polach jest używany do kontroli dostępu (nie dotyczy kanałów SDR, SVR, CLNTCONN lub CLUSSDR).

MCAUserIdentifier
Pierwsze 12 bajtów identyfikatora dopełnione jest z prawej strony spacjami.

LongMCAUserIdPtr
Wskaźnik do buforu zawierającego identyfikator pełnej długości (brak gwarantowanej wartości NULL został zakończony) ma pierwszeństwo przed identyfikatorem MCAUserIdentifier.

LongMCAUserIdLength
Długość łańcucha wskazywaanego przez LongMCAUserIdPtr -musi być ustawiona, jeśli ustawiona jest wartość LongMCAUserIdPtr .

Identyfikator użytkownika zdalnego
Ma zastosowanie tylko do par kanałów CLNTCONN/SVRCONN. Jeśli nie zdefiniowano wyjścia zabezpieczeń CLNTCONN, to te trzy pola są inicjowane przez agenta MCA klienta, tak więc mogą zawierać identyfikator użytkownika ze środowiska klienta, który może być używany przez wyjście zabezpieczeń SVRCONN do uwierzytelniania oraz podczas określania identyfikatora użytkownika MCA. Jeśli zdefiniowano wyjście zabezpieczeń CLNTCONN, to pola te nie są inicjowane i można je ustawić za pomocą komendy CLNTCONN Security Exit (Wyjście zabezpieczeń CLNTCONN) lub komunikaty zabezpieczeń mogą być używane do przekazywania identyfikatora użytkownika z klienta do serwera.

Identyfikator RemoteUser
Pierwsze 12 bajtów identyfikatora dopełnione z prawej strony spacjami.

LongRemoteUserIdPtr
Wskaźnik do buforu zawierającego identyfikator pełnej długości (brak gwarantowanej wartości NULL został zakończony) ma wyższy priorytet niż identyfikator RemoteUser.

LongRemoteDługośćUserId
Długość łańcucha wskazanego przez parametr LongRemoteUserIdPtr-musi być ustawiona, jeśli ustawiona jest wartość LongRemoteUserIdPtr.

Długość PMQLONG pData
Wejście/wyjście

Wskaźnik do tabeli MQLONG. Zawiera długość dowolnego wyjścia zabezpieczeń zawartego w polu AgentBuffer po wywołaniu procedury zewnętrznej zabezpieczeń. Musi być ustawiony przez wyjście zabezpieczeń na długość dowolnego komunikatu wysyłanego w AgentBuffer lub ExitBuffer.

PMQLONG pAgentBufferLength
wejściowe,

Wskaźnik do tabeli MQLONG. Długość danych zawartych w AgentBuffer przy wywoływaniu wyjścia zabezpieczeń.

Bufor PMQVOID pAgent
Wejście/wyjście

W przypadku wywołania wyjścia zabezpieczeń wskazuje to na dowolny komunikat wysłany z wyjścia partnera. Jeśli parametr ExitResponse2 w strukturze MQCXP ma ustawioną flagę MQXR2_USE_AGENT_BUFFER (wartość domyślna), to wyjście zabezpieczeń musi ustawić ten parametr w taki sposób, aby wskazywało na wysyłane dane komunikatu.

PMQLONG pExitBufferLength
Wejście/wyjście

Wskaźnik do tabeli MQLONG. Ten parametr jest inicjowany na 0 przy pierwszym wywołaniu procedury zewnętrznej zabezpieczeń, a zwracana wartość jest utrzymywana między wywołaniami wyjścia zabezpieczeń podczas wymiany zabezpieczeń.

PMQPTR pExitBufferAddr
Wejście/wyjście

Ten parametr jest inicjowany do pustego wskaźnika przy pierwszym wywołaniu procedury zewnętrznej zabezpieczeń, a zwracana wartość jest utrzymywana między wywołaniami wyjścia zabezpieczeń podczas wymiany zabezpieczeń. Jeśli opcja MQXR2_USE_EXIT_BUFFER jest ustawiona w elemencie ExitResponse2 w strukturze MQCXP, to wyjście zabezpieczeń musi ustawić ten parametr w taki sposób, aby wskazywało na wysyłane dane komunikatu.

Różnice w zachowaniu między wyjściami bezpieczeństwa zdefiniowanymi w parach kanału CLNTCONN/SVRCONN i innymi parami kanału
Wyjścia bezpieczeństwa mogą być definiowane na wszystkich typach kanałów. Jednak zachowanie wyjść zabezpieczeń zdefiniowanych w parach kanału CLNTCONN/SVRCONN różni się nieco od wyjść zabezpieczeń zdefiniowanych w innych parach kanałów.

Wyjście zabezpieczeń na kanale CLNTCONN może ustawić identyfikator zdalnego użytkownika w definicji kanału na potrzeby przetwarzania przez partnerskie wyjście SVRCONN lub dla autoryzacji OAM, jeśli nie zdefiniowano wyjścia zabezpieczeń SVRCONN, a pole MCAUSER parametru SVRCONN nie jest ustawione.

Jeśli nie zdefiniowano wyjścia zabezpieczeń CLNTCONN, to identyfikator zdalnego użytkownika w definicji kanału jest ustawiany na identyfikator użytkownika ze środowiska klienta (który może być pusty) przez agenta MCA klienta.

Wymiana zabezpieczeń między wymianą zabezpieczeń zdefiniowana w parze kanałów CLNTCONN i SVRCONN zakończy się pomyślnie, gdy wyjście zabezpieczeń SVRCONN zwróci wartość ExitResponse w przypadku wywołania MQXCC_OK. Wymiana zabezpieczeń między innymi parami kanału zakończy się pomyślnie, gdy wyjście zabezpieczeń, które zainicjował wymianę, zwróci wartość ExitResponse (MQXCC_OK).

Jednak kod MQXCC_SEND_AND_REQUEST_SEC_MSG ExitResponse może być używany w celu wymuszenia kontynuacji wymiany zabezpieczeń: jeśli ExitResponse z MQXCC_SEND_AND_REQUEST_SEC_MSG jest zwracana przez komendę CLNTCONN lub SVRCONN Security Exit, wyjście partnerskie musi odpowiedzieć, wysyłając komunikat bezpieczeństwa (a nie MQXCC_OK lub odpowiedź null) lub kanał kończy działanie. W przypadku wyjść zabezpieczeń zdefiniowanych dla innych typów kanału odpowiedź ExitResponse MQXCC_OK zwrócona w odpowiedzi na wartość MQXCC_SEND_AND_REQUEST_SEC_MSG z partnerskiego wyjścia zabezpieczeń powoduje kontynuację wymiany zabezpieczeń tak, jakby została zwrócona odpowiedź o wartości NULL, a nie podczas kończenia działania kanału.

Wyjście zabezpieczeń SSPI
Produkt WebSphere MQ for Windows udostępnia wyjście zabezpieczeń, które udostępnia uwierzytelnianie dla kanałów produktu WebSphere MQ przy użyciu interfejsu SSPI (Security Services Programming Interface). Interfejs SSPI udostępnia zintegrowane zabezpieczenia systemu Windows.

To wyjście zabezpieczeń jest przeznaczone zarówno dla klienta WebSphere MQ , jak i dla serwera WebSphere MQ .

Pakiety zabezpieczeń są ładowane z pliku security.dll lub z pliku secur32.dll. Te biblioteki DLL są dostarczane wraz z systemem operacyjnym.

Uwierzytelnianie jednokierunkowe jest udostępniane w systemie Windowsza pomocą usług uwierzytelniania NTLM. Uwierzytelnianie dwukierunkowe jest udostępniane w systemie Windows 2000 za pomocą usług uwierzytelniania Kerberos .

Program obsługi wyjścia zabezpieczeń jest dostarczany w formacie źródłowym i obiektowym. Można użyć kodu wynikowego, który jest, lub użyć kodu źródłowego jako punktu wyjścia do tworzenia własnych programów obsługi wyjścia użytkownika. Więcej informacji na temat korzystania z obiektu lub kodu źródłowego wyjścia zabezpieczeń SSPI zawiera sekcja “Korzystanie z wyjścia zabezpieczeń SSPI w systemach Windows” na stronie 173

Programy obsługi wyjścia wysyłania i odbierania kanału
Istnieje możliwość użycia wyjść wysyłania i odbierania w celu wykonania takich zadań, jak kompresja danych i dekompresja danych. Istnieje możliwość określenia listy programów obsługi wyjścia wysyłania i odbierania, które mają być uruchamiane w ramach dziedziczenia.

Programy obsługi wyjścia wysyłania i odbierania kanału są wywoływane w następujących miejscach w cyklu przetwarzania agenta MCA:

- Programy obsługi wyjścia wysyłania i odbierania są wywoływane do inicjowania w inicjacji MCA, a także do zakończenia w momencie zakończenia agenta MCA.
- Program obsługi wyjścia wysyłania jest wywoływany na jednym lub drugim końcu kanału, w zależności od zakończenia transmisji dla jednego przesyłania komunikatów, bezpośrednio przed wysłaniem transmisji przez łącze. Uwaga 4 wyjaśnia, dlaczego wyjścia są dostępne w obu kierunkach, mimo że kanały komunikatów wysyłają wiadomości tylko w jednym kierunku.
- Program obsługi wyjścia odbierania jest wywoływany na jednym lub drugim końcu kanału, w zależności od zakończenia transmisji dla jednego przesyłania komunikatów, natychmiast po odebraniu transmisji z łącza. Uwaga 4 wyjaśnia, dlaczego wyjścia są dostępne w obu kierunkach, mimo że kanały komunikatów wysyłają wiadomości tylko w jednym kierunku.

Może istnieć wiele transmisji dla jednego transferu komunikatów, a może istnieć wiele iteracji programów obsługi wyjścia wysyłania i odbierania, zanim komunikat osiągnie wyjście komunikatu na końcu odbierającym.

Programy obsługi wyjścia wysyłania i odbierania kanału są przekazywane do buforu agenta zawierającego dane transmisji jako wysłane lub odebrane z łącza komunikacyjnego. W przypadku programów obsługi wyjścia wysyłania pierwszych 8 bajtów buforu jest zarezerwowanych do użycia przez agenta MCA i nie może być zmieniane. Jeśli program zwraca inny bufor, to te pierwsze 8 bajtów musi znajdować się w nowym buforze. Format danych prezentowanych w programach obsługi wyjścia nie jest zdefiniowany.

Dobry kod odpowiedzi musi zostać zwrócony przez programy obsługi wyjścia wysyłania i odbierania. Każda inna odpowiedź powoduje nieprawidłowe zakończenie MCA (abend).

Uwaga: Nie należy wywoływać wywołania MQGET, MQPUT lub MQPUT1 w punkcie synchronizacji z poziomu wyjścia wysyłania lub odbierania.

Uwaga:

1. Wyjścia wysyłania i odbierania zwykle pracują w parach. Na przykład wyjście wysyłania może kompresować dane, a wyjście odbierania jest dekompresowane, albo wyjście wysyłania może szyfrować dane, a wyjście odbierania odszyfrować je. Po zdefiniowaniu odpowiednich kanałów należy się upewnić, że zgodne programy obsługi wyjścia są nazwane dla obu końców kanału.
2. Jeśli kompresja jest włączona dla kanału, wyjścia są przekazywane skompresowane dane.
3. Programy zewnętrzne wysyłania i odbierania kanału mogą być wywoływane dla segmentów komunikatów innych niż dla danych aplikacji, na przykład komunikatów o statusie. Nie są one wywoływane podczas uruchamiania okna dialogowego ani fazy sprawdzania zabezpieczeń.
4. Mimo że kanały komunikatów wysyłają komunikaty tylko w jednym kierunku, dane sterowania kanałami, takie jak bity serca i koniec przetwarzania wsadowego, przepływają w obu kierunkach, a wyjścia te są dostępne w obu kierunkach, również. Jednak niektóre z początkowych przepływów danych uruchamiania kanału są zwolnione z przetwarzania przez żaden z wyjść.
5. Istnieją okoliczności, w których można wywołać wyjścia wysyłania i odbierania z sekwencji, na przykład w przypadku uruchamiania serii programów obsługi wyjścia lub w przypadku uruchamiania wyjść zabezpieczeń. Następnie, po pierwszym wywołaniu wyjścia odbierania w celu przetwarzania danych, może on odbierać dane, które nie zostały przekazane przez odpowiednie wyjście wysyłania. Jeśli program obsługi wyjścia odbierania wykonał operację, na przykład dekompresję, nie sprawdzając przy tym, czy jest ona wymagana, wyniki byłyby nieoczekiwane.

   Należy zakodować wyjścia wysyłania i odbierania w taki sposób, aby wyjście odbierania było możliwe sprawdzenie, czy dane, które jest odbierany, zostały przetworzone przez odpowiednie wyjście wysyłania. Zalecanym sposobem działania jest zakodowanie programów obsługi wyjścia w taki sposób, aby:

   - Wyjście wysyłania ustawia wartość dziewiątego bajtu danych na 0 i przenosi wszystkie dane wzdłuż 1 bajtu, przed wykonaniem operacji. (Pierwsze 8 bajtów jest zarezerwowane do użycia przez agenta MCA).
   - Jeśli wyjście odbierania odbiera dane, które mają wartość 0 w bajcie 9, to wie, że dane pochodzą z wyjścia wysyłania. Usuwa wartość 0, wykonuje operację komplementarną i przenosi dane wynikowe z powrotem o 1 bajt.
   - Jeśli wyjście odbierania odbiera dane, które mają coś innego niż 0 w bajcie 9, zakłada, że wyjście wysyłania nie zostało uruchomione, a następnie wysyła dane z powrotem do programu wywołującego w niezmienionej postaci.

   W przypadku korzystania z wyjść zabezpieczeń, jeśli kanał jest zakończony przez wyjście zabezpieczeń, możliwe jest wywołanie wyjścia wysyłania bez odpowiadającego mu wyjścia odbierania. Jednym ze sposobów zapobiegania temu problemowi jest zakodowanie wyjścia zabezpieczeń w celu ustawienia flagi, na przykład MQCD.SecurityUserData lub MQCD.SendUserData, gdy wyjście decyduje o zakończeniu działania kanału. Następnie wyjście wysyłania musi sprawdzić to pole i przetwarzać dane tylko wtedy, gdy flaga nie jest ustawiona. To sprawdzenie zapobiega niepotrzebnej zmianie danych przez wyjście wysyłania, a tym samym zapobiega występowaniu błędów konwersji, które mogą wystąpić, jeśli wyjście zabezpieczeń odebrało zmienione dane.

Programy obsługi wyjścia wysyłania kanału-zmiana miejsca na dysku
Istnieje możliwość użycia wyjść wysyłania i odbierania w celu transformowania danych przed transmisją. Programy obsługi wyjścia wysyłania kanału mogą dodawać własne dane na temat transformacji przez ponowne udostępnianie miejsca w buforze transmisji.

Dane te są przetwarzane przez program obsługi wyjścia odbierania, a następnie usuwane z buforu. Na przykład można zaszyfrować dane i dodać klucz zabezpieczeń do deszyfrowania.

Jak zarezerwować miejsce i korzystać z niego

Gdy program obsługi wyjścia wysyłania jest wywoływany w celu zainicjowania, ustaw pole *ExitSpace* zmaterializowanej tabeli zapytania (MQXCP) na liczbę bajtów, które mają być zarezerwowane. Szczegółowe informacje na ten temat zawiera sekcja MQCXP . Parametr *ExitSpace* można ustawić tylko podczas inicjowania, to znaczy, gdy wartość *ExitReason* ma wartość MQXR_INIT. Gdy wyjście wysyłania jest wywoływane bezpośrednio przed transmisją, z *ExitReason* ustawionym na MQXR_XMIT, bajty *ExitSpace* są zarezerwowane w buforze transmisji. Produkt *ExitSpace* nie jest obsługiwany w systemie z/OS.

Wyjście wysyłania nie musi używać całej zarezerwowanej przestrzeni. Może używać mniej niż *ExitSpace* bajtów lub, jeśli bufor transmisji nie jest zapełniony, wyjście może wykorzystać więcej niż zarezerwowaną kwotę. Podczas ustawiania wartości parametru *ExitSpace*należy pozostawić co najmniej 1 kB dla danych komunikatu w buforze transmisji. Wydajność kanałów może mieć wpływ, jeśli zarezerwowane miejsce jest używane dla dużych ilości danych.

Co dzieje się na odbierającym końcu kanału

Programy obsługi wyjścia odbierania kanału muszą być skonfigurowane w taki sposób, aby były zgodne z odpowiednimi wyjściami nadawczym. Wyjścia odbierania muszą znać liczbę bajtów w zarezerwowanym obszarze i muszą usunąć te dane w tym obszarze.

Wiele wyjść wysyłania

Istnieje możliwość określenia listy programów obsługi wyjścia wysyłania i odbierania, które mają być uruchamiane w ramach dziedziczenia. Produkt WebSphere MQ obsługuje łączną ilość miejsca zarezerwowanego przez wszystkie wyjścia wysyłania. W tym obszarze musi być co najmniej 1 kB dla danych komunikatu w buforze transmisji.

W poniższym przykładzie pokazano, w jaki sposób przestrzeń jest przydzielana dla trzech wyjść nadawanych, wywołanych kolejno:

1. Po wywołaniu do inicjalizacji:
   - Wysłanie wyjścia A rezerwuje 1 KB.
   - Wysłanie wyjścia B rezerwuje 2 KB.
   - Wysłanie wyjścia C rezerwuje 3 KB.
2. Maksymalna wielkość transmisji wynosi 32 kB, a dane użytkownika o długości 5 kB.
3. Wyjście A jest wywoływane z 5 KB danych; do 27 KB są dostępne, ponieważ 5 KB jest zarezerwowane dla wyjść B i C. Wyjście A dodaje 1 kB, ilość zarezerwowana.
4. Wyjście B jest wywoływane z 6 KB danych; dostępne są do 29 KB, ponieważ 3 KB jest zarezerwowane dla wyjścia C. Wyjście B dodaje 1 KB, mniej niż 2 KB zarezerwowanego.
5. Wyjście C jest wywoływane z 7 KB danych; do 32 kB dostępne są dane. Wyjście C dodaje 10K, więcej niż zarezerwowane 3 KB. Ta kwota jest poprawna, ponieważ łączna ilość danych, 17 kB, jest mniejsza niż maksimum 32 kB.

Programy obsługi wyjścia komunikatów kanału
Wyjścia komunikatów kanału można użyć do wykonywania zadań, takich jak szyfrowanie łącza, sprawdzanie poprawności lub zastępowanie przychodzących identyfikatorów użytkowników, konwersja danych komunikatów, kronikowanie i obsługa komunikatów referencyjnych. Istnieje możliwość określenia listy programów obsługi wyjścia komunikatów, które mają być uruchamiane w ramach dziedziczenia.

Programy obsługi wyjścia komunikatów kanału są wywoływane w następujących miejscach w cyklu przetwarzania agenta MCA:

- Inicjacja i zakończenie MCA
- Natychmiast po wysłaniu przez wysyłającego agenta MCA wywołania MQGET
- Zanim odbierający agent MCA zgłosi wywołanie MQPUT

Wyjście komunikatu jest przekazywane do buforu agenta zawierającego nagłówek kolejki transmisji, MQXQH oraz tekst komunikatu aplikacji pobierany z kolejki. (Format wartości MQXQH jest podany w pliku MQXQH). Jeśli używane są komunikaty odniesienia, czyli komunikaty, które zawierają tylko nagłówek, który wskazuje inny obiekt, który ma zostać wysłany, wyjście komunikatu rozpoznaje nagłówek, MQRMH. Identyfikuje obiekt, pobiera go w dowolny sposób, dodaje go do nagłówka i przekazuje je do agenta MCA w celu przesłania do odbierającego agenta MCA. W odbierającym MCA inny program obsługi wyjścia rozpoznaje, że ten komunikat jest komunikatem odniesienia, wyodrębnia obiekt i przekazuje nagłówek do kolejki docelowej. Więcej informacji na temat komunikatów referencyjnych oraz niektórych przykładowych wyjść komunikatów, które obsługują te komunikaty, można znaleźć w sekcji “Komunikaty odniesienia” na stronie 273 i “Uruchamianie przykładów komunikatów odniesienia” na stronie 145 .

Wyjścia komunikatów mogą zwracać następujące odpowiedzi:

- Wyślij wiadomość (wyjście GET). Komunikat mógł zostać zmieniony przez wyjście. (Ta wartość zwraca wartość MQXCC_OK).
- Umieść komunikat w kolejce (wyjście PUT). Komunikat mógł zostać zmieniony przez wyjście. (Ta wartość zwraca wartość MQXCC_OK).
- Nie przetwarzaj komunikatu. Komunikat jest umieszczany w kolejce niedostarczonych komunikatów (niedostarczona kolejka komunikatów) przez agenta MCA.
- Zamknij kanał.
- Błędny kod powrotu, który powoduje nieprawidłowe zakończenie działania agenta MCA.

Uwaga:

1. Wyjścia komunikatów są wywoływane raz dla każdego przesyłanego kompletnego komunikatu, nawet jeśli komunikat jest podzielony na części.
2. W systemach UNIX , jeśli użytkownik udostępni wyjście komunikatów z jakiegokolwiek powodu, automatyczne przekształcanie identyfikatorów użytkowników na małe litery nie działa. Więcej informacji na ten temat zawiera sekcja Bezpieczeństwo obiektów w systemach UNIX and Linux.
3. Wyjście jest uruchamiane w tym samym wątku co sam agent MCA. Działa on również wewnątrz tej samej jednostki pracy (UOW) co agent MCA, ponieważ korzysta z tego samego uchwytu połączenia. Oznacza to, że wszystkie wywołania wykonane w punkcie synchronizacji są zatwierdzane lub wycofane przez kanał na końcu zadania wsadowego. Na przykład jeden program obsługi wyjścia komunikatów kanału może wysyłać komunikaty powiadomień do innego, a te komunikaty są zatwierdzane tylko w kolejce, gdy zadanie wsadowe zawierające oryginalny komunikat zostanie zatwierdzone.

   Z tego powodu możliwe jest wystawianie wywołań MQI punktów synchronizacji z programu obsługi wyjścia komunikatów kanału.

Konwersja komunikatów poza wyjściem komunikatu
Przed wywołaniem wyjścia komunikatu odbierający agent MCA wykonuje pewne konwersje w komunikacie. W tej sekcji opisano algorytmy używane do przeprowadzania konwersji.

Które nagłówki są przetwarzane

Procedura konwersji jest uruchamiana w MCA odbiornika przed wywołaniem wyjścia komunikatu. Procedura konwersji rozpoczyna się od nagłówka MQXQH na początku komunikatu. Następnie procedura konwersji przetwarza przez połączone nagłówki, które są zgodne z MQXQH, w razie potrzeby przeprowadzając konwersję. Nagłówki łańcuchowe mogą wykraczać poza przesunięcie zawarte w parametrze HeaderLength danych MQCXP przekazywanych do wyjścia komunikatu odbiornika. Następujące nagłówki są przekształcane w miejsce:

- MQXQH (nazwa formatu "`MQXMIT `")
- MQMD (ten nagłówek jest częścią tabeli MQXQH i nie ma nazwy formatu)
- MQMDE (nazwa formatu "`MQHMDE `")
- MQDH (nazwa formatu "`MQHDIST `")

- MQWIH (nazwa formatu "MQHWIH ")

Następujące nagłówki nie są przekształcane, ale są one stopniowane, ponieważ agent MCA kontynuuje przetwarzanie połączonych nagłówków:

- MQDLH (nazwa formatu "MQDEAD ")
- wszystkie nagłówki o nazwach rozpoczynających się od trzech znaków 'MQH' (na przykład "MQHRF ") które nie są wymienione w inny sposób

Sposób przetwarzania nagłówków

Parametr Format każdego nagłówka WebSphere MQ jest odczytyany przez agenta MCA. Parametr Format to 8 bajtów w nagłówku, które są 8 jednobajtowymi znakami zawierającymi nazwę.

Następnie agent MCA interpretuje dane po każdym nagłówku jako typ nazwanego typu. Jeśli format jest nazwą typu nagłówka kwalifikującego się do konwersji danych produktu WebSphere MQ , jest on przekształcany. Jeśli jest to inna nazwa wskazująca dane inne niżMQ (na przykład MQFMT_NONE lub MQFMT_STRING), agent MCA zatrzyma przetwarzanie nagłówków.

Co to jest HeaderLengthMQCXP?

Parametr HeaderLength w danych MQCXP dostarczanych do wyjścia komunikatów to łączna długość nagłówków MQXQH (w tym MQMD), MQMDE i MQDH na początku komunikatu. Nagłówki te są łańcuchowe przy użyciu nazw i długości formatu 'Format'.

MQWIH

Nagłówki łańcuchowe mogą wykraczać poza obszar HeaderLength do obszaru danych użytkownika. Nagłówek MQWIH, jeśli jest obecny, jest jednym z tych nagłówków, które pojawiają się poza HeaderLength.

Jeśli istnieje nagłówek MQWIH w nagłówkach łańcuchowych, jest on przekształcany w miejscu przed wywołaniem wyjścia komunikatu odbiorcy.

Program obsługi wyjścia dla ponowienia komunikatu kanału
Wywołanie wyjścia komunikatu kanału jest wywoływane w przypadku, gdy próba otwarcia kolejki docelowej nie powiodła się. Można użyć wyjścia, aby określić, w jakich okolicznościach należy ponowić próbę, ile razy należy ponowić próbę, a także jak często.

To wyjście jest również wywoływane na odbierającym końcu kanału w inicjacji MCA i zakończeniu.

Wyjście komunikatu kanału-wyjście ponowienia jest przekazywane do buforu agenta zawierającego nagłówek kolejki transmisji, MQXQH oraz tekst komunikatu aplikacji pobierany z kolejki. Format wartości MQXQH jest podany w sekcji Przegląd dla MQXQH.

Wyjście jest wywoływane dla wszystkich kodów przyczyny; wyjście określa, dla których kodów przyczyny ma być ponawiane przez agenta MCA, przez ile razy i w jakich odstępach czasu. (Wartość licznika ponowień komunikatu, gdy kanał został zdefiniowany, jest przekazywany do wyjścia na zmaterializowanej tabeli MQCD, ale wyjście może zignorować tę wartość).

Pole Licznik MsgRetryw produkcie MQCXP jest zwiększane przez agenta MCA przy każdym wywołaniu wyjścia, a wyjście zwraca wartość MQXCC_OK z czasem oczekiwania zawartym w polu MsgRetryw polu Odstęp czasu MQCXP lub MQXCC_SUPPRESS_FUNCTION. Ponowne próby są kontynuowane w nieskończoność do momentu, aż wyjście zwróci MQXCC_SUPPRESS_FUNCTION w polu ExitResponse w MQCXP. Informacje na temat działań podjętych przez agenta MCA dla tych kodów zakończenia zawiera sekcja MQCXP .

Jeśli wszystkie ponowienia nie powiodą się, komunikat zostanie zapisany w kolejce niedostarczonych komunikatów. Jeśli nie jest dostępna żadna kolejka niedostarczonych komunikatów, kanał zostanie zatrzymany.

Jeśli dla kanału nie zostanie zdefiniowane wyjście dla ponowienia komunikatu, a wystąpi błąd, który prawdopodobnie będzie tymczasowy, na przykład MQRC_Q_FULL, agent MCA użyje liczby ponowień komunikatu i przedziały czasu ponowienia komunikatu, gdy kanał został zdefiniowany. Jeśli błąd ma charakter bardziej trwały, a użytkownik nie zdefiniował programu obsługi wyjścia do obsługi tego błędu, komunikat jest zapisywany w kolejce niedostarczonych komunikatów.

Program obsługi wyjścia automatycznej definicji kanału
Wyjście automatyczne definicji kanału może być używane, gdy odebrano żądanie uruchomienia kanału odbiorczego lub kanału połączenia serwera, ale nie istnieje definicja dla tego kanału (nie dotyczy produktu WebSphere MQ for z/OS). Można ją również wywołać na wszystkich platformach dla kanałów wysyłających klastry i kanały odbierające klastry w celu umożliwienia modyfikacji definicji dla instancji kanału.

Wyjście automatyczne definicji kanału może być wywoływane na wszystkich platformach z wyjątkiem systemu z/OS , gdy odebrano żądanie uruchomienia kanału odbiorczego lub kanału połączenia z serwerem, ale nie istnieje definicja kanału. Można go użyć do zmodyfikowania podanej definicji domyślnej dla automatycznie zdefiniowanego kanału odbiorczego lub kanału połączenia z serwerem, SYSTEM.AUTO.RECEIVERlub SYSTEM.AUTO.SVRCON. Informacje na temat automatycznego tworzenia definicji kanałów można znaleźć w sekcji Przygotowywanie kanałów .

Wyjście automatyczne definicji kanału może być również wywoływane po odebraniu żądania w celu uruchomienia kanału nadawczego klastra. Można go wywołać dla kanałów wysyłających klastry i kanały odbierające klastry w celu umożliwienia modyfikacji definicji dla tej instancji kanału. W takim przypadku wyjście ma również zastosowanie do produktu WebSphere MQ for z/OS. Częstym zastosowaniem wyjścia z automatycznego definiowania kanału jest zmiana nazw wyjść komunikatów (MSGEXIT, RCVEXIT, SCYEXIT i SENDEXIT), ponieważ nazwy wyjść mają różne formaty na różnych platformach. Jeśli nie określono wyjścia automatycznego definiowania kanału, domyślnym zachowaniem w systemie z/OS jest sprawdzenie rozproszonej nazwy wyjścia w postaci *[path]/libraryname(function)* i podjęcie maksymalnie ośmiu znaków funkcji, jeśli istnieje, lub nazwy biblioteki. W systemie z/OSprogram obsługi wyjścia automatycznej definicji kanału musi zmieniać pola adresowane przez MsgExitPtr, MsgUserDataPtr, SendExitPtr, SendUserDataPtr, ReceiveExitPtr i ReceiveUserDataPtr, a nie MsgExit, MsgUserData, SendExit, Same pola danych SendUserData, ReceiveExit i ReceiveUser.

Więcej informacji na ten temat zawiera sekcja Auto-definition of channels(Automatyczne definiowanie kanałów).

Podobnie jak w przypadku innych wyjść kanału, lista parametrów jest następująca:

```
MQ_CHANNEL_AUTO_DEF_EXIT (ChannelExitParms, ChannelDefinition)
```

Produkt `ChannelExitParms` jest opisany w sekcji MQCXP. Produkt `ChannelDefinition` jest opisany w sekcji MQCD.

MQCD zawiera wartości, które są używane w domyślnej definicji kanału, jeśli nie zostały zmienione przez wyjście. Wyjście może modyfikować tylko podzbiór pól; patrz MQ_CHANNEL_AUTO_DEF_EXIT. Jednak próba zmiany innych pól nie powoduje błędu.

Wyjście automatycznego definiowania kanału zwraca odpowiedź albo MQXCC_OK, albo MQXCC_SUPPRESS_FUNCTION. Jeśli żadna z tych odpowiedzi nie zostanie zwrócona, agent MCA będzie kontynuować przetwarzanie, tak jak gdyby zwrócono MQXCC_SUPPRESS_FUNCTION. Oznacza to, że automatyczna definicja jest porzucona, nie jest tworzona nowa definicja kanału i nie można uruchomić kanału.

Kompilowanie programów obsługi wyjścia kanału w systemach Windows, UNIX and Linux

Poniższe przykłady ułatwiają kompilowanie programów obsługi wyjścia kanału w systemach Windows i UNIX and Linux .

Windows

Komenda kompilatora i konsolidatora dla programów obsługi wyjścia kanału w systemie Windows:

```
cl.exe /Ic:\mqm\tools\c\include /nologo /c myexit.c
link.exe /nologo /dll myexit.obj /def:myexit.def /out:myexit.dll
```

Systemy UNIX i Linux

W tych przykładach `exit` jest nazwą biblioteki, a `ChannelExit` jest nazwą funkcji. W systemie AIX plik eksportu nosi nazwę `exit.exp`. Te nazwy są używane przez definicję kanału do odwołania się do programu obsługi wyjścia przy użyciu formatu opisanego w sekcji Definicja kanału MQCD. Zapoznaj się także z parametrem MSGEXIT komendy DEFINE CHANNEL .

Przykładowe komendy kompilatora i konsolidatora dla wyjść kanału w systemie AIX:

```
$ xlc_r -q64 -e MQStart -bE:exit.exp -bM:SRE -o /var/mqm/exits64/exit
exit.c -I/usr/mqm/inc
```

Przykładowe komendy kompilatora i konsolidatora dla wyjść kanału w systemie HP-UX

```
$ c89 +DD64 +z -c -D_HPUX_SOURCE -o exit.o exit.c -I/opt/mqm/inc
$ ld -b exit.o +ee MQStart +ee ChannelExit -o
/var/mqm/exits64/exit -L/usr/lib/pa20_64 -lpthread
$ rm exit.o
```

Przykładowe komendy kompilatora i konsolidatora dla wyjść kanału na platformach Linux , na których menedżer kolejek jest 32-bitowy:

```
$ gcc -shared -fPIC -o /var/mqm/exits/exit exit.c -I/opt/mqm/inc
```

Przykładowe komendy kompilatora i konsolidatora dla wyjść kanału na platformach Linux , na których menedżer kolejek jest 64-bitowy:

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
```

Przykładowe komendy kompilatora i konsolidatora dla wyjść kanału w systemie Solaris:

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
-R/usr/lib/64 -lsocket -lnsl -ldl
```

Na kliencie można użyć wyjścia 32-bitowego lub 64-bitowego. To wyjście musi być połączone z mqic_r.

W systemie AIXwszystkie funkcje wywoływane przez program IBM WebSphere MQ muszą zostać wyeksportowane. Przykładowy plik eksportu dla tego pliku make:

```
#!
channelExit
MQStart
```

Konfigurowanie wyjść kanału

Aby wywołać wyjście kanału, należy podać nazwę w definicji kanału.

Wyjścia kanału muszą być nazwane w definicji kanału. To nazewnictwo można wykonać po pierwszym zdefiniowaniu kanałów lub później można dodać informacje później, na przykład za pomocą komendy MQSC ALTER CHANNEL. Nazwy wyjść kanału można również nadać w strukturze danych kanału MQCD. Format nazwy wyjścia zależy od używanej platformy IBM WebSphere MQ . Informacje na ten temat można znaleźć w sekcji Komendy MQCD lub Script (MQSC) Commands .

Jeśli definicja kanału nie zawiera nazwy programu użytkownika obsługi wyjścia, program zewnętrzny nie jest wywoływany.

Wyjście automatyczne definicji kanału to właściwość menedżera kolejek, a nie pojedynczego kanału. Aby to wyjście zostało wywołane, musi ono być nazwane w definicji menedżera kolejek. Aby zmienić definicję menedżera kolejek, należy użyć komendy MQSC ALTER QMGR.

Pisanie wyjść konwersji danych

Ta kolekcja tematów zawiera informacje na temat sposobu pisania wyjść konwersji danych.

Uwaga: Nieobsługiwane w programie MQSeries dla VSE/ESA.

Podczas wykonania operacji MQPUT aplikacja tworzy deskryptor komunikatu (MQMD) komunikatu. Ponieważ produkt WebSphere MQ musi być w stanie zrozumieć zawartość deskryptora MQMD niezależnie od platformy, na której jest ona tworzona, jest ona automatycznie przekształcana przez system.

Dane aplikacji nie są jednak przekształcane automatycznie. Jeśli dane znakowe są wymieniane między platformami, w których pola *CodedCharSetId* i *Encoding* różnią się między innymi, na przykład między ASCII a EBCDIC, aplikacja musi zorganizować konwersję tego komunikatu. Konwersja danych aplikacji może być wykonywana przez sam menedżer kolejek lub przez program obsługi wyjścia użytkownika, zwany *wyjściem konwersji danych*. Jeśli dane aplikacji znajdują się w jednym z wbudowanych formatów (takich jak MQFMT_STRING), menedżer kolejek może samodzielnie wykonać konwersję danych, korzystając z jednej z wbudowanych procedur konwersji. Ten temat zawiera informacje na temat narzędzia wyjścia konwersji danych WebSphere MQ w przypadku, gdy dane aplikacji nie są w formacie wbudowanym.

Sterowanie może być przekazywane do wyjścia konwersji danych podczas wywołania MQGET. Pozwala to na uniknięcie konwersji na różne platformy przed dotarciem do miejsca docelowego. Jeśli jednak ostatnim miejscem docelowym jest platforma, która nie obsługuje konwersji danych w tabeli MQGET, należy określić CONVERT (YES) w kanale nadawczym, który wysyła dane do jego miejsca docelowego. Dzięki temu produkt WebSphere MQ przekształca dane w czasie transmisji. W takim przypadku wyjście konwersji danych musi znajdować się w systemie, w którym zdefiniowany jest kanał nadawczy.

Wywołanie MQGET jest wysyłane bezpośrednio przez aplikację. Ustaw wartości pól *CodedCharSetId* i *Encoding* w strukturze MQMD na wymagany zestaw znaków i kodowanie. Jeśli aplikacja używa tego samego zestawu znaków i kodowania co menedżer kolejek, należy ustawić wartość *CodedCharSetId* na wartość MQCCSI_Q_MGR, a wartość *Encoding* na MQENC_NATIVE. Po zakończeniu wywołania MQGET pola te mają wartości odpowiednie dla zwracanych danych komunikatu. Wartości te mogą się różnić od wartości wymaganych w przypadku, gdy konwersja nie powiodła się. Aplikacja powinna zresetować te pola do wartości wymaganych przed każdym wywołaniem MQGET.

Warunki wymagane dla wyjścia konwersji danych, które mają zostać wywołane, są definiowane dla wywołania MQGET w MQGET.

Opis parametrów, które są przekazywane do wyjścia konwersji danych oraz szczegółowe informacje o używaniu, zawiera sekcja Konwersja danych dla wywołania MQ_DATA_CONV_EXIT i struktury MQDXP.

Programy przekształcające dane aplikacji między różnymi kodowaniami maszyn i identyfikatorami CCSID muszą być zgodne z interfejsem DCI (WebSphere MQ data conversion interface).

Dzięki wprowadzeniu klientów Multicast wyjścia funkcji API i wyjścia konwersji danych muszą być możliwe do uruchomienia po stronie klienta, ponieważ niektóre komunikaty mogą nie przechodzić przez menedżer kolejek. Następujące biblioteki są teraz częścią pakietów klienta, a także pakietów serwera:

Tabela 56. Biblioteki, które znajdują się teraz w pakietach klienta i serwera

System operacyjny	**Biblioteki**
Windows	32 bity & 64 bit: mqm.dll & mqm.pdb
Linux & HP-UX	32 bity & 64 bit: libmqm.so & libmqm_r.so
AIX	32 bity & 64 bit: libmqm.a & libmqm_r.a
Solaris	32 bity & 64 bit: libmqm.so

Wywoływanie wyjścia konwersji danych

Wyjście konwersji danych to wyjście napisane przez użytkownika, które odbiera sterowanie podczas przetwarzania wywołania MQGET.

Wyjście jest wywoływane, jeśli spełnione są następujące warunki:

- Opcja MQGMO_CONVERT została określona w wywołaniu MQGET.
- Niektóre lub wszystkie dane komunikatu nie znajdują się w żądanym zestawie znaków ani kodowaniu.
- Pole *Format* w strukturze MQMD powiązanej z komunikatem nie ma wartości MQFMT_NONE.
- Wartość *BufferLength* podana w wywołaniu MQGET nie jest równa zero.
- Długość danych komunikatu nie jest równa zero.
- Komunikat zawiera dane, które mają format zdefiniowany przez użytkownika. Format zdefiniowany przez użytkownika może zajmować cały komunikat lub być poprzedzony jednym lub większą liczbę wbudowanych formatów. Na przykład: format zdefiniowany przez użytkownika może być poprzedzony formatem MQFMT_DEAD_LETTER_HEADER. Wyjście jest wywoływane w celu przekształcenia tylko formatu zdefiniowanego przez użytkownika. Menedżer kolejek przekształca wszystkie wbudowane formaty, które poprzedzają format zdefiniowany przez użytkownika.

  Program obsługi wyjścia napisany przez użytkownika można również wywołać w celu przekształcenia wbudowanego formatu, ale dzieje się tak tylko wtedy, gdy wbudowane procedury konwersji nie mogą pomyślnie przekształcić wbudowanego formatu.

Istnieją inne warunki, które zostały w pełni opisane w uwagach dotyczących użycia wywołania MQ_DATA_CONV_EXIT w MQ_DATA_CONV_EXIT.

Szczegółowe informacje na temat wywołania MQGET zawiera sekcja MQGET . Wyjścia konwersji danych nie mogą używać wywołań MQI, innych niż wywołania MQXCNVC.

Nowa kopia wyjścia jest ładowana, gdy aplikacja próbuje pobrać pierwszy komunikat, który używa tego produktu *Format* od momentu połączenia aplikacji z menedżerem kolejek. Nowa kopia może być również ładowana w innych sytuacjach, jeśli menedżer kolejek odrzuciło wcześniej załadowaną kopię.

Wyjście konwersji danych jest uruchamiane w środowisku takim jak program, który wywołał wywołanie MQGET. Program może także być agentem MCA (agenta kanału komunikatów) wysyłającym komunikaty do docelowego menedżera kolejek, który nie obsługuje konwersji komunikatów. Środowisko obejmuje przestrzeń adresową i profil użytkownika, tam gdzie ma to zastosowanie. Wyjście nie może spowodować naruszenia integralności menedżera kolejek, ponieważ nie jest ono uruchamiane w środowisku menedżera kolejek.

Zapisywanie wyjścia konwersji danych dla produktu WebSphere MQ w systemach UNIX and Linux

Informacje na temat kroków, które należy wziąć pod uwagę podczas pisania programów obsługi wyjścia konwersji danych dla produktu WebSphere MQ w systemach UNIX and Linux .

Wykonaj następujące kroki:

1. Podaj nazwę formatu wiadomości. Nazwa musi mieścić się w polu *Format* deskryptora MQMD i musi być podana wielkimi literami, na przykład MYFORMAT. Nazwa *Format* nie może zawierać odstępów początkowych. Odstępy końcowe są ignorowane. Nazwa obiektu nie może zawierać więcej niż osiem znaków, ponieważ *Format* ma tylko osiem znaków długości. Pamiętaj, aby używać tej nazwy za każdym razem, gdy wysyłany jest komunikat.

   Jeśli wyjście konwersji danych jest używane w środowisku wielowątkowym, obiekt ładujący musi być poprzedzony przez _r, aby wskazać, że jest to wersja wielowątkowa.
2. Utwórz strukturę, która będzie reprezentowana przez komunikat. Przykład można znaleźć w sekcji poprawna składnia .
3. Tę strukturę należy uruchomić za pomocą komendy `crtmqcvx` , aby utworzyć fragment kodu dla wyjścia konwersji danych.

   Funkcje generowane przez komendę `crtmqcvx` używają makr, które zakładają, że wszystkie struktury są spakowane; należy je zmienić, jeśli nie jest to przypadek.
4. Skopiuj dostarczony plik źródłowy szkieletu, zmieniając jego nazwę na nazwę formatu komunikatu ustawionego w kroku “1” na stronie 427. Plik źródłowy szkieletu, a także kopia, są tylko do odczytu.

Plik źródłowy szkieletu nosi nazwę amqsvfc0.c.

5. W produkcie WebSphere MQ for AIXdostarczany jest również plik eksportu szkieletu o nazwie amqsvfc.exp . Skopiuj ten plik, zmieniając jego nazwę na MYFORMAT.EXP.
6. Szkielet zawiera przykładowy plik nagłówkowy amqsvmha.hw katalogu *MQ_INSTALLATION_PATH*/inc, gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ . Upewnij się, że ścieżka zawiera punkty ścieżki do tego katalogu, aby pobrać ten plik.

   Plik amqsvmha.h zawiera makra, które są używane przez kod wygenerowany przez komendę `crtmqcvx` . Jeśli struktura, która ma zostać przekształcona, zawiera dane znakowe, te makra wywołują MQXCNVC.
7. Znajdź następujące pola komentarza w pliku źródłowym i wstaw kod zgodnie z opisem:
   a. W celu zakończenia pliku źródłowego, pole komentarza rozpoczyna się od:

   ```
   /* Insert the functions produced by the data-conversion exit */
   ```

   W tym miejscu wstaw fragment kodu wygenerowany w kroku “3” na stronie 427.

   b. W pobliżu środka pliku źródłowego, pole komentarza rozpoczyna się od:

   ```
   /* Insert calls to the code fragments to convert the format's */
   ```

   Następuje to za pomocą skomentowanego wywołania funkcji `Converttag`STRUCT.

   Zmień nazwę funkcji na nazwę funkcji, która została dodana w kroku “7.a” na stronie 428. Usuń znaki komentarza, aby aktywować funkcję. Jeśli istnieje kilka funkcji, utwórz połączenia dla każdego z nich.

   c. W pobliżu początku pliku źródłowego, pole komentarza rozpoczyna się od:

   ```
   /* Insert the function prototypes for the functions produced by */
   ```

   W tym miejscu należy wstawić instrukcje prototypu funkcji dla funkcji dodanych w kroku “3” na stronie 427 powyżej.
8. Skompiluj wyjście jako bibliotekę współużytkowaną, używając komendy MQStart jako punktu wejścia. Aby to zrobić, należy zapoznać się z “Kompilowanie wyjść konwersji danych w systemach UNIX and Linux” na stronie 428.
9. Umieść dane wyjściowe w katalogu wyjściowym. Domyślnym katalogiem wyjścia jest `/var/mqm/exits` dla 32-bitowych systemów i `/var/mqm/exits64`dla systemów 64-bitowych. Te katalogi można zmienić w pliku qm.ini lub mqclient.ini . Ta ścieżka może być ustawiona dla każdego menedżera kolejek, a wyjście jest wyszukiwawcze tylko w tej ścieżce lub ścieżkach.

Uwaga:

1. Jeśli produkt `crtmqcvx` używa spakowanych struktur, wszystkie aplikacje produktu WebSphere MQ muszą być kompilowane w ten sposób.
2. Programy obsługi wyjścia konwersji danych muszą być ponownie wejściowane.
3. MQXCNVC jest to *tylko* wywołanie MQI, które może zostać wysłane z wyjścia konwersji danych.

Kompilowanie wyjść konwersji danych w systemach UNIX and Linux
Przykłady kompilacji wyjścia konwersji danych w systemach UNIX and Linux .

Na wszystkich platformach punktem wejścia do modułu jest MQStart.

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

AIX

Skompiluj kod źródłowy wyjścia, wydając jedną z następujących komend:

Aplikacje 32-bitowe

Niewątkowa

```
cc -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Wątkowy

```
xlc_r -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT_r \
     MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Aplikacje 64-bitowe

Niewątkowa

```
cc -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Wątkowy

```
xlc_r -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT_r \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Platforma HP-UX Itanium

Skompiluj i dowiąż kod źródłowy wyjścia, wydając jeden z następujących zestawów komend:

Aplikacje 32-bitowe

Niewątkowa

Skompiluj kod źródłowy wyjścia:

```
c89 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Połącz obiekt wyjścia:

```
ld +b: -b MYFORMAT.o +ee MQStart -o  \
    /var/mqm/exits/MYFORMAT -L/usr/lib/hpux32
rm MYFORMAT.o
```

Wątkowy

Skompiluj kod źródłowy wyjścia:

```
c89 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Połącz obiekt wyjścia:

```
ld +b: -b MYFORMAT.o +ee MQStart -o  \
    /var/mqm/exits/MYFORMAT_r -L/usr/lib/hpux32 \
    -lpthread
rm MYFORMAT.o
```

Aplikacje 64-bitowe

Niewątkowa

Skompiluj kod źródłowy wyjścia:

```
 c89 +DD64 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Połącz obiekt wyjścia:

```
ld -b MYFORMAT.o +ee MQStart \
    -o  /var/mqm/exits64/MYFORMAT \
    -L/usr/lib/hpux64
rm MYFORMAT.o
```

Wątkowy

Skompiluj kod źródłowy wyjścia:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Połącz obiekt wyjścia:

```
ld -b MYFORMAT.o +ee MQStart  \
    -o  /var/mqm/exits64/MYFORMAT_r  \
    -L/usr/lib/hpux64 -lpthread
rm MYFORMAT.o
```

Linux

Skompiluj kod źródłowy wyjścia, wydając jedną z następujących komend:

31-bitowe aplikacje

Niewątkowa

```
 gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c \
      -IMQ_INSTALLATION_PATH/inc
```

Wątkowy

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

Aplikacje 32-bitowe

Niewątkowa

```
 gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c
      -IMQ_INSTALLATION_PATH/inc
```

Wątkowy

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

Aplikacje 64-bitowe

Niewątkowa

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT MYFORMAT.c
      -IMQ_INSTALLATION_PATH/inc
```

Wątkowy

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT_r MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

Solaris

Skompiluj kod źródłowy wyjścia, wydając jedną z następujących komend:

Aplikacje 32-bitowe

platforma SPARC

```
cc -xarch=v8plus -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

platformax86-64

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

Aplikacje 64-bitowe

platforma SPARC

```
cc -xarch=v9 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/64 -lsocket -lnsl -ldl
```

platformax86-64

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/64  -lsocket -lnsl -ldl
```

Zapisywanie wyjścia konwersji danych dla produktu WebSphere MQ for Windows

Informacje na temat kroków, które należy wziąć pod uwagę podczas zapisywania programów obsługi wyjścia konwersji danych dla produktu WebSphere MQ for Windows.

Wykonaj następujące kroki:

1. Podaj nazwę formatu wiadomości. Nazwa musi mieścić się w polu *Format* deskryptora MQMD. Nazwa *Format* nie może zawierać odstępów początkowych. Odstępy końcowe są ignorowane. Nazwa obiektu nie może zawierać więcej niż osiem znaków, ponieważ *Format* ma tylko osiem znaków długości.

   Plik .DEF o nazwie amqsvfcn.def jest również dostarczany w katalogu przykładów, *MQ_INSTALLATION_PATH*\Tools\C\Samples. *MQ_INSTALLATION_PATH* Jest to katalog, w którym zainstalowano produkt WebSphere MQ . Należy wykonać kopię tego pliku i zmienić jej nazwę na przykład na MYFORMAT.DEF. Upewnij się, że nazwa biblioteki DLL jest tworzona i ma nazwę określoną w polu MYFORMAT.DEF jest taka sama. Zastąp nazwę FORMAT1 w polu MYFORMAT.DEF z nową nazwą formatu.

   Pamiętaj, aby używać tej nazwy za każdym razem, gdy wysyłany jest komunikat.
2. Utwórz strukturę, która będzie reprezentowana przez komunikat. Przykład można znaleźć w sekcji poprawna składnia .
3. Tę strukturę należy uruchomić za pomocą komendy `crtmqcvx` , aby utworzyć fragment kodu dla wyjścia konwersji danych.

   Funkcje generowane przez komendę CRTMQCVX używają makr, które są napisane przy założeniu, że wszystkie struktury są spakowane; należy je zmienić, jeśli nie jest to przypadek.
4. Skopiuj dostarczony plik źródłowy szkieletu amqsvfc0.c, zmieniając nazwę na nazwę formatu komunikatu ustawionego w kroku “1” na stronie 431.

Plik amqsvfc0.c znajduje się w katalogu *MQ_INSTALLATION_PATH*\Tools\C\Samples , gdzie *MQ_INSTALLATION_PATH* jest katalogiem, w którym zainstalowano produkt WebSphere MQ . Domyślny katalog instalacyjny to C:\Program Files\IBM\WebSphere MQ.

Szkielet zawiera przykładowy plik nagłówkowy amqsvmha.h w katalogu *MQ_INSTALLATION_PATH*\Tools\C\include . Upewnij się, że ścieżka zawiera punkty ścieżki do tego katalogu, aby pobrać ten plik.

Plik amqsvmha.h zawiera makra, które są używane przez kod wygenerowany przez komendę CRTMQCVX. Jeśli struktura, która ma zostać przekształcona, zawiera dane znakowe, te makra wywołują MQXCNVC.

5. Znajdź następujące pola komentarza w pliku źródłowym i wstaw kod zgodnie z opisem:
   a. W celu zakończenia pliku źródłowego, pole komentarza rozpoczyna się od:

   ```
   /* Insert the functions produced by the data-conversion exit */
   ```

   W tym miejscu wstaw fragment kodu wygenerowany w kroku “3” na stronie 431.
   b. W pobliżu środka pliku źródłowego, pole komentarza rozpoczyna się od:

   ```
   /* Insert calls to the code fragments to convert the format's */
   ```

   Następuje to za pomocą skomentowanego wywołania funkcji ConverttagSTRUCT.

   Zmień nazwę funkcji na nazwę funkcji, która została dodana w kroku “5.a” na stronie 432. Usuń znaki komentarza, aby aktywować funkcję. Jeśli istnieje kilka funkcji, utwórz połączenia dla każdego z nich.
   c. W pobliżu początku pliku źródłowego, pole komentarza rozpoczyna się od:

   ```
   /* Insert the function prototypes for the functions produced by */
   ```

   W tym miejscu należy wstawić instrukcje prototypu funkcji dla funkcji dodanych w kroku “3” na stronie 431.
6. Utwórz następujący plik komend:

   ```
   cl -I MQ_INSTALLATION_PATH\Tools\C\Include -Tp   \
   MYFORMAT.C

   MYFORMAT.DEF
   ```

   gdzie *MQ_INSTALLATION_PATH* to katalog, w którym zainstalowany jest produkt WebSphere MQ .
7. Uruchom plik komend, aby skompilować program obsługi wyjścia jako plik DLL.
8. Umieść dane wyjściowe w podkatalogu wyjścia poniżej katalogu danych produktu WebSphere MQ . Domyślnym katalogiem na potrzeby instalowania wyjść w systemach 32-bitowych jest *MQ_DATA_PATH*\Exits , a dla systemów 64-bitowych jest to *MQ_DATA_PATH*\Exits64

   Ścieżka używana do wyszukiwania wyjść konwersji danych jest podana w rejestrze. Folder rejestru jest następujący:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\ClientExitPa
   th\
   ```

   a kluczem rejestru jest: ExitsDefaultPath. Ta ścieżka może być ustawiona dla każdego menedżera kolejek, a wyjście jest wyszukiwawcze tylko w tej ścieżce lub ścieżkach.

Uwaga:

1. Jeśli komenda CRTMQCVX korzysta ze spakowanych struktur, wszystkie aplikacje produktu WebSphere MQ muszą być kompilowane w ten sposób.

2. Programy obsługi wyjścia konwersji danych muszą być ponownie wejściowane.
3. MQXCNVC jest to *tylko* wywołanie MQI, które może zostać wysłane z wyjścia konwersji danych.

Wyjdź i przełączaj pliki ładowania w systemach operacyjnych Windows

Procesy menedżera kolejek produktu IBM WebSphere MQ for Windows Version 7.5 są 32-bitowe. W rezultacie w przypadku używania aplikacji 64-bitowych niektóre typy plików ładowania wyjścia i ładowania z przełącznikami XA również muszą mieć wersję 32-bitową dostępną do użycia przez menedżer kolejek. Jeśli wymagana jest 32-bitowa wersja wyjścia lub plik ładowania przełącznika XA, a nie jest on dostępny, to wywołanie lub wykonanie odpowiedniej komendy API nie powiedzie się.

W polu `qm.ini file` dla parametru *ExitPath*obsługiwane są dwa atrybuty. Są to produkty `ExitsDefaultPath=`*`MQ_INSTALLATION_PATH`*`\exits` i `ExitsDefaultPath64=`*`MQ_INSTALLATION_PATH`*`\exits64`. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ . Dzięki temu możliwe będzie odnalezcie odpowiedniej biblioteki. Jeśli wyjście jest używane w klastrze WebSphere MQ , to zapewnia również, że można znaleźć odpowiednią bibliotekę w systemie zdalnym.

Poniższa tabela zawiera listę różnych typów plików ładowania wyjścia i przełączników oraz informacje o tym, czy są wymagane 32-bitowe, czy 64-bitowe wersje, czy są używane aplikacje 32-lub 64-bitowe:

Typy plików	**Aplikacje 32-bitowe**	**Aplikacje 64-bitowe**
zewnętrzny program obsługi wywołań API	32 bity	32-i 64-bitowy
Wyjście konwersji danych	32 bity	64-bitowe
Wyjścia kanału serwera (wszystkie typy)	32 bity	32 bity
Wyjścia kanału klienta (wszystkie typy)	32 bity	64-bitowe
Wyjście usługi instalowalnej	32 bity	32 bity
Moduł śledzenia usługi	32 bity	32-i 64-bitowy
Wyjście WLM klastra	32 bity	32 bity
Wyjście routingu publikowania/ subskrypcji	32 bity	32 bity
Pliki ładowania przełączników bazy danych	32 bity	32-i 64-bitowy
Zewnętrzne biblioteki AX menedżera transakcji zewnętrznych	32 bity	64-bitowe

Odwołanie do definicji połączenia przy użyciu wyjścia wstępnego z połączeniem z repozytorium

Klienty MQI produktu WebSphere MQ MQI można skonfigurować w celu wyszukiwania repozytorium w celu uzyskania definicji połączeń przy użyciu biblioteki wyjścia wstępnego połączenia.

Wprowadzenie

Aplikacja kliencka może łączyć się z menedżerem kolejek przy użyciu tabel definicji kanału klienta (CCDT). Zwykle plik CCDT znajduje się na centralnym serwerze plików sieciowych i ma do niego odniesienie klientów. Ponieważ zarządzanie różnymi aplikacjami klienckim odwołujących się do pliku CCDT jest trudne, elastycznym podejściem jest przechowywanie definicji klientów w repozytorium globalnym, takich jak katalog LDAP, rejestr produktu WebSphere i repozytorium lub inne repozytorium.

Przechowywanie definicji połączeń klienta w repozytorium ułatwia zarządzanie definicjami połączenia klienckiego, a aplikacje mogą uzyskiwać dostęp do poprawnych i najbardziej aktualnych definicji połączeń klientów.

Podczas wykonywania wywołania MQCONN/X program IBM WebSphere MQ MQI client ładuje określoną bibliotekę wyjścia do połączenia z aplikacją, a następnie wywołuje funkcję wyjścia w celu pobrania definicji połączeń. Pobrane definicje połączeń są następnie używane do nawiązania połączenia z menedżerem kolejek. Szczegóły dotyczące biblioteki obsługi wyjścia i funkcji do wywołania są określone w pliku konfiguracyjnym mqclient.ini .

Składnia

`void MQ_PRECONNECT_EXIT` (parametry*pExitParms*, *pQMgrName*, *ppConnectOpts*, *pCompCode*, *pReason*);

Parametry

pExitParms

Typ: PMQNXP wejście/wyjście

Struktura parametru wyjścia **`PreConnection`** .

Struktura jest przydzielana i obsługiwana przez program wywołujący wyjście.

NazwapQMgr

Typ: input/output PMQCHAR

Nazwa menedżera kolejek.

Na wejściu parametr ten jest łańcuchem filtru dostarczonym do wywołania MQCONN API za pomocą parametru **`QMgrName`** . To pole może być puste, jawne lub zawierać określone znaki wieloznaczne. Pole zostanie zmienione przez wyjście. Parametr ma wartość NULL, gdy wyjście jest wywoływane z MQXR_TERM.

ppConnectOpts (Opts)

Wpisz: ppConnectOpts input/output

Opcje, które sterują działaniem MQCONNX.

Jest to wskaźnik do struktury opcji połączenia MQCNO, która steruje działaniem wywołania interfejsu API MQCONN. Parametr ma wartość NULL, gdy wyjście jest wywoływane z MQXR_TERM. Klient MQI zawsze udostępnia strukturę MQCNO do wyjścia, nawet jeśli nie została ona pierwotnie udostępniona przez aplikację. Jeśli aplikacja udostępnia strukturę MQCNO, klient tworzy duplikat w celu przekazania go do wyjścia, w którym jest on modyfikowany. Klient zachowuje prawo własności do obiektu MQCNO.

Tabela MQCD, do której odwołuje się parametr MQCNO, ma pierwszeństwo przed każdą definicją połączenia udostępnionej przez tablicę. Klient używa struktury MQCNO do łączenia się z menedżerem kolejek, a pozostałe są ignorowane.

KodpComp

Typ: PMQLONG input/output

Kod zakończenia.

Wskaźnik do obiektu MQLONG, który odbiera kod zakończenia wyjścia. Musi to być jedna z następujących wartości:

- `MQCC_OK` -pomyślne zakończenie
- `MQCC_WARNING` -ostrzeżenie (częściowe zakończenie)
- `MQCC_FAILED` -wywołanie nie powiodło się

pReason

Typ: PMQLONG input/output

Przyczyna kwalifikowania kodu pComp.

Wskaźnik do obiektu MQLONG, który odbiera kod przyczyny wyjścia. Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

- MQRC_NONE-(0, x '000') Nie ma powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie C

```
void MQ_PRECONNECT_EXIT (&ExitParms, &QMgrName, &pConnectOpts, &CompCode, &Reason);
```

Parameter

```
PMQNXP  pExitParms   /*PreConnect exit parameter structure*/
PMQCHAR pQMgrName    /*Name of the queue manager*/
PPMQCNO ppConnectOpts/*Options controlling the action of MQCONNX*/
PMQLONG pCompCode    /*Completion code*/
PMQLONG pReason      /*Reason qualifying pCompCode*/
```

Sekcja PreConnect w pliku konfiguracyjnym klienta

Użyj sekcji PreConnect , aby skonfigurować wyjście programu PreConnect w pliku `mqclient.ini` .

W sekcji PreConnect można dołączyć następujące atrybuty:

Data=*< URL>*
Adres URL repozytorium, w którym przechowywane są definicje połączeń. Na przykład podczas korzystania z serwera LDAP:

Dane = ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com

Function=*<myFunc>*
Nazwa punktu wejścia funkcjonalnego w bibliotece, w której znajduje się kod wyjścia PreConnect .

Definicja funkcji jest zgodna z prototypem wyjścia PreConnect (MQ_PRECONNECT_EXIT).

Maksymalna długość tego pola to MQ_EXIT_NAME_LENGTH.

Module=*< amqldapi>*
Nazwa modułu zawierającego kod wyjścia API.

Jeśli w tym polu znajduje się pełna nazwa ścieżki modułu, jest ona używana w takiej postaci.

Sequence=*< numer_kolejny>*
Sekwencja, w której to wyjście jest wywoływane w stosunku do innych wyjść. Wyjście z niskim numerem kolejnym jest wywoływane przed wyjściem z wyższym numerem kolejnym. Nie ma potrzeby, aby numeracja sekwencji wyjść była ciągła; sekwencja 1, 2, 3 ma ten sam wynik co sekwencja 7, 42, 1096. Ten atrybut jest niepodpisaną wartością liczbową.

W pliku `mqclient.ini` można zdefiniować wiele sekcji PreConnect . Kolejność przetwarzania każdego wyjścia jest określana przez atrybut sekwencji sekcji.

Pisanie i kompilowanie wyjść publikowania

Istnieje możliwość skonfigurowania wyjścia publikowania w menedżerze kolejek w celu zmiany treści opublikowanego komunikatu, zanim zostanie on odebrany przez subskrybentów. Można również zmienić nagłówek komunikatu lub nie dostarczyć go do subskrypcji.

Publikowanie wyjść nie jest obsługiwane w systemie z/OS.

Wyjścia publikowania można używać do sprawdzania i modyfikowania komunikatów dostarczanych do subskrybentów:

- Sprawdzanie treści komunikatu publikowanego dla każdego subskrybenta
- Modyfikowanie treści komunikatu publikowanego dla każdego subskrybenta

- Zmień kolejkę, do której jest wstawiany komunikat
- Zatrzymaj dostarczanie komunikatu do subskrybenta

Zapisywanie wyjścia publikowania

Wykonaj kroki opisane w sekcji “Pisanie i kompilowanie wyjść i usług instalowalnych” na stronie 384, aby ułatwić pisanie i kompilowanie wyjścia.

Dostawca wyjścia publikowania definiuje to, co robi wyjście. Wyjście musi jednak być zgodne z regułami zdefiniowanymi w MQPSXP.

Produkt WebSphere MQ nie udostępnia implementacji punktu wejścia MQ_PUBLISH_EXIT. Udostępnia deklarację typu C języka C. Użyj typedef, aby poprawnie zadeklarować parametry do wyjścia napisanego przez użytkownika. W poniższym przykładzie przedstawiono sposób użycia deklaracji typedef:

```
#include "cmqec.h"

MQ_PUBLISH_EXIT MyPublishExit;

void MQENTRY MyPublishExit( PMQPSXP pExitParms,
                            PMQPBC  pPubContext,
                            PMQSBC  pSubContext )
{
  /* C language statements to perform the function of the exit */
}
```

Wyjście publikowania jest uruchamiane w ramach procesu menedżera kolejek w wyniku następujących operacji:

- Operacja publikowania, w której komunikat jest dostarczany do jednego lub większej liczby subskrybentów
- Operacja Subskrybuj, w której jest dostarczany jeden lub więcej zachowanych komunikatów.
- Operacja żądania subskrypcji, w której jeden lub więcej zatrzymanych komunikatów jest dostarczanych

Jeśli wyjście publikowania jest wywoływane dla połączenia, po raz pierwszy jest wywoływany kod *ExitReason* produktu MQXR_INIT . Przed rozłączonym połączeniem po użyciu wyjścia publikowania, wyjście jest wywoływane z kodem *ExitReason* produktu MQXR_TERM.

Jeśli wyjście publikowania jest skonfigurowane, ale nie można go załadować po uruchomieniu menedżera kolejek, operacje komunikatów publikowania/subskrypcji są blokowane dla menedżera kolejek. Należy naprawić problem lub zrestartować menedżer kolejek, aby przesyłanie komunikatów w trybie publikowania/subskrypcji było ponownie włączone.

Każde połączenie produktu WebSphere MQ , które wymaga wyjścia publikowania, może nie zostać załadowane lub zainicjowane wyjście. Jeśli operacja wyjścia nie powiedzie się lub zainicjowano, operacje publikowania/subskrypcji wymagające wyjścia publikowania są wyłączone dla tego połączenia. Operacje nie powiodą się w przypadku kodu przyczyny WebSphere MQ MQRC_PUBLISH_EXIT_ERROR.

Kontekst, w którym wywoływane jest wyjście publikowania, jest połączeniem przez aplikację z menedżerem kolejek. Obszar danych użytkownika jest obsługiwany przez menedżera kolejek dla każdego połączenia, które wykonuje operacje publikowania. Wyjście może zachować informacje w obszarze danych użytkownika dla każdego połączenia.

Wyjście publikowania może korzystać z niektórych wywołań MQI. Mogą one używać tylko tych wywołań MQI, które manipulują właściwościami komunikatów. Połączenia są następujące:

- MQBUFMH
- MQCRTMH
- MQDLTMH
- MQDLTMP
- MQMHBUF
- MQINQMP

- MQSETMP

Jeśli wyjście publikowania zmieni docelowy menedżer kolejek lub nazwę kolejki, nie jest przeprowadzane żadne nowe sprawdzanie uprawnień.

Kompilowanie wyjścia publikowania

Wyjście publikowania jest biblioteką ładowaną dynamicznie; może być ona pomyślana jako wyjście kanału. Więcej informacji na temat kompilowania wyjść zawiera sekcja "Pisanie i kompilowanie wyjść i usług instalowalnych" na stronie 384.

Przykładowe wyjście publikowania

Przykładowy program obsługi wyjścia nosi nazwę `amqspse0.c`. Zapisuje on inny komunikat w pliku dziennika w zależności od tego, czy została wywołana dla operacji inicjowania, publikowania lub zakończenia. Demonstruje również użycie pola obszaru użytkownika obsługi wyjścia w celu odpowiedniego przydzielania i swobodnej pamięci masowej.

Konfigurowanie wyjść publikowania

Aby skonfigurować wyjście publikowania, należy zdefiniować niektóre atrybuty.

W systemie Windows i w produkcie Linux można użyć eksploratora produktu WebSphere MQ w celu zdefiniowania atrybutów. Atrybuty są definiowane na stronie właściwości menedżera kolejek w obszarze Publikowanie/subskrypcja.

Aby skonfigurować wyjście publikowania w pliku `qm.ini` w systemach UNIX i Linux , należy utworzyć sekcję o nazwie `PublishSubscribe`. Sekcja `PublishSubscribe` zawiera następujące atrybuty:

PublishExitPath=[path]|*module_name*
Nazwa modułu i ścieżka zawierająca kod wyjścia publikowania. Maksymalna długość tego pola to `MQ_EXIT_NAME_LENGTH`. Wartością domyślną jest brak wyjścia publikowania.

PublishExitFunction=*function_name*
Nazwa punktu wejścia funkcji w module, który zawiera kod wyjścia publikowania. Maksymalna długość tego pola to `MQ_EXIT_NAME_LENGTH`.

PublishExitData=*string*
Jeśli menedżer kolejek wywołuje wyjście publikowania, przekazuje on strukturę `MQPSXP` jako dane wejściowe. Dane określone za pomocą atrybutu *PublishExitData* są dostępne w polu *ExitData* struktury. Łańcuch może mieć długość do `MQ_EXIT_DATA_LENGTH` znaków. Wartością domyślną są 32 puste znaki.

Pisanie i kompilowanie wyjść obciążenia klastra

Napisz program obsługi wyjścia obciążenia klastra, aby dostosować zarządzanie obciążeniem klastrów. Podczas kierowania komunikatów można wziąć pod uwagę koszt korzystania z kanału o różnych porach dnia lub treści komunikatu. Są to czynniki, które nie są brane pod uwagę przy użyciu standardowego algorytmu zarządzania obciążeniem.

W większości przypadków algorytm zarządzania obciążeniem jest wystarczający dla potrzeb użytkownika. Jednak w celu udostępnienia własnego programu obsługi wyjścia użytkownika w celu dostosowania zarządzania obciążeniem, program WebSphere MQ zawiera wyjście użytkownika, wyjście obciążenia klastra.

Użytkownik może mieć pewne konkretne informacje na temat sieci lub komunikatów, których można użyć w celu wpływania na równoważenie obciążenia. Może się okazać, że są to kanały o dużej pojemności lub tanie trasy sieciowe, lub też można kierować komunikaty w zależności od ich zawartości. Użytkownik może zdecydować się na napisanie programu obsługi wyjścia obciążenia klastra lub użyć jednej z nich dostarczonych przez osobę trzecią.

Wyjście obciążenia klastra jest wywoływane podczas uzyskiwania dostępu do kolejki klastra. Jest on wywoływany przez produkty `MQOPEN`, `MQPUT1` i `MQPUT`.

Docelowy menedżer kolejek wybrany w czasie MQOPEN jest stały, jeśli określono MQOO_BIND_ON_OPEN . W takim przypadku wyjście jest uruchamiane tylko raz.

Jeśli docelowy menedżer kolejek nie jest ustalony w czasie MQOPEN , docelowy menedżer kolejek jest wybierany w czasie wywołania programu MQPUT . Jeśli docelowy menedżer kolejek jest niedostępny lub wystąpił błąd w czasie, gdy komunikat nadal znajduje się w kolejce transmisji, to wyjście jest ponownie wywoływane. Zostanie wybrany nowy docelowy menedżer kolejek. Jeśli kanał komunikatów nie powiedzie się podczas przesyłania komunikatu, a komunikat zostanie wycofany, zostanie wybrany nowy docelowy menedżer kolejek.

Na platformach innych niż z/OSmenedżer kolejek ładuje nowe wyjście obciążenia klastra przy następnym uruchomieniu menedżera kolejek.

Jeśli definicja menedżera kolejek nie zawiera nazwy programu obsługi wyjścia obciążenia klastra, to wyjście obciążenia klastra nie jest wywoływane.

Różne dane są przekazywane do wyjścia obciążenia klastra w strukturze parametru wyjścia, MQWXP:

- Struktura definicji komunikatu MQMD.
- Parametr długości komunikatu.
- Kopia komunikatu lub część komunikatu.

Na platformach innych niżz/OS , jeśli używany jest produkt CLWLMode=FAST, każdy proces systemu operacyjnego ładuje własną kopię wyjścia. Różne połączenia z menedżerem kolejek mogą spowodować wywołanie różnych kopii wyjścia. Jeśli wyjście zostanie uruchomione w domyślnym trybie bezpiecznym, CLWLMode=SAFE, pojedyncza kopia wyjścia działa w osobnym procesie.

Zapisywanie wyjść obciążenia klastra

W przypadku platform innych niż z/OSwyjścia obciążenia klastra nie mogą używać wywołań MQI. W innych aspektach reguły pisania i kompilowania programów obsługi wyjścia obciążenia klastra są podobne do reguł, które mają zastosowanie do programów obsługi wyjścia kanału. Wykonaj kroki opisane w sekcji “Pisanie i kompilowanie wyjść i usług instalowalnych” na stronie 384, a następnie użyj przykładowego programu, “Przykładowe wyjście obciążenia klastra” na stronie 438 , aby pomóc w napisanie i skompilowaniu wyjścia.

Aby uzyskać więcej informacji na temat wyjść kanału, zobacz: “Pisanie programów obsługi wyjścia kanału” na stronie 409.

Konfigurowanie wyjść obciążenia klastra

Należy podać wyjścia obciążenia klastra w definicji menedżera kolejek, podając atrybut wyjścia obciążenia klastra w komendzie ALTER QMGR . Na przykład:

```
ALTER QMGR CLWLEXIT(myexit)
```

Przykładowe wyjście obciążenia klastra

Produkt WebSphere MQ zawiera przykładowy program obsługi wyjścia obciążenia klastra. Można skopiować przykład i użyć go jako podstawy dla własnych programów.

Na platformach innych niż z/OS

Przykładowy program obsługi wyjścia obciążenia klastra jest dostarczany w języku C i jest nazywany amqswlm0.c. Można go znaleźć w:

Tabela 57. Przykładowe położenie programu obsługi wyjścia obciążenia klastra (nie dotyczy systemu z/OS)

Platforma	**Ścieżka do pliku**
AIX, HP-UX, Sun Solaris	*MQ_INSTALLATION_PATH*/samp
Windows	*MQ_INSTALLATION_PATH*\Tools\c\Samples

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

Ten przykładowy program obsługi wyjścia kieruje wszystkie komunikaty do konkretnego menedżera kolejek, chyba że ten menedżer kolejek stanie się niedostępny. Reaguje ona na awarię menedżera kolejek przez kierowanie komunikatów do innego menedżera kolejek.

Wskaż, do którego menedżera kolejek, do którego mają być wysyłane komunikaty. Podaj nazwę kanału odbierającego klastry w atrybucie CLWLDATA w definicji menedżera kolejek. Na przykład:

```
ALTER QMGR CLWLDATA('my-cluster-name.my-queue-manager')
```

Aby włączyć wyjście, podaj jego pełną ścieżkę i nazwę w atrybucie CLWLEXIT :

W systemach UNIX and Linux :

```
ALTER QMGR CLWLEXIT('path/amqswlm(clwlFunction)')
```

W systemie Windows:

```
ALTER QMGR CLWLEXIT('path\amqswlm(clwlFunction)')
```

Teraz, zamiast używać dostarczonego algorytmu zarządzania obciążeniem, produkt WebSphere MQ wywołuje to wyjście, aby skierować wszystkie komunikaty do wybranego menedżera kolejek.

Budowanie aplikacji IBM WebSphere MQ

Ta sekcja zawiera informacje na temat budowania aplikacji produktu IBM WebSphere MQ na różnych platformach.

Budowanie aplikacji w systemie AIX

Publikacje w systemie AIX opisują sposób budowania aplikacji wykonywalnych z napisanych programów.

W tym temacie opisano dodatkowe zadania i zmiany w standardowych zadaniach, które należy wykonać podczas budowania aplikacji WebSphere MQ for AIX w celu uruchomienia w systemie AIX. Obsługiwane są: C, C + + i COBOL. Informacje na temat przygotowywania programów w języku C + + zawiera sekcja Używanie języka C++.

Zadania, które należy wykonać, aby utworzyć aplikację wykonywalną przy użyciu produktu WebSphere MQ for AIX , są różne w zależności od języka programowania, w którym kod źródłowy jest zapisywany. Oprócz kodowania wywołań MQI w kodzie źródłowym, należy dodać odpowiednie instrukcje języka, aby uwzględnić w programie WebSphere MQ for AIX pliki dla używanego języka. Zaznajomić się z zawartością tych plików. Pełny opis znajduje się w sekcji “Pliki definicji danych produktu IBM WebSphere MQ” na stronie 82 .

Podczas uruchamiania wielowątkowego serwera lub wielowątkowych aplikacji klienckich należy ustawić zmienną środowiskową AIXTHREAD_SCOPE = S.

Przygotowywanie programów w języku C w systemie AIX

Ten temat zawiera informacje na temat łączenia bibliotek niezbędnych do przygotowania programów w języku C w systemie AIX.

Wstępnie skompilowane programy w języku C są dostarczane w katalogu *MQ_INSTALLATION_PATH*/samp/bin . Użyj kompilatora ANSI i uruchom następujące komendy. Więcej informacji na temat programowania 64-bitowych aplikacji można znaleźć w sekcji Standardy kodowania na platformach 64-bitowych.

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

Dla aplikacji 32-bitowych:

```
  $ xlc_r -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm
```

gdzie amqsput0 jest programem przykładowym.

Dla aplikacji 64-bitowych:

```
  $ xlc_r -q64 -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64
-lmqm
```

gdzie amqsput0 jest programem przykładowym.

Jeśli używany jest kompilator VisualAge C/C++ dla programów w języku C++, należy uwzględnić opcję -q namemangling=v5 , aby wszystkie symbole WebSphere MQ zostały rozstrzygnięte podczas tworzenia połączeń z bibliotekami.

Aby użyć programów na komputerze, na którym jest zainstalowany tylko klient MQI produktu WebSphere MQ dla systemu AIX , należy ponownie skompilować programy w taki sposób, aby dowiązali je do biblioteki klienta (-lmqic).

Łączenie bibliotek

Potrzebne są następujące biblioteki:

- Powiąż programy z odpowiednią biblioteką udostępnionej przez produkt WebSphere MQ.

  W środowisku innym niż wielowątkowe należy utworzyć odsyłacz do jednej z następujących bibliotek:

Zbiór biblioteki	Typ programu/wyjścia
libmqm.a	Serwer dla C
libmqic.a & libmqm.a	Klient dla C

  W środowisku wielowątkowym należy połączyć się z jedną z następujących bibliotek:

Zbiór biblioteki	Typ programu/wyjścia
libmqm_r.a	Serwer dla C
libmqic_r.a & libmqm_r.a	Klient dla C

  Na przykład, aby zbudować prostą, wielowątkową aplikację WebSphere MQ z pojedynczej jednostki kompilacji, uruchom następujące komendy.

  Dla aplikacji 32-bitowych:

```
  $ xlc_r -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm_r
```

  gdzie amqsput0 jest programem przykładowym.

  Dla aplikacji 64-bitowych:

```
  $ xlc_r -q64 -o amqsputc_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64
-lmqm_r
```

  gdzie amqsput0 jest programem przykładowym.

  Aby użyć programów na komputerze, na którym jest zainstalowany tylko klient MQI produktu WebSphere MQ dla systemu AIX , należy ponownie skompilować programy w taki sposób, aby dowiązali je do biblioteki klienta (-lmqic).

  Uwaga:

1. Jeśli użytkownik zapisuje usługę instalowalną (więcej informacji na ten temat zawiera Administrowanie), należy utworzyć odsyłacz do biblioteki produktu `libmqmzf.a` w aplikacji, która nie jest wielowątkowa, oraz do biblioteki produktu `libmqmzf_r.a` w aplikacji wielowątkowej.
2. W przypadku tworzenia aplikacji na potrzeby zewnętrznej koordynacji za pomocą menedżera transakcji zgodnego z interfejsem XA, takiego jak IBM TXSeries, Encinalub BEA Tuxedo, należy połączyć się z `libmqmxa.a` (lub `libmqmxa64.a` , jeśli menedżer transakcji traktuje typ long jako 64-bitowe) oraz biblioteki produktu `libmqz.a` w aplikacji niewielowątkowej oraz w bibliotekach `libmqmxa_r.a` (lub `libmqmxa64_r.a`) i `libmqz_r.a` w aplikacji wielowątkowej.
3. Należy połączyć zaufane aplikacje z wątkami bibliotek produktu WebSphere MQ . Jednak tylko jeden wątek w zaufanej aplikacji w produkcie WebSphere MQ w systemach UNIX and Linux może być połączony w danym momencie.
4. Przed innymi bibliotekami produktu należy połączyć biblioteki produktu WebSphere MQ .

Przygotowywanie programów w języku COBOL w produkcie AIX

Te informacje są używane podczas przygotowywania programów w języku COBOL w produkcie AIX przy użyciu produktu IBM COBOL Set i Micro Focus COBOL.

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

- 32-bitowe podręczniki do kopiowania w języku COBOL są instalowane w następującym katalogu:

  ```
  MQ_INSTALLATION_PATH/inc/cobcpy32
  ```

  i dowiązania symboliczne są tworzone w:

  ```
  MQ_INSTALLATION_PATH/inc
  ```

- 64-bitowe podręczniki do kopiowania w języku COBOL są instalowane w następującym katalogu:

  ```
  MQ_INSTALLATION_PATH/inc/cobcpy64
  ```

W poniższych przykładach ustaw zmienną środowiskową **COBCPY** na:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

dla aplikacji 32-bitowych, oraz:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

dla aplikacji 64-bitowych.

Musisz połączyć swój program z jednym z następujących plików bibliotecznych:

Zbiór biblioteki	Typ programu/wyjścia
libmqmcb.a	Serwer dla języka COBOL (aplikacja wielowątkowa)
libmqmcb_r.a	Serwer dla języka COBOL (aplikacja wielowątkowa)
libmqicb.a	Klient dla języka COBOL (aplikacja wielowątkowa
libmqicb_r.a	Klient dla języka COBOL (aplikacja wielowątkowa)

Kompilator IBM COBOL Set lub kompilator Micro Focus COBOL można używać w zależności od programu:

- Programy rozpoczynające się od `amqm` są odpowiednie dla kompilatora Micro Focus COBOL, oraz
- Programy rozpoczynające się od `amq0` są odpowiednie dla każdego kompilatora.

Przygotowywanie programów w języku COBOL za pomocą programu IBM COBOL Set for AIX

Przykładowe programy w języku COBOL są dostarczane razem z programem IBM WebSphere MQ. Aby skompilować taki program, wprowadź odpowiednią komendę z następującej listy:

32-bitowa aplikacja serwera, która nie jest wielowątkowa

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb -qLIB \
-I<COBCPY>
```

32-bitowa aplikacja kliencka

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb -qLIB \
-I<COBCPY>
```

32-bitowa aplikacja serwera wielowątkowego

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -I<COBCPY>
```

32-bitowa aplikacja kliencka

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -I<COBCPY>
```

64-bitowa aplikacja serwera bez wątków

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqmcb \
-qLIB -I<COBCPY>
```

64-bitowa aplikacja kliencka, która nie jest wielowątkowa

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqicb \
-qLIB -I<COBCPY>
```

Aplikacja serwera 64-bitowego

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -I<COBCPY>
```

Aplikacja kliencka 64-bitowego

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -I<COBCPY>
```

Przygotowywanie programów w języku COBOL przy użyciu Micro Focus COBOL

Ustaw zmienne środowiskowe przed kompilacją programu w następujący sposób:

```
export COBCPY=<COBCPY>
export LIBPATH=MQ_INSTALLATION_PATH/lib:$LIBPATH
```

Aby skompilować 32-bitowy program w języku COBOL przy użyciu Micro Focus COBOL, wpisz:

- Serwer dla języka COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb
```

- Klient dla języka COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb
```

- Serwer wątków dla języka COBOL

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r
```

- Klient wielowątkowy dla COBOL

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r
```

Aby skompilować 64-bitowy program w języku COBOL przy użyciu programu Micro Focus COBOL, wpisz:

- Serwer dla języka COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb
```

- Klient dla języka COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb
```

- Serwer wątków dla języka COBOL

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r
```

- Klient wielowątkowy dla COBOL

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r
```

gdzie `amqminqx` jest programem przykładowym

Opis zmiennych środowiskowych, które należy skonfigurować, można znaleźć w dokumentacji produktu Micro Focus COBOL.

Przygotowywanie programów aplikacji CICS w systemie AIX

Te informacje są używane podczas przygotowywania programów CICS w programie AIX.

Dostępne są moduły przełączników XA, które umożliwiają połączenie produktu CICS z produktem IBM WebSphere MQ:

Tabela 58. Podstawowy kod dla programów aplikacji CICS w systemie AIX: procedura inicjowania interfejsu XA		
Opis	**C (źródło)**	**C (exec)-dodaj do XAD.Stanza**
Procedura inicjowania interfejsu XA	amqzscix.c	*amqzsc* - CICS dla AIX

Należy użyć gotowej wersji pliku ładowania przełącznika IBM WebSphere MQ *amqzsc* , który jest dostarczany wraz z produktem.

Zawsze dowiąż transakcje C z biblioteką IBM WebSphere MQ wątkowo bezpieczną *libmqm_r.a.*, oraz transakcje w języku COBOL za pomocą biblioteki COBOL *libmqmcb_r.a.*.

Więcej informacji na temat obsługi transakcji CICS można znaleźć w Administrowanie.

Obsługa programu TXSeries CICS

Produkt IBM WebSphere MQ na serwerze AIX obsługuje interfejs TXSeries CICS za pomocą interfejsu XA. Upewnij się, że aplikacje produktu CICS są dowiązane do wielowątkowej wersji bibliotek produktu IBM WebSphere MQ .

Programy CICS można uruchamiać za pomocą zestawu IBM COBOL Set for AIX lub Micro Focus COBOL. W poniższych sekcjach opisano różnice między uruchomieniem programów CICS w produkcie IBM COBOL Set for AIX i Micro Focus COBOL.

Napisz programy WebSphere MQ , które są ładowane do tego samego regionu CICS w języku C lub COBOL. Nie można utworzyć połączenia wywołań interfejsu MQI języka C i języka COBOL w tym samym regionie CICS . Większość wywołań MQI w drugim języku nie powiodła się z kodem przyczyny `MQRC_HOBJ_ERROR`.

Przygotowywanie programów w języku COBOL produktu CICS przy użyciu zestawu COBOL produktu IBM dla produktu AIX

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

Aby użyć języka COBOL produktu IBM , wykonaj następujące kroki:

1. Wyeksportuj następującą zmienną środowiskową:

   ```
   export LDFLAGS="-qLIB -bI:/usr/lpp/cics/lib/cicsprIBMCOB.exp  \
                   -IMQ_INSTALLATION_PATH/inc -I/usr/lpp/cics/include  \
                   -e _iwz_cobol_main  \
   ```

   gdzie `LIB` jest dyrektywą kompilatora.
2. Konwertuj, skompiluj i dowiązaj program wpisując:

   ```
   cicstcl -l IBMCOB <yourprog>.ccp
   ```

Przygotowywanie programów w języku COBOL produktu CICS przy użyciu programu Micro Focus COBOL

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

Aby użyć programu Micro Focus COBOL, należy wykonać następujące czynności:

1. Dodaj moduł biblioteki środowiska wykonawczego produktu IBM WebSphere MQ COBOL do biblioteki środowiska wykonawczego za pomocą następującej komendy:

```
cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
            MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
```

   Uwaga: W przypadku produktu `cicsmkcobol`produkt IBM WebSphere MQ nie umożliwia tworzenia wywołań MQI w języku programowania w języku C z poziomu aplikacji COBOL.

   Jeśli w istniejących aplikacjach istnieją takie wywołania, zalecane jest przeniesienie tych funkcji z aplikacji w języku COBOL do własnej biblioteki, na przykład `myMQ.so`. Po przeniesieniu funkcji nie należy dołączać biblioteki IBM WebSphere MQ `libmqmcbrt.o` podczas budowania aplikacji COBOL dla produktu CICS.

   Ponadto, jeśli aplikacja COBOL nie wykonuje żadnego wywołania MQI COBOL, nie należy łączyć `libmqmz_r` z `cicsmkcobol`.

   Spowoduje to utworzenie pliku metody języka Micro Focus COBOL i włączenie biblioteki środowiska wykonawczego języka COBOL produktu CICS w celu wywołania produktu IBM WebSphere MQ w systemach UNIX and Linux .

   Uwaga: Uruchom program `cicsmkcobol` tylko wtedy, gdy instalowany jest jeden z następujących produktów:

   - Nowa wersja lub wydanie Micro Focus COBOL
   - Nowa wersja lub wydanie produktu CICS dla produktu AIX
   - Nowa wersja lub wydanie dowolnego obsługiwanego produktu bazodanowego (tylko dla transakcji w języku COBOL)
   - Nowa wersja lub wydanie produktu IBM WebSphere MQ
2. Wyeksportuj następującą zmienną środowiskową:

```
COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
```

3. Konwertuj, skompiluj i dowiązaj program wpisując:

```
cicstcl -l COBOL -e <yourprog>.ccp
```

Przygotowywanie programów CICS C

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

Kompilowanie programów CICS C przy użyciu standardowych narzędzi CICS :

1. Wyeksportuj ***jeden*** z następujących zmiennych środowiskowych:
   - LDFLAGS = "-L/*MQ_INSTALLATION_PATH*lib -lmqm_r" export LDFLAGS
   - USERLIB = "-L*MQ_INSTALLATION_PATH*lib -lmqm_r" export USERLIB
2. Konwertuj, skompiluj i dowiązaj program wpisując:

```
cicstcl -l C amqscic0.ccs
```

Przykładowa transakcja CICS C

Przykładowe źródło C dla transakcji AIX IBM WebSphere MQ jest udostępniane przez program AMQSCIC0.CCS. Transakcja odczytuje komunikaty z kolejki transmisji SYSTEM.SAMPLE.CICS.WORKQUEUE w domyślnym menedżerze kolejek i umieszcza je w kolejce lokalnej z nazwą kolejki, która jest zawarta w nagłówku transmisji komunikatu. Wszystkie

niepowodzenia są wysyłane do kolejki SYSTEM.SAMPLE.CICS.DLQ. Użyj przykładowego skryptu MQSC AMQSCIC0.TST , aby utworzyć te kolejki i przykładowe kolejki wejściowe.

Budowanie aplikacji w systemie HP Integrity NonStop Server

Te informacje opisują dodatkowe zadania oraz zmiany w standardowych zadaniach, które należy wykonać podczas budowania klienta IBM WebSphere MQ dla aplikacji HP Integrity NonStop Server w celu uruchomienia w ramach produktu HP Integrity NonStop Server.

Obsługiwane są języki C, COBOL i pTAL .

Nagłówki OSS i Guardian oraz biblioteki publiczne

Udostępnia listy nagłówków OSS i Guardian oraz bibliotek publicznych. Wyświetlane są nagłówki OSS, publiczne biblioteki wykonywalne OSS oraz publiczne biblioteki importu, nagłówki Guardian i publiczne biblioteki programu Guardian oraz publiczne biblioteki importu.

“Nagłówki OSS” na stronie 446

“Publiczne biblioteki programu OSS i publiczne biblioteki importu” na stronie 447

“Nagłówki kuratora” na stronie 447

“Publiczne biblioteki wykonywalne i publiczne biblioteki importu” na stronie 448

Nagłówki OSS

Tabela 59. Nagłówki OSS

Obiekt	Położenie	Opis
cmqbc.h	<mqinstall>/inc	IBM WebSphere MQ C-language header (OSS) (nagłówek języka C)
cmqc.h	<mqinstall>/inc	IBM WebSphere MQ C-language header (OSS) (nagłówek języka C)
cmqfc.h	<mqinstall>/inc	IBM WebSphere MQ C-language header (OSS) (nagłówek języka C)
cmqec.h	<mqinstall>/inc	IBM WebSphere MQ C-language header (OSS) (nagłówek języka C)
cmqpsc.h	<mqinstall>/inc	IBM WebSphere MQ C-language header (OSS) (nagłówek języka C)
cmqxc.h	<mqinstall>/inc	IBM WebSphere MQ C-language header (OSS) (nagłówek języka C)
cmqzc.h	<mqinstall>/inc	IBM WebSphere MQ C-language header (OSS) (nagłówek języka C)
cmqcobol.cpy	<mqinstall>/inc	IBM WebSphere MQ COBOL copybook (OSS)
cmqbt.tal	<mqinstall>/inc	Nagłówek IBM WebSphere MQ pTAL (OSS)
cmqcft.tal	<mqinstall>/inc	Nagłówek IBM WebSphere MQ pTAL (OSS)
cmqpst.tal	<mqinstall>/inc	Nagłówek IBM WebSphere MQ pTAL (OSS)

Tabela 59. Nagłówki OSS (kontynuacja)		
Obiekt	**Położenie**	**Opis**
cmqt.tal	<mqinstall>/inc	Nagłówek IBM WebSphere MQ pTAL (OSS)
cmqxt.tal	<mqinstall>/inc	Nagłówek IBM WebSphere MQ pTAL (OSS)

Publiczne biblioteki programu OSS i publiczne biblioteki importu

Tabela 60. Publiczne biblioteki programu OSS i publiczne biblioteki importu		
Obiekt	**Położenie**	**Opis**
libmqic.so	<mqinstall>/bin	Publiczna biblioteka wykonywalna IBM WebSphere MQ (OSS ungwintowana)
libmqic_r.so	<mqinstall>/bin	Publiczna biblioteka programu IBM WebSphere MQ (wielowątkowa OSS)
libmqic.so	<mqinstall>/lib	IBM WebSphere MQ publiczna biblioteka importu (OSS ungwintowana)
libmqic_r.so	<mqinstall>/lib	Biblioteka publiczna importu IBM WebSphere MQ (wielowątkowe OSS)
mqicb	<mqinstall>/lib	Publiczna biblioteka importu produktu IBM WebSphere MQ dla języka COBOL (OSS)

Nagłówki kuratora

Tabela 61. Nagłówki kuratora		
Obiekt	**Położenie**	**Opis**
cmqbch	<mqinstall>/inc/G	IBM WebSphere MQ C-nagłówek języka (Guardian)
cmqch	<mqinstall>/inc/G	IBM WebSphere MQ C-nagłówek języka (Guardian)
cmqfch	<mqinstall>/inc/G	IBM WebSphere MQ C-nagłówek języka (Guardian)
cmqech	<mqinstall>/inc/G	IBM WebSphere MQ C-nagłówek języka (Guardian)
cmqpsch	<mqinstall>/inc/G	IBM WebSphere MQ C-nagłówek języka (Guardian)
cmqxch	<mqinstall>/inc/G	IBM WebSphere MQ C-nagłówek języka (Guardian)
cmqzch	<mqinstall>/inc/G	IBM WebSphere MQ C-nagłówek języka (Guardian)

Tabela 61. Nagłówki kuratora (kontynuacja)		
Obiekt	**Położenie**	**Opis**
cmqcobol	<mqinstall>/inc/G	IBM WebSphere MQ COBOL copybook (Guardian)
cmqbt	<mqinstall>/inc/G	Nagłówek IBM WebSphere MQ pTAL (Guardian)
cmqcft	<mqinstall>/inc/G	Nagłówek IBM WebSphere MQ pTAL (Guardian)
cmqpst	<mqinstall>/inc/G	Nagłówek IBM WebSphere MQ pTAL (Guardian)
cmqt	<mqinstall>/inc/G	Nagłówek IBM WebSphere MQ pTAL (Guardian)
cmqxt	<mqinstall>/inc/G	Nagłówek IBM WebSphere MQ pTAL (Guardian)

Publiczne biblioteki wykonywalne i publiczne biblioteki importu

Tabela 62. Publiczne biblioteki wykonywalne i publiczne biblioteki importu		
Obiekt	**Położenie**	**Opis**
mqic	<mqinstall>/bin/G	Publiczna biblioteka programu IBM WebSphere MQ (Guardian)
mqicb	<mqinstall>/lib/G	Publiczna biblioteka importu produktu IBM WebSphere MQ dla języka COBOL (Guardian)

Przygotowywanie programów w języku C w programie HP Integrity NonStop Server

Ten temat zawiera informacje, które należy wziąć pod uwagę podczas przygotowywania programów w języku C w programie HP Integrity NonStop Server wraz z przykładami komend, które są używane podczas tworzenia aplikacji podczas korzystania z kompilatora OSS C, a także w przypadku korzystania z kompilatora Guardian C.

Wstępnie skompilowane programy w języku C są dostarczane w katalogu `MQ_INSTALLATION_PATH/opt/mqm/samp/bin` . Aby zbudować przykład z kodu źródłowego, należy użyć kompilatora c89 .

Należy połączyć programy z odpowiednią biblioteką udostępnionej przez program IBM WebSphere MQ. W poniższej tabeli znajduje się lista bibliotek, z którymi należy się połączyć podczas przygotowywania programów w języku C w systemie HP Integrity NonStop Server.

Tabela 63. . Biblioteki odsyłaczy produktu HP Integrity NonStop Server	
Biblioteka	**Opis**
libmqic.so	Ungwintowane OSS
libmqic_r.so	Wielowątkowy OSS
mqic	Strażnik

Wielowątkowe rodzime aplikacje produktu IBM WebSphere MQ muszą korzystać z funkcji Wątki użytkownika Posix (PUT). W tym produkcie nie ma wsparcia dla standardowych wątków pozyx (Standard Posix Threads-SPT).

Budowanie aplikacji przy użyciu kompilatora OSS C

Ta sekcja zawiera przykłady komend używanych do budowania programów, które są przeznaczone dla OSS lub Guardian, gdy używany jest kompilator OSS.

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

W poniższym przykładzie kompilacja aplikacji OSS klienta C jest niewielowątkowa:

```
c89 -Wsystype=oss -o amqsputc amqsput0.c -IMQ_INSTALLATION_PATH/opt/mqm/inc
-LMQ_INSTALLATION_PATH/opt/mqm/lib -lmqic
```

W poniższym przykładzie przedstawiono budowanie wielowątkowej aplikacji OSS klienta C:

```
c89 -Wsystype=oss -D_PUT_MODEL_ -o amqsputc amqsput0.c -IMQ_INSTALLATION_PATH/opt/mqm/inc
-LMQ_INSTALLATION_PATH/opt/mqm/lib -lmqic_r -lput
```

W poniższym przykładzie przedstawiono budowanie aplikacji klienta Guardian C:

```
c89 -Wsystype=guardian -o /G/vol/subvol/amqsputc amqsput0.c -IMQ_INSTALLATION_PATH/opt/mqm/inc
-LMQ_INSTALLATION_PATH/opt/mqm/lib/G -lmqic
```

Budowanie aplikacji przy użyciu kompilatora Guardian C

W tej sekcji znajdują się przykłady komend używanych do budowania programów przeznaczonych dla Guardiana, gdy używany jest kompilator Guardian.

MQ_INSTALLATION_PATH reprezentuje wolumin i podwolumin Guardian, w którym zainstalowany jest produkt IBM WebSphere MQ .

W poniższym przykładzie przedstawiono budowanie aplikacji klienta Guardian C:

```
CCOMP /in AMQSPUT0/ AMQSPUTC;&
       runnable,systype guardian,nolist,&
       ssv0 "$system.system",&
       ssv1 "MQINSTALLATION_SUBVOL",&
       ld(-LMQINSTALLATION_SUBVOL -lmqic)
```

Przygotowywanie programów w języku COBOL

Ten temat zawiera informacje, które należy wziąć pod uwagę podczas przygotowywania programów w języku C dla klienta IBM WebSphere MQ dla produktu HP Integrity NonStop Server. Zawiera ona przykłady komend, które są używane podczas tworzenia aplikacji podczas używania kompilatora OSS ECOBOL, a także w przypadku korzystania z kompilatora Guardian ECOBOL.

Aby zbudować przykład COBOL z kodu źródłowego, należy użyć kompilatora ECOBOL.

W poniższej tabeli znajduje się lista bibliotek, które są wymagane podczas przygotowywania programów w języku COBOL w systemie HP Integrity NonStop Server. Należy połączyć programy z odpowiednią biblioteką udostępnionej przez program IBM WebSphere MQ.

Tabela 64. . Biblioteki odsyłaczy produktu HP Integrity NonStop Server

Biblioteka	**Opis**
libmqic.so	Ungwintowane OSS
mqic	Strażnik

Podczas uruchamiania aplikacji w języku COBOL, która łączy się z menedżerem kolejek, należy najpierw ustawić zmienną *SAVE-ENVIRONMENT* na wartość 0N. Aby ustawić zmienną *SAVE-ENVIRONMENT* na wartość 0N:

- W systemie OSS wpisz następującą komendę:

```
export SAVE-ENVIRONMENT=ON
```

- W przypadku programu guardian wprowadź następującą komendę:

```
param SAVE-ENVIRONMENT ON
```

Jeśli zmienna *SAVE-ENVIRONMENT* nie zostanie ustawiona na wartość 0N, to gdy aplikacja podejmie próbę nawiązania połączenia z menedżerem kolejek, nie powiedzie się ona z kodem przyczyny 2058 (080A) (RC2058): MQRC_Q_MGR_NAME_ERROR.

Budowanie aplikacji przy użyciu kompilatora OSS ECOBOL

W tej sekcji znajdują się przykłady komend używanych do budowania programów, których celem jest OSS lub Guardian, w przypadku korzystania z kompilatora OSS ECOBOL.

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

W poniższym przykładzie przedstawiono budowanie aplikacji OSS klienta COBOL:

```
ecobol -Wsystype=oss
       -Wcobol="ansi;port"
       -Wcobol="consult MQ_INSTALLATION_PATH/opt/mqm/lib/mqicb"
       -Wcopylib=MQ_INSTALLATION_PATH/opt/mqm/inc/cmqcobol.cpy
       -LMQ_INSTALLATION_PATH/opt/mqm/lib -lmqic
       -o amq0put0
       MQ_INSTALLATION_PATH/opt/mqm/samp/amq0put0.cbl
```

W poniższym przykładzie przedstawiono budowanie aplikacji IBM Guardian w języku COBOL:

```
ecobol -Wsystype=guardian
       -Wcobol="ansi;port;save all"
       -Wcobol="consult MQ_INSTALLATION_PATH/opt/mqm/lib/mqicb"
       -Wcopylib=MQ_INSTALLATION_PATH/opt/mqm/inc/cmqcobol.cpy
       -LMQ_INSTALLATION_PATH/opt/mqm/lib/G -lmqic
       -o amq0put0
       MQ_INSTALLATION_PATH/opt/mqm/samp/amq0put0.cbl
```

Budowanie aplikacji przy użyciu kompilatora Guardian ECOBOL

Ta sekcja zawiera przykłady komend używanych do budowania programów, które są kierowane do Guardiana, gdy używany jest kompilator ECOBOL Guardian.

MQ_INSTALLATION_SUBVOL reprezentuje wolumin i podwolumin Guardian, w którym zainstalowany jest produkt IBM WebSphere MQ .

W poniższym przykładzie przedstawiono budowanie aplikacji IBM Guardian w języku COBOL:

```
ECOBOL /in MQSPUTL/ MQSPUT,MQINSTALLATION_SUBVOL.cmqcobol;
       call-shared;ansi;port;save all;nolist;runnable;
       consult MQINSTALLATION_SUBVOL.mqicb;
       eld(-LMQINSTALLATION_SUBVOL -lmqic)
```

Przygotowywanie programów pTAL

Sekcja zawiera informacje na temat budowania programów pTAL dla klienta IBM WebSphere MQ na platformie HP Integrity NonStop Server .

Aby zbudować próbę pTAL z kodu źródłowego, należy użyć kompilatora EPTAL.

Uwaga:

- Aplikacje pTAL IBM WebSphere MQ muszą używać głównej procedury, która jest napisana w językach C lub COBOL.
- Aplikacje pTAL mogą być budowane tylko w Guardian.

Poniższa tabela zawiera listę bibliotek, które są wymagane podczas przygotowywania programów pTAL w systemie HP Integrity NonStop Server. Należy połączyć programy z odpowiednią biblioteką udostępnionej przez program IBM WebSphere MQ.

Tabela 65. . Biblioteka odsyłaczy produktu HP Integrity NonStop Server	
Biblioteka	**Opis**
mqic	Strażnik

Budowanie aplikacji przy użyciu kompilatora Guardian EPTAL

Ta sekcja zawiera przykłady komend używanych do budowania programów, które są kierowane do Guardiana, gdy używany jest kompilator EPTAL Guardian.

MQINSTALLATION_SUBVOL reprezentuje wolumin i podwolumin Guardian, w którym zainstalowany jest produkt IBM WebSphere MQ .

Aplikacje pTAL IBM WebSphere MQ muszą używać głównej procedury, która jest napisana w językach C lub COBOL.

W poniższym przykładzie przedstawiono budowanie aplikacji Guardian klienta pTAL :

```
ASSIGN SSV0, $SYSTEM.SYSTEM
ASSIGN SSV1, MQINSTALLATION_SUBVOL

EPTAL /in MQINSTALLATION_SUBVOL.MQSPUTT/ MQSPUTO;nolist

CCOMP /in MQINSTALLATION_SUBVOL.MQSPTMC/ MQSPUT;
       runnable,systype guardian,extensions,nolist,
       ssv0 "$system.system",
       ssv1 "MQINSTALLATION_SUBVOL",
       eld(MQSPUTO -LMQINSTALLATION_SUBVOL -lmqic)
```

Budowanie aplikacji w systemie HP-UX

Te informacje opisują dodatkowe zadania i zmiany w standardowych zadaniach, które należy wykonać podczas budowania aplikacji WebSphere MQ for HP-UX w celu uruchomienia w systemie HP-UX.

Obsługiwane są: C, C + + i COBOL. Informacje na temat przygotowywania programów w języku C + + zawiera sekcja Używanie języka C++.

Zadania, które należy wykonać, aby utworzyć aplikację wykonywalną przy użyciu produktu WebSphere MQ for HP-UX , różnią się w zależności od języka programowania, w którym jest zapisany kod źródłowy. Oprócz kodowania wywołań MQI w kodzie źródłowym, należy dodać odpowiednie instrukcje języka, aby uwzględnić w programie WebSphere MQ for HP-UX pliki dla używanego języka. Zaznajomić się z zawartością tych plików. Pełny opis znajduje się w sekcji “Pliki definicji danych produktu IBM WebSphere MQ” na stronie 82 .

W tym temacie użyjemy znaku ukośnika odwrotnego (\), aby podzielić długie komendy na więcej niż jedną linię. Nie należy wprowadzać tego znaku; do każdej komendy należy wprowadzić jedną linię.

Przygotowywanie programów w języku C w systemie HP-UX

Ten temat zawiera informacje, które należy wziąć pod uwagę podczas przygotowywania programów w języku C w systemie HP-UX; z przykładami dla platformy IA64 (IPF).

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

Praca w normalnym środowisku. Wstępnie skompilowane programy w języku C są dostarczane w katalogu *MQ_INSTALLATION_PATH*/samp/bin .

Więcej informacji na temat programowania aplikacji 64-bitowych można znaleźć w sekcji Standardy kodowania na platformach 64-bitowych.

Aby można było używać protokołu SSL, klienty MQI produktu WebSphere MQ w systemie HP-UX muszą być budowane za pomocą wątków POSIX .

Oto kilka przykładów do rozważenia:

- "Platforma IA64 (IPF)" na stronie 452
- "Łączenie bibliotek" na stronie 454

Platforma IA64 (IPF)

Przykłady budowania platformy amqsput0, cliexit i srvexit na platformie IA64(IPF).

W poniższym przykładzie przykładowy program amqsput0 jest kompilowany jako aplikacja kliencka w niewielowątkowym środowisku 32-bitowym:

```
c89 -Wl,+b,: +e -D_HPUX_SOURCE -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqic
```

W poniższym przykładzie przykładowy program amqsput0 jest kompilowany jako aplikacja kliencka w wielowątkowym środowisku 32-bitowym:

```
c89 -mt -Wl,+b,: +e -D_HPUX_SOURCE -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqic_r -lpthread
```

W poniższym przykładzie przykładowy program amqsput0 jest kompilowany jako aplikacja kliencka w 64-bitowej wersji środowiska 64-bitowego:

```
c89 +DD64 +e -D_HPUX_SOURCE -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic
```

W poniższym przykładzie przykładowy program amqsput0 jest kompilowany jako aplikacja kliencka w wielowątkowym środowisku 64-bitowym:

```
c89 -mt +DD64 +e -D_HPUX_SOURCE -o amqsputc_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic_r -lpthread
```

W poniższym przykładzie przykładowy program amqsput0 jest kompilowany jako aplikacja serwera w niewielowątkowym środowisku 32-bitowym:

```
c89 -Wl,+b,: +e -D_HPUX_SOURCE -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqm
```

W poniższym przykładzie przykładowy program amqsput0 jest kompilowany jako aplikacja serwera w wielowątkowym środowisku 32-bitowym:

```
c89 -mt -Wl,+b,: +e -D_HPUX_SOURCE -o amqsput_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqm_r -lpthread
```

W poniższym przykładzie przykładowy program amqsput0 jest kompilowany jako aplikacja serwera w 64-bitowej wersji środowiska 64-bitowego:

```
c89 +DD64 +e -D_HPUX_SOURCE -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm
```

W poniższym przykładzie przykładowy program amqsput0 jest kompilowany jako aplikacja serwera w wielowątkowym środowisku 64-bitowym:

```
c89 -mt +DD64 +e -D_HPUX_SOURCE -o amqsput_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm_r -lpthread
```

W poniższym przykładzie przedstawiono kompilację programu zewnętrznego cliexit klienta w niewielowątkowym środowisku 32-bitowym:

```
c89 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_32 -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqic
```

W poniższym przykładzie przedstawiono kompilację programu zewnętrznego cliexit klienta w wielowątkowym środowisku 32-bitowym:

```
c89 -mt +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_32_r -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqic_r -lpthread
```

W poniższym przykładzie przedstawiono kompilację programu zewnętrznego cliexit klienta w niewielowątkowym środowisku 64-bitowym:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld -b cliexit.o +ee MQStart -o /var/mqm/exits64/cliexit_64 \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic
```

W poniższym przykładzie przedstawiono kompilację programu zewnętrznego cliexit klienta w wielowątkowym środowisku 64-bitowym:

```
c89 -mt +DD64 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld  -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_64_r \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic_r -lpthread
```

W poniższym przykładzie przedstawiono budowanie wyjścia serwera srvexit w środowisku 32-bitowym bez obsługi wątków:

```
c89 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_32 -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqm
```

W poniższym przykładzie przedstawiono budowanie wyjścia serwera srvexit w wielowątkowym środowisku 32-bitowym:

```
c89 -mt +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_32_r -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqm_r -lpthread
```

W poniższym przykładzie przedstawiono budowanie wyjścia serwera srvexit w 64-bitowej wersji środowiska 64-bitowego:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c
-IMQ_INSTALLATION_PATHMQ_INSTALLATION_PATH/inc
ld -b srvexit.o +ee MQStart -o /var/mqm/exits64/srvexit_64 \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm
```

W poniższym przykładzie przedstawiono budowanie wyjścia serwera srvexit w wielowątkowym środowisku 64-bitowym:

```
c89 -mt +DD64 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -IMQ_INSTALLATION_PATH/inc
ld  -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_64_r \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm_r -lpthread
```

Łączenie bibliotek

Konieczne jest połączenie programów z odpowiednią biblioteką udostępnianej przez produkt WebSphere MQ.

W poniższej tabeli przedstawiono bibliotekę, która ma być używana w różnych środowiskach.

Platforma sprzętowa	Środowisko wielowątkowe lub niewielowątkowe	Typ programu/wyjścia	Zbiór biblioteki
IA64 (IPF)	Wątkowy	Serwer i klient dla C	libmqm_r.so
IA64 (IPF)	Wątkowy	Klient dla C	libmqic_r.so
IA64 (IPF)	Bez wątków	Serwer i klient dla C	libmqm.so
IA64 (IPF)	Bez wątków	Klient dla C	libmqic.so

Uwaga:

1. Jeśli użytkownik zapisuje usługę instalowalną (więcej informacji na ten temat zawiera sekcja Administrowanie), należy utworzyć odsyłacz do biblioteki produktu `libmqmzf.sl` .
2. W przypadku tworzenia aplikacji na potrzeby zewnętrznej koordynacji za pomocą menedżera transakcji zgodnego z interfejsem XA, takiego jak IBM TXSeries Encinalub BEA Tuxedo, należy połączyć się z `libmqmxa.sl` (lub `libmqmxa64.sl` , jeśli menedżer transakcji traktuje typ long jako 64-bitowe) i biblioteki produktu `libmqz.sl` w aplikacji niewielowątkowej oraz w bibliotekach `libmqmxa_r.sl` (lub `libmqmxa64_r.sl`) i `libmqz_r.sl` w aplikacji wielowątkowej.
3. Przed innymi bibliotekami produktu należy połączyć biblioteki produktu WebSphere MQ .

Przygotowywanie programów w języku COBOL w systemie HP-UX

Sekcja zawiera informacje na temat przygotowywania programów w języku COBOL w systemie HP-UXprzy użyciu produktu Micro Focus Server Express z produktem WebSphere MQ na platformie IA64 (IPF) oraz uruchamiania programów w środowisku klienta MQI produktu WebSphere MQ .

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

Uwagi do użytkowników

1. 32-bitowe podręczniki do kopiowania w języku COBOL są instalowane w następującym katalogu:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   i dowiązania symboliczne są tworzone w:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. 64-bitowe podręczniki do kopiowania w języku COBOL są instalowane w następującym katalogu:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. W poniższych przykładach ustaw COBCPY na:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

dla aplikacji 32-bitowych, oraz:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

dla aplikacji 64-bitowych.

Skompiluj programy, korzystając z kompilatora Micro Focus. Pliki kopii, które deklarują struktury, znajdują się w katalogu *MQ_INSTALLATION_PATH*/inc:

```
$ export LIB=MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="<COBCPY>"
```

Kompilowanie 32-bitowych programów:

```
$ cob32 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb Server for COBOL
$ cob32 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb Client for COBOL
$ cob32 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r Threaded Server for COBOL
$ cob32 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r Threaded Client for COBOL
```

Kompilowanie 64-bitowych programów:

```
$ cob64 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb Server for COBOL
$ cob64 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb Client for COBOL
$ cob64 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r Threaded Server for COBOL
$ cob64 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r Threaded Client for COBOL
```

gdzie `amqsput` jest programem przykładowym

Należy upewnić się, że określono odpowiednie wielkości stosu środowiska wykonawczego. Minimalna zalecana wielkość to 16 kB.

Konieczne jest połączenie programów z odpowiednią biblioteką udostępnianej przez produkt WebSphere MQ. W poniższej tabeli przedstawiono bibliotekę, która ma być używana w różnych środowiskach.

Platforma sprzętowa	Typ programu/wyjścia	Zbiór biblioteki
IA64 (IPF)	Serwer dla języka COBOL	libmqmcb.so
IA64 (IPF)	Klient dla języka COBOL	libmqicb.so
IA64 (IPF)	Aplikacje gwintowane	libmqmcb_r.so

Korzystanie z produktu Micro Focus Server Express z produktem WebSphere MQ na platformie IA64 (IPF)

Szczegółowe informacje na temat korzystania z programu Micro Focus Server Express w połączeniu z produktem WebSphere MQ na platformie HP/IPF zawiera sekcja “Modele przestrzeni adresowej obsługiwane przez produkt WebSphere MQ for HP-UX na platformie IA64 (IPF)” na stronie 457 .

Programy do uruchomienia w środowisku klienta MQI produktu WebSphere MQ

Jeśli do łączenia klienta MQI z serwerem używana jest jednostka logiczna 6.2 , należy połączyć aplikację z plikiem libsna.a, częścią produktu SNAplusAPI . Użyj opcji `-lV3` i `-lstr` w komendzie kompilowania i łączenia.

- Opcja `-lV3` umożliwia programowi dostęp do biblioteki sygnalizacji AT & T (funkcja SNAplusAPI używa sygnałów AT & T)
- Opcja `-lstr` łączy swój program z komponentem strumienie

Przygotowywanie programów CICS w systemie HP-UX

Dowiedz się, jak budować programy transakcyjne CICS w systemie HP-UX.

Aby zbudować przykładową transakcję CICS (amqscic0.ccs), uruchom następującą komendę:

```
$ export USERLIB="-lmqm_r"
$ cicstcl -l C amqscic0.ccs
```

Dostępny jest moduł przełącznika XA, który umożliwia połączenie programu CICS z produktem WebSphere MQ:

Tabela 66. Podstawowy kod dla aplikacji CICS (HP-UX)

Opis	**C (źródło)**	**C (exec)**
Procedura inicjowania interfejsu XA	amqzscix.c	amqzsc

Więcej informacji na temat obsługi transakcji CICS można znaleźć w Administrowanie.

Obsługa programu TXSeries CICS

Produkt WebSphere MQ w systemie HP-UX obsługuje program TXSeries CICS przy użyciu interfejsu XA. Upewnij się, że aplikacje CICS są połączone z wielowątkową wersją bibliotek produktu MQ .

Napisz programy WebSphere MQ , które są ładowane do tego samego regionu CICS w języku C lub COBOL. Nie można utworzyć połączenia wywołań interfejsu MQI języka C i języka COBOL w tym samym regionie CICS . Większość wywołań MQI w drugim języku nie powiodła się z kodem przyczyny `MQRC_HOBJ_ERROR`.

Przykładowa transakcja CICS C

Sample C source for a CICS WebSphere MQ transaction is provided by AMQSCIC0.CCS. Transakcja odczytuje komunikaty z kolejki transmisji SYSTEM.SAMPLE.CICS.WORKQUEUE w domyślnym menedżerze kolejek i umieszcza je w kolejce lokalnej z nazwą kolejki, która jest zawarta w nagłówku transmisji komunikatu. Wszystkie niepowodzenia są wysyłane do kolejki SYSTEM.SAMPLE.CICS.DLQ. Użyj przykładowego skryptu MQSC AMQSCIC0.TST , aby utworzyć te kolejki i przykładowe kolejki wejściowe.

Przygotowywanie programów w języku COBOL programu CICS przy użyciu programu Micro Focus COBOL

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

Aby użyć programu Micro Focus COBOL, należy wykonać następujące czynności:

1. Dodaj moduł biblioteki środowiska wykonawczego języka COBOL produktu WebSphere MQ do biblioteki środowiska wykonawczego za pomocą następującej komendy:

   ```
   cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
               MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
   ```

   Uwaga: W produkcie `cicsmkcobol`produkt WebSphere MQ nie zezwala na wywozy wywołań MQI w języku programowania C z poziomu aplikacji COBOL.

   Jeśli w istniejących aplikacjach istnieją takie wywołania, zalecane jest przeniesienie tych funkcji z aplikacji w języku COBOL do własnej biblioteki, na przykład `myMQ.so`. Po przeniesieniu tych funkcji nie należy uwzględniać biblioteki produktu WebSphere MQ `libmqmcbrt.o` podczas budowania aplikacji w języku COBOL dla programu CICS.

   Ponadto, jeśli aplikacja COBOL nie wykonuje żadnego wywołania MQI COBOL, nie należy łączyć `libmqmz_r` z `cicsmkcobol`.

Spowoduje to utworzenie pliku metody języka Micro Focus COBOL i włączenie biblioteki COBOL środowiska wykonawczego CICS w celu wywołania produktu WebSphere MQ w systemach UNIX and Linux .

Uwaga: Uruchom program `cicsmkcobol` tylko wtedy, gdy instalowany jest jeden z następujących produktów:

- Nowa wersja lub wydanie Micro Focus COBOL
- Nowa wersja lub wydanie programu CICS for HP-UX
- Nowa wersja lub wydanie dowolnego obsługiwanego produktu bazodanowego (tylko dla transakcji w języku COBOL)
- Nowa wersja lub wydanie produktu WebSphere MQ

2. Wyeksportuj następującą zmienną środowiskową:

```
COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
```

3. Konwertuj, skompiluj i dowiązaj program wpisując:

```
cicstcl -l COBOL -e <yourprog>.ccp
```

Modele przestrzeni adresowej obsługiwane przez produkt WebSphere MQ for HP-UX na platformie IA64 (IPF)

System HP-UX udostępnia kilka modeli przestrzeni adresowej, które mogą być wykorzystywane przez aplikacje produktu WebSphere MQ .

System HP-UX obsługuje dwa modele przestrzeni adresowej:

- MGAS-Najdroższa przestrzeń adresowa globalnego (jest to wartość domyślna i jest używana przez produkt WebSphere MQ)
- MPAS-Najdroższa przestrzeń adresowa prywatnego

Aplikacje, które łączą się z produktem WebSphere MQ , mogą używać modeli przestrzeni adresowej MGAS lub MPAS. Aplikacje zbudowane przy użyciu modelu MPAS, które łączą się z produktem WebSphere MQ korzystaniem z pamięci współużytkowanej, mogą ponieść niewielki koszt wydajności ze względu na brak efektywności w odwzorowaniu stron pamięci współużytkowanej używanej przez produkt WebSphere MQ w przestrzeń adresową wirtualnego programu MPAS.

Aplikacje w języku COBOL zbudowane przy użyciu programu Micro Focus Server Express używają domyślnie modelu MPAS.

Za pomocą programu **chatr** można sprawdzać i zmieniać model adresowania używany przez program.

Jeśli wystąpią problemy podczas nawiązywania połączenia z produktem WebSphere MQ z 32-bitowych programów MPAS, należy rozważyć użycie modelu adresowania MGAS lub budowanie aplikacji jako 64-bitowej aplikacji MPAS, a nie 32-bitowej aplikacji MPAS.

Więcej szczegółowych informacji na temat modeli przestrzeni adresowej MGAS i MPAS można znaleźć w dokumentacji systemu HP-UX .

Budowanie aplikacji w systemie Linux

Te informacje opisują dodatkowe zadania i zmiany w standardowych zadaniach, które należy wykonać podczas budowania aplikacji WebSphere MQ for Linux do uruchomienia.

Opcje C i C++ są obsługiwane. Informacje na temat przygotowywania programów w języku C + + zawiera sekcja Używanie języka C++.

Przygotowywanie programów w języku C w programie Linux

Wstępnie skompilowane programy w języku C są dostarczane w katalogu *MQ_INSTALLATION_PATH*/samp/bin . Aby zbudować przykład z kodu źródłowego, należy użyć kompilatora gcc .

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

Praca w normalnym środowisku. Więcej informacji na temat programowania 64-bitowych aplikacji można znaleźć w sekcji Standardy kodowania na platformach 64-bitowych.

Łączenie bibliotek

W poniższych tabelach przedstawiono biblioteki, które są wymagane podczas przygotowywania programów w języku C w systemie Linux.

- Konieczne jest połączenie programów z odpowiednią biblioteką udostępnianej przez produkt WebSphere MQ.

  W środowisku innym niż wielowątkowe należy utworzyć odsyłacz do jednej z następujących bibliotek:

Zbiór biblioteki	Typ programu/wyjścia
libmqm.so	Serwer dla C
libmqic.so & libmqm.so	Klient dla C

  W środowisku wielowątkowym należy połączyć się z jedną z następujących bibliotek:

Zbiór biblioteki	Typ programu/wyjścia
libmqm_r.so	Serwer dla C
libmqic_r.so & libmqm_r.so	Klient dla C

  Uwaga:

  1. Jeśli użytkownik zapisuje usługę instalowalną (więcej informacji na ten temat zawiera sekcja Administrowanie), należy utworzyć odsyłacz do biblioteki produktu libmqmzf.so .
  2. W przypadku tworzenia aplikacji na potrzeby zewnętrznej koordynacji za pomocą menedżera transakcji zgodnego z interfejsem XA, takiego jak IBM TXSeries Encinalub BEA Tuxedo, należy połączyć się z libmqmxa.so (lub libmqmxa64.so , jeśli menedżer transakcji traktuje typ long jako 64-bitowe) i biblioteki produktu libmqz.so w aplikacji niewielowątkowej oraz w bibliotekach libmqmxa_r.so (lub libmqmxa64_r.so) i libmqz_r.so w aplikacji wielowątkowej.
  3. Przed innymi bibliotekami produktu należy połączyć biblioteki produktu WebSphere MQ .

Budowanie aplikacji 31-bitowych

Ten temat zawiera przykłady komend używanych do budowania 31-bitowych programów w różnych środowiskach.

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

Aplikacja kliencka C, 31-bitowa, niewielowątkowa

```
gcc -m31  -o famqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

Aplikacja kliencka C, 31-bitowa, wielowątkowa

```
gcc -m31  -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

Aplikacja serwera C, 31-bitowa, niewielowątkowa

```
gcc -m31  -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

Aplikacja serwera C, 31-bitowa, wielowątkowa

```
gcc -m31  -o amqsput_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Aplikacja kliencka C + +, 31-bitowa, niewielowątkowa

```
g++ -m31 -fsigned-char  -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl
-limqb23gl -lmqic
```

Aplikacja kliencka C++, 31-bitowa, wielowątkowa

```
g++ -m31 -fsigned-char  -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r
-limqb23gl_r -lmqic_r -lpthread
```

Aplikacja serwera C + +, 31-bitowa, niewielowątkowa

```
g++ -m31 -fsigned-char  -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl
-limqb23gl -lmqm
```

Aplikacja serwera C++, 31-bitowa, wielowątkowa

```
g++ -m31 -fsigned-char  -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r
-limqb23gl_r -lmqm_r -lpthread
```

Wyjście klienta C, 31-bitowy, Niewątkowy

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic
```

Wyjście klienta C, 31-bitowa, wielowątkowa

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/cliexit_32_r cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic_r -lpthread
```

Wyjście serwera C, 31-bitowy, niewątkowy

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/srvexit_32 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqm
```

Wyjście serwera C, 31-bitowa, wielowątkowa

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/srvexit_32_r srvexit.c
```

```
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqm_r -lpthread
```

Budowanie aplikacji 32-bitowych

Ten temat zawiera przykłady komend używanych do budowania 32-bitowych programów w różnych środowiskach.

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

Aplikacja kliencka C, 32-bitowa, niewielowątkowa

```
gcc -m32 -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

Aplikacja kliencka C, 32-bitowa, wielowątkowa

```
gcc -m32 -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

Aplikacja serwera C, 32-bitowa, niewielowątkowa

```
gcc -m32 -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

Aplikacja serwera C, 32-bitowa, wielowątkowa

```
gcc -m32 -o amqsput_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Aplikacja kliencka C + +, 32-bitowa, nie wielowątkowa

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib  -Wl,-rpath=/usr/lib
-limqc23gl -limqb23gl -lmqic
```

aplikacja kliencka C + +, 32-bitowa, wielowątkowa

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib  -Wl,-rpath=/usr/lib
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Aplikacja serwera C + +, 32-bitowa, nie wielowątkowa

```
g++ -m32 -fsigned-char -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl -limqb23gl -lmqm
```

Aplikacja serwera C + +, 32-bitowa, wielowątkowa

```
g++ -m32 -fsigned-char -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

Wyjście klienta C, 32-bitowe, bez wątków

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic
```

Wyjście klienta C, 32-bitowe, wielowątkowe

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/cliexit_32_r cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic_r -lpthread
```

Wyjście serwera C, 32-bitowe, nie są gwintowane

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/srvexit_32 srvexit.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

Wyjście serwera C, 32-bitowe, wielowątkowe

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/srvexit_32_r srvexit.c
IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
lmqm_r -lpthread
```

Budowanie aplikacji 64-bitowych

Ten temat zawiera przykłady komend używanych do tworzenia programów 64-bitowych w różnych środowiskach.

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

Aplikacja kliencka C, 64-bitowa, niewielowątkowa

```
gcc -m64  -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqic
```

Aplikacja kliencka C, 64-bitowa, wielowątkowa

```
gcc -m64  -o amqsputc_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqic_r
-lpthread
```

Aplikacja serwera C, 64-bitowa, nie wielowątkowa

```
gcc -m64  -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm
```

Aplikacja serwera C, 64-bitowa, wielowątkowa

```
gcc -m64  -o amqsput_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm_r
-lpthread
```

Aplikacja kliencka C + +, 64-bitowa, nie wielowątkowa

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Aplikacja kliencka C++, 64-bitowa, wielowątkowa

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Aplikacja serwera C + +, 64-bitowa, nie jest wielowątkowa

```
g++ -m64 -fsigned-char  -o imqsput_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -limqs23gl -limqb23gl -lmqm
```

Aplikacja serwera C + +, 64-bitowa, wielowątkowa

```
g++ -m64 -fsigned-char  -o imqsput_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

Wyjście klienta C, 64-bitowe, niewielowątkowe

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/cliexit_64 cliexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqic
```

Wyjście klienta C, 64-bitowe, wielowątkowe

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/cliexit_64_r cliexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqic_r -lpthread
```

Wyjście serwera C, 64-bitowe, niewielowątkowe

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/srvexit_64 srvexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqm
```

Wyjście serwera C, 64-bitowe, wielowątkowe

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/srvexit_64_r srvexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqm_r -lpthread
```

Przygotowywanie programów w języku COBOL w produkcie Linux

Sekcja zawiera informacje na temat przygotowywania programów w języku COBOL w programie Linux i przygotowywania programów w języku COBOL za pomocą programu Micro Focus COBOL

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

1. 32-bitowe podręczniki do kopiowania w języku COBOL są instalowane w następującym katalogu:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

i dowiązania symboliczne są tworzone w:

```
MQ_INSTALLATION_PATH/inc
```

2. Na platformach 64-bitowych w następującym katalogu instalowane są 64-bitowe podręczniki w języku COBOL:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

3. W poniższych przykładach ustaw COBCPY na:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

dla aplikacji 32-bitowych, oraz:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

dla aplikacji 64-bitowych.

Należy połączyć program z jednym z następujących elementów:

Zbiór biblioteki	Typ programu/wyjścia
libmqmcb.so	Serwer dla języka COBOL
libmqicb.so	Klient dla języka COBOL
libmqmcb_r.so	Serwer dla języka COBOL (aplikacja wielowątkowa)
libmqicb_r.so	Klient dla języka COBOL (aplikacja wielowątkowa)

Przygotowywanie programów w języku COBOL przy użyciu Micro Focus COBOL

Ustaw zmienne środowiskowe przed kompilacją programu w następujący sposób:

```
export COBCPY=<COBCPY>
export LIB=MQ_INSTALLATION_PATHlib:$LIB
```

Aby skompilować 32-bitowy program w języku COBOL, w którym jest to obsługiwane, używając Micro Focus COBOL, wpisz:

```
$ cob32 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb Server for COBOL
$ cob32 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb Client for COBOL
$ cob32 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r Threaded Server for COBOL
$ cob32 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r Threaded Client for COBOL
```

Aby skompilować 64-bitowy program w języku COBOL przy użyciu programu Micro Focus COBOL, wpisz:

```
$ cob64 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb Server for COBOL
$ cob64 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb Client for COBOL
$ cob64 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r Threaded Server for COBOL
$ cob64 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r Threaded Client for COBOL
```

gdzie `amqsput` jest programem przykładowym

Opis zmiennych środowiskowych, które są potrzebne, można znaleźć w dokumentacji produktu Micro Focus COBOL.

Budowanie aplikacji w systemie Solaris

W tej sekcji opisano dodatkowe zadania oraz zmiany w standardowych zadaniach, które należy wykonać podczas budowania aplikacji WebSphere MQ dla systemu Solaris w systemie Solaris.

Obsługiwane są języki programowania COBOL, C i C++. Informacje na temat przygotowywania programów w języku C + + zawiera sekcja Używanie języka C++.

Oprócz kodowania wywołań MQI w kodzie źródłowym konieczne jest dodanie odpowiednich plików włączanych. Zaznajomić się z zawartością tych plików. Pełny opis znajduje się w sekcji “Pliki definicji danych produktu IBM WebSphere MQ” na stronie 82 .

W tym temacie znak ukośnika odwrotnego (\) jest używany do dzielenia długich komend na więcej niż jedną linię. Nie wprowadzaj tego znaku, wpisz każdą komendę jako pojedynczą linię.

Przygotowywanie programów w języku C w systemie Solaris

Wstępnie skompilowane programy w języku C są dostarczane w katalogu *MQ_INSTALLATION_PATH*/samp/bin .

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

Więcej informacji na temat programowania 64-bitowych aplikacji można znaleźć w sekcji Standardy kodowania na platformach 64-bitowych.

Aby użyć programów na komputerze, na którym zainstalowano tylko klienta MQI produktu WebSphere MQ dla systemu Solaris, należy skompilować te programy, aby dowiązali je do biblioteki klienta (`-lmqic`).

Jeśli używany jest nieobsługiwany kompilator `?usr?ucb?cc`, aplikacja może pomyślnie skompilować i połączyć się pomyślnie. Jednak po uruchomieniu aplikacji kończy się ona niepowodzeniem podczas próby nawiązania połączenia z menedżerem kolejek.

Uwaga: Działanie 32-bitowych klientów SSL i TLS systemu Solaris x86 skonfigurowanych na potrzeby operacji zgodnych ze standardem FIPS 140-2 zakończy się niepowodzeniem w przypadku uruchomienia w systemie z procesorem Intel. To niepowodzenie występuje, ponieważ 32-bitowy plik biblioteki GSKit-Crypto systemu Solaris x86 zgodny ze standardem FIPS 140-2 nie jest ładowany w układzie Intel. W systemach, których to dotyczy, w dzienniku błędów klienta zgłaszany jest błąd AMQ9655. Aby rozwiązać ten problem, należy wyłączyć zgodność ze standardem FIPS 140-2 lub ponownie skompilować aplikację kliencką w formacie 64-bitowym, ponieważ problem ten nie dotyczy kodu 64-bitowego.

Łączenie bibliotek

Należy połączyć się z bibliotekami WebSphere MQ , które są odpowiednie dla danego typu aplikacji:

Pliki w komponencie Biblioteka	Typ programu/wyjścia
libmqm.so	Serwer dla C
libmqic.so & libmqm.so	Klient dla C

Uwaga:

1. Jeśli zapisujesz usługę instalowalną (więcej informacji na ten temat zawiera Administrowanie), należy przejść do biblioteki `libmqmzf.so` .
2. W przypadku tworzenia aplikacji na potrzeby zewnętrznej koordynacji za pomocą menedżera transakcji zgodnego z interfejsem XA, takiego jak IBM TXSeries Encinalub BEA Tuxedo, należy połączyć się z `libmqmxa.so` (lub `libmqmxa64.so` , jeśli menedżer transakcji traktuje typ long jako 64-bitowe) i biblioteki produktu `libmqz.so` .
3. Przed innymi bibliotekami produktu należy połączyć biblioteki produktu WebSphere MQ .

Budowanie aplikacji na platformie x86-64

Ten temat zawiera przykłady komend używanych do budowania programów w różnych środowiskach na platformie x86-64 .

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

Aplikacja kliencka C, 32-bitowa

```
cc -xarch=386 -mt -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqic -lsocket -lnsl -ldl
```

Aplikacja kliencka C, 64-bitowe

```
cc -xarch=amd64 -mt -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqic -lsocket
-lnsl -ldl
```

Aplikacja serwera C, 32-bitowa

```
cc -xarch=386 -mt -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqm -lsocket -lnsl -ldl
```

Aplikacja serwera C, 64-bitowe

```
cc -xarch=amd64 -mt -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm -lsocket
-lnsl -ldl
```

Aplikacja kliencka C + +, 32-bitowa

```
CC -xarch=386 -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as -lmqic -lsocket -lnsl -ldl
```

Aplikacja kliencka C++ (64-bitowe)

```
CC -xarch=amd64 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Aplikacja serwera C + +, 32-bitowa

```
CC -xarch=386 -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as -lmqm
-lsocket -lnsl -ldl
```

Aplikacja serwera C + +, 64-bitowa

```
CC -xarch=amd64 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as -lmqm
-lsocket -lnsl -ldl
```

Wyjście klienta C, 32-bitowe

```
cc -xarch=386 -mt -G -KPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
```

```
-R/usr/lib/32 -lmqic
-lsocket -lnsl -ldl
```

Wyjście klienta C, 64-bitowe

```
cc -xarch=amd64 -mt -G -KPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqic
-lsocket -lnsl -ldl
```

Wyjście serwera C, 32-bitowe

```
cc -xarch=386 -mt -G -KPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqm
-lsocket -lnsl -ldl
```

Wyjście serwera C, 64-bitowe

```
cc -xarch=amd64 -mt -G -KPIC -o /var/mqm/exits64/srvexit_64 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqm
-lsocket -lnsl -ldl
```

Budowanie aplikacji na platformie SPARC

Ten temat zawiera przykłady komend używanych do budowania programów w różnych środowiskach na platformie SPARC.

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

Aplikacja kliencka C, 32-bitowa

```
cc -xarch=v8plus -mt -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqic -lsocket -lnsl -ldl
```

Aplikacja kliencka C, 64-bitowe

```
cc -xarch=v9 -mt -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqic
-lsocket -lnsl -ldl
```

Aplikacja serwera C, 32-bitowa

```
cc -xarch=v8plus -mt -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqm -lsocket -lnsl -ldl
```

Aplikacja serwera C, 64-bitowe

```
cc -xarch=v9 -mt -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm
-lsocket -lnsl -ldl
```

Aplikacja kliencka C + +, 32-bitowa

```
CC -xarch=v8plus -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic
-lsocket -lnsl -ldl
```

Aplikacja kliencka C++ (64-bitowe)

```
CC -xarch=v9 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Aplikacja serwera C + +, 32-bitowa

```
CC -xarch=v8plus -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as -lmqm
-lsocket -lnsl -ldl
```

Aplikacja serwera C + +, 64-bitowa

```
CC -xarch=v9 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as -lmqm
-lsocket -lnsl -ldl
```

Wyjście klienta C, 32-bitowe

```
cc -xarch=v8plus -mt -G -KPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqic
-lsocket -lnsl -ldl
```

Wyjście klienta C, 64-bitowe

```
cc -xarch=v9 -mt -G -KPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqic
-lsocket -lnsl -ldl
```

Wyjście serwera C, 32-bitowe

```
cc -xarch=v8plus -mt -G -KPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqm
-lsocket -lnsl -ldl
```

Wyjście serwera C, 64-bitowe

```
cc -xarch=v9 -mt -G -KPIC -o /var/mqm/exits64/srvexit_64 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqm
-lsocket -lnsl -ldl
```

Przygotowywanie programów w języku COBOL w systemie Solaris

Informacje na temat przygotowywania programów w języku COBOL w systemie Solaris.

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

1. 32-bitowe podręczniki do kopiowania w języku COBOL są instalowane w następującym katalogu:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   i dowiązania symboliczne są tworzone w:

```
MQ_INSTALLATION_PATH/inc
```

2. 64-bitowe podręczniki do kopiowania w języku COBOL są instalowane w następującym katalogu:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

3. W poniższych przykładach ustaw COBCPY na:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

dla aplikacji 32-bitowych, oraz:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

dla aplikacji 64-bitowych.

Skompiluj programy za pomocą kompilatora Micro Focus. Pliki kopii, które deklarują struktury, znajdują się w programie *MQ_INSTALLATION_PATH*/inc:

```
$ export LIB=MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="<COBCPY>"
```

Kompilowanie 32-bitowych programów:

- `$ cob32 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb`

  Serwer dla języka COBOL
- `$ cob32 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb`

  Klient dla języka COBOL
- `$ cob32 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r`

  Serwer wątków dla języka COBOL
- `$ cob32 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r`

  Klient wielowątkowy dla COBOL

Kompilowanie programów 64-bitowych:

- `$ cob64 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb`

  Serwer dla języka COBOL
- `$ cob64 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb`

  Klient dla języka COBOL
- `$ cob64 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r`

  Serwer wątków dla języka COBOL
- `$ cob64 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r`

  Klient wielowątkowy dla COBOL

gdzie *amqs0put0.cbl* jest programem przykładowym.

Należy połączyć program z jednym z następujących elementów:

- libmqmcb.so

  Serwer dla języka COBOL
- libmqicb.so

  Klient dla języka COBOL

Przygotowywanie programów CICS w systemie Solaris

Informacje na temat przygotowywania programów CICS w systemie Solaris.

Dostępny jest moduł przełącznika XA, który umożliwia połączenie programu CICS z produktem WebSphere MQ:

Tabela 67. Podstawowy kod dla aplikacji CICS (Solaris)		
Opis	**C (źródło)**	**C (exec)**
Procedura inicjowania interfejsu XA	amqzscix.c	amqzsc- TXSeries dla systemu Solaris

Zawsze należy połączyć transakcje z wątkową bezpieczną biblioteką WebSphere MQ libmqm.so.

Więcej informacji na temat obsługi transakcji CICS można znaleźć w Administrowanie.

Obsługa programu TXSeries CICS

Produkt WebSphere MQ for Solaris obsługuje program TXSeries CICS przy użyciu interfejsu XA.

Napisz programy WebSphere MQ , które są ładowane do tego samego regionu CICS w języku C lub COBOL. Nie można utworzyć połączenia wywołań interfejsu MQI języka C i języka COBOL w tym samym regionie CICS . Większość wywołań MQI w drugim języku nie powiodła się z kodem przyczyny MQRC_H0BJ_ERR0R.

Przygotowywanie programów w języku COBOL programu CICS przy użyciu programu Micro Focus COBOL

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

Aby użyć programu Micro Focus COBOL, należy wykonać następujące czynności:

1. Dodaj moduł biblioteki środowiska wykonawczego języka COBOL produktu WebSphere MQ do biblioteki środowiska wykonawczego za pomocą następującej komendy:

   ```
   cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
               MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe
   ```

   Uwaga: W produkcie `cicsmkcobol`produkt WebSphere MQ nie zezwala na wywozy wywołań MQI w języku programowania C z poziomu aplikacji COBOL.

   Jeśli istniejące aplikacje mają takie wywołania, należy przenieść te funkcje z aplikacji w języku COBOL do własnej biblioteki, na przykład `myMQ.so`. Po przeniesieniu tych funkcji nie należy uwzględniać biblioteki produktu WebSphere MQ `libmqmcbrt.o` podczas budowania aplikacji w języku COBOL dla programu CICS.

   Ponadto, jeśli aplikacja COBOL nie wykonuje żadnego wywołania MQI COBOL, nie należy łączyć `libmqmz_r` z `cicsmkcobol`.

   Spowoduje to utworzenie pliku metody języka Micro Focus COBOL i włączenie biblioteki COBOL środowiska wykonawczego CICS w celu wywołania produktu WebSphere MQ w systemach UNIX and Linux .

   Uwaga: Uruchom program `cicsmkcobol` tylko wtedy, gdy instalowany jest jeden z następujących produktów:

   - Nowa wersja lub wydanie Micro Focus COBOL
   - Nowa wersja lub wydanie TXSeries dla systemu Solaris
   - Nowa wersja lub wydanie dowolnego obsługiwanego produktu bazodanowego (tylko dla transakcji w języku COBOL)

- Nowa wersja lub wydanie produktu WebSphere MQ

2. Wyeksportuj następującą zmienną środowiskową:

```
COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
```

3. Konwertuj, skompiluj i dowiązaj program wpisując:

```
cicstcl -l COBOL -e <yourprog>.ccp
```

Przygotowywanie programów CICS C

Budowanie programów CICS C przy użyciu standardowych narzędzi CICS :

1. Wyeksportuj ***jeden*** z następujących zmiennych środowiskowych:
   - LDFLAGS = "-L*MQ_INSTALLATION_PATH*Biblioteka główna -lmqm_r" export LDFLAGS
   - USERLIB = "-L*MQ_INSTALLATION_PATH*biblioteka katalogowa -lmqm_r" export USERLIB
2. Konwertuj, skompiluj i dowiązaj program wpisując:

```
cicstcl -l C amqscic0.ccs
```

Przykładowa transakcja CICS C

Sample C source for a CICS WebSphere MQ transaction is provided by AMQSCIC0.CCS. Transakcja odczytuje komunikaty z kolejki transmisji SYSTEM.SAMPLE.CICS.WORKQUEUE w domyślnym menedżerze kolejek i umieszcza je w kolejce lokalnej z nazwą kolejki, która jest zawarta w nagłówku transmisji komunikatu. Wszystkie niepowodzenia są wysyłane do kolejki SYSTEM.SAMPLE.CICS.DLQ. Użyj przykładowego skryptu MQSC AMQSCIC0.TST , aby utworzyć te kolejki i przykładowe kolejki wejściowe.

Budowanie aplikacji w systemach Windows

W publikacjach systemów Windows opisano sposób budowania aplikacji wykonywalnych z programów, które są zapisywane.

W tym temacie opisano dodatkowe zadania oraz zmiany w standardowych zadaniach, które należy wykonać podczas budowania aplikacji WebSphere MQ for Windows w celu uruchamiania w systemach Windows . Obsługiwane są następujące języki programowania: ActiveX, C, C + +, COBOL i Visual Basic. Informacje na temat przygotowywania programów ActiveX zawiera sekcja Używanie interfejsu Component Object Model Interface (WebSphere MQ Automation Classes for ActiveX). Informacje na temat przygotowywania programów w języku C + + zawiera sekcja Używanie języka C++.

Zadania, które należy wykonać, aby utworzyć aplikację wykonywalną przy użyciu produktu WebSphere MQ for Windows , różnią się w zależności od języka programowania, w którym kod źródłowy jest zapisywany. Oprócz kodowania wywołań MQI w kodzie źródłowym należy dodać odpowiednie instrukcje języka w celu uwzględnienia plików włączanych w produkcie WebSphere MQ for Windows dla używanego języka. Zaznajomić się z zawartością tych plików. Pełny opis znajduje się w sekcji “Pliki definicji danych produktu IBM WebSphere MQ” na stronie 82 .

Budowanie aplikacji 64-bitowych w systemie Windows

W produkcie IBM WebSphere MQ for Windows Version 7.5obsługiwane są aplikacje 32-i 64-bitowe. Pliki wykonywalne i pliki biblioteki produktu IBM WebSphere MQ są dostarczane zarówno w postaci 32-bitowej, jak i 64-bitowej, w zależności od aplikacji, z którą pracuje użytkownik.

Pliki wykonywalne i biblioteki

Zarówno 32-bitowe, jak i 64-bitowe wersje bibliotek produktu IBM WebSphere MQ są dostarczane w następujących miejscach:

Tabela 68. Położenie bibliotek produktu IBM WebSphere MQ	
Wersja biblioteki	**Katalog zawierający pliki bibliotek**
32 bity	*MQ_INSTALLATION_PATH*\Tools\Lib
64-bitowe	*MQ_INSTALLATION_PATH*\Tools\Lib64

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

Aplikacje 32-bitowe nadal działają normalnie po migracji. Pliki 32-bitowe istnieją w tym samym katalogu, co w poprzednich wersjach produktu.

Aby utworzyć wersję 64-bitową, należy upewnić się, że środowisko jest skonfigurowane do korzystania z plików biblioteki w programie *MQ_INSTALLATION_PATH*\Tools\Lib64. Upewnij się, że zmienna środowiskowa LIB nie jest ustawiona do wyszukiwania w folderze, w którym znajdują się biblioteki 32-bitowe.

Przygotowywanie programów w języku C w systemie Windows

Praca w typowym środowisku Windows ; produkt WebSphere MQ for Windows nie wymaga żadnego specjalnego działania.

Więcej informacji na temat programowania aplikacji 64-bitowych zawiera sekcja Standardy kodowania na 64-bitowych platformach.

- Powiąż programy z odpowiednimi bibliotekami udostępnionym przez produkt WebSphere MQ:

Zbiór biblioteki	**Typ programu/wyjścia**
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	serwer dla 32-bitowego C
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	klient dla 32-bitowego C
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	klient dla 32-bitowego C z koordynacją transakcji
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	serwer dla 64-bitowego C
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	klient dla 64-bitowego C
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	klient dla 64-bitowego C z koordynacją transakcji

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

Poniższa komenda przedstawia przykład kompilowania przykładowego programu amqsget0 (za pomocą kompilatora Microsoft Visual C++).

Dla aplikacji 32-bitowych:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib
```

Dla aplikacji 64-bitowych:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib
```

Uwaga:

- Jeśli użytkownik zapisuje usługę instalowalną (więcej informacji na ten temat zawiera sekcja Administrowanie), należy utworzyć odsyłacz do biblioteki mqmzf.lib .
- W przypadku tworzenia aplikacji na potrzeby zewnętrznej koordynacji za pomocą menedżera transakcji zgodnego z interfejsem XA, takiego jak IBM TXSeries Encina, lub BEA Tuxedo, należy utworzyć dowiązanie do biblioteki mqmxa.lib lub mqmxa.lib .
- W przypadku zapisywania programu zewnętrznego programu CICS należy utworzyć odsyłacz do biblioteki mqmcics4.lib .
- Przed innymi bibliotekami produktu należy połączyć biblioteki produktu WebSphere MQ .

- Biblioteki DLL muszą znajdować się w podanej ścieżce (PATH).
- Jeśli w miarę możliwości używane są małe litery, można przejść z produktu WebSphere MQ for Windows do produktu WebSphere MQ w systemach UNIX and Linux , w których konieczne jest użycie małych liter.

Przygotowywanie programów CICS i Transaction Server

Sample C source for a CICS WebSphere MQ transaction is provided by AMQSCIC0.CCS. Budujesz go przy użyciu standardowych narzędzi CICS . Na przykład dla TXSeries dla Windows 2000:

1. Ustaw zmienną środowiskową (wpisz następujący kod w jednym wierszu):

   ```
   set CICS_IBMC_FLAGS=-IMQ_INSTALLATION_PATH\Tools\C\Include;
   %CICS_IBMC_FLAGS%
   ```

2. Ustaw zmienną środowiskową USERLIB:

   ```
   set USERLIB=MQM.LIB;%USERLIB%
   ```

3. Przetłumacz, skompiluj i dowiązaj przykładowy program:

   ```
   cicstcl -l IBMC amqscic0.ccs
   ```

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

Jest to opisane w publikacji *Transaction Server for Windows NT Application Programming Guide (CICS) V4*.

Więcej informacji na temat obsługi transakcji CICS można znaleźć w Administrowanie.

Przygotowywanie programów w języku COBOL w produkcie Windows

Ta sekcja zawiera informacje na temat przygotowywania programów w języku COBOL w produkcie Windowsoraz przygotowywania programów CICS i serwera transakcji.

1. 32-bitowe podręczniki w języku COBOL są instalowane w następującym katalogu: *MQ_INSTALLATION_PATH*\Tools\cobol\CopyBook.

2. 64-bitowe podręczniki dla języka COBOL są instalowane w następującym katalogu: *MQ_INSTALLATION_PATH*\Tools\cobol\CopyBook64
3. W poniższych przykładach ustaw CopyBook na:

```
CopyBook
```

   dla aplikacji 32-bitowych, oraz:

```
CopyBook64
```

   dla aplikacji 64-bitowych.

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

Aby przygotować programy w języku COBOL w systemach Windows , należy połączyć program z jedną z następujących bibliotek udostępnianych przez produkt IBM WebSphere MQ:

Zbiór biblioteki	Typ programu lub wyjścia
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb	32-bitowy serwer dla IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb	32-bitowy serwer dla Micro Focus COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb	Klient 32-bitowy dla produktu IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb	32-bitowy klient dla Micro Focus COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb	Serwer 64-bitowy produktu IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb	64-bitowy serwer dla Micro Focus COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb	Klient 64-bitowy produktu IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb	Klient 64-bitowy dla Micro Focus COBOL

W przypadku uruchamiania programu w środowisku klienta MQI upewnij się, że biblioteka DOSCALLS jest wyświetlana przed dowolną biblioteką języka COBOL lub IBM WebSphere MQ .

Kompilator IBM COBOL Set lub kompilator Micro Focus COBOL można używać w zależności od programu:

- Programy rozpoczynające się od amqi są odpowiednie dla kompilatora IBM COBOL Set,
- Programy rozpoczynające się od amqm są odpowiednie dla kompilatora Micro Focus COBOL, oraz
- Programy rozpoczynające się od amq0 są odpowiednie dla każdego kompilatora.

IBM i Micro Focus COBOL

Zreluj wszystkie istniejące 32-bitowe programy IBM WebSphere MQ Micro Focus COBOL za pomocą mqmcb.lib lub mqiccb.lib, a nie bibliotek mqmcbb i mqiccbb .

Aby skompilować, na przykład, przykładowy program amq0put0, używając programu IBM VisualAge COBOL:

1. Ustaw zmienną środowiskową SYSLIB tak, aby zawierała ścieżkę do struktury copybook języka COBOL produktu IBM WebSphere MQ VisualAge (wprowadź następujący kod w jednym wierszu):

```
set SYSLIB=MQ_INSTALLATION_PATH\
Tools\Cobol\Copybook\VAcobol;%SYSLIB%
```

2. Do użycia na serwerze IBM WebSphere MQ :

```
cob2 amq0put0.cbl -qlib "MQ_INSTALLATION_PATH\
Tools\Lib\mqmcbb.lib"
```

3. Do użycia na kliencie IBM WebSphere MQ :

```
cob2 amq0put0.cbl -qlib "MQ_INSTALLATION_PATH\
Tools\Lib\mqiccbb.lib"
```

Uwaga: Mimo że należy użyć opcji kompilatora `CALLINT (SYSTEM)`, jest to ustawienie domyślne dla produktu `cob2`.

Aby skompilować, na przykład, przykładowy program `amq0put0`, używając Micro Focus COBOL:

1. Ustaw zmienną środowiskową COBCPY w taki sposób, aby wskazywała na struktury copybook języka COBOL produktu IBM WebSphere MQ (wprowadź następujący kod w jednym wierszu):

```
set COBCPY=MQ_INSTALLATION_PATH\
Tools\Cobol\Copybook
```

2. Skompiluj program, aby nadać mu plik wynikowy:

```
cobol amq0put0 LITLINK
```

3. Dowiąże plik wynikowy do systemu wykonawczego.
   - Ustaw zmienną środowiskową LIB tak, aby wskazywała na biblioteki COBOL kompilatora.
   - Odsyłacz do pliku obiektu, który ma być używany na serwerze IBM WebSphere MQ :

     ```
     cbllink amq0put0.obj mqmcb.lib
     ```

   - Lub dowiązaj plik obiektu do użycia na kliencie IBM WebSphere MQ :

     ```
     cbllink amq0put0.obj mqiccb.lib
     ```

Przygotowywanie programów CICS i serwera transakcji

Aby skompilować i połączyć program TXSeries for Windows NT, program V5.1 za pomocą programu IBM VisualAge COBOL:

1. Ustaw zmienną środowiskową (wpisz następujący kod w jednym wierszu):

```
set CICS_IBMCOB_FLAGS=MQ_INSTALLATION_PATH\
Cobol\Copybook\VAcobol;%CICS_IBMCOB_FLAGS%
```

2. Ustaw zmienną środowiskową USERLIB:

```
set USERLIB=MQMCBB.LIB
```

3. Konwertuj, skompiluj i dowiązaj swój program:

```
cicstcl -l IBMCOB myprog.ccp
```

Jest to opisane w podręczniku *Transaction Server for Windows NT, V4 Application Programming* Guide.

Aby skompilować i połączyć program CICS dla programu Windows V5 za pomocą programu Micro Focus COBOL:

- Ustaw zmienną INCLUDE:

```
set
INCLUDE=<drive>:\<programname>\ibm\websphere\tools\c\include;
        <drive>:\opt\cics\include;%INCLUDE%
```

- Ustaw zmienną środowiskową COBCPY:

```
 setCOBCPY=<drive>:\<programname>\ibm\websphere\tools\cobol\copybook;
           <drive>:\opt\cics\include
```

- Ustaw opcje języka COBOL:
  - `set`
  - `COBOPTS=/LITLINK /NOTRUNC`

i uruchom następujący kod:

```
cicstran cicsmq00.ccp
cobol cicsmq00.cbl /LITLINK /NOTRUNC
cbllink -D -Mcicsmq00 -Ocicsmq00.cbmfnt cicsmq00.obj
%CICSLIB%\cicsprCBMFNT.lib user32.lib msvcrt.lib kernel32.lib mqmcb.lib
```

Przygotowywanie programów Visual Basic w systemie Windows

Tych informacji należy używać podczas rozważania korzystania z programów Visual Basic w systemie Windows.

Uwaga: Wersje 64-bitowe plików modułu Visual Basic nie są dostarczane.

Aby przygotować programy Visual Basic w systemie Windows:

1. Utwórz nowy projekt.
2. Dodaj dostarczony plik modułu CMQB.BAS, do projektu.
3. Dodaj inne dostarczone pliki modułów, jeśli są one potrzebne:

CMQBB.BAS	Obsługa interfejsu MQAI
CMQCFB.BAS	Obsługa PCF
CMQXB.BAS	Obsługa wyjść kanału
CMQPSB.BAS	Publikowanie/subskrypcja

Sekcja “Kodowanie w języku Visual Basic” na stronie 89 zawiera informacje na temat używania wywołania MQCONNXAny z poziomu języka Visual Basic.

Wywołaj procedurę MQ_SETDEFAULTS przed wywołaniem dowolnego wywołania MQI w kodzie projektu. Ta procedura służy do konfigurowania struktur domyślnych wymaganych przez wywołania MQI.

Przed skompilowaniem lub uruchomieniem projektu określ, czy tworzony jest serwer lub klient WebSphere MQ , poprzez ustawienie warunkowej zmiennej kompilacji *MqType*. Ustaw *MqType* w projekcie Visual Basic na 1 dla serwera lub 2 dla klienta w następujący sposób:

1. Wybierz menu Projekt.
2. Wybierz opcję *Name* Właściwości (gdzie *Name* jest nazwą bieżącego projektu).
3. W oknie dialogowym wybierz kartę Make.
4. W polu Argumenty kompilacji warunkowej wprowadź tę wartość dla serwera:

   ```
   MqType=1
   ```

   lub w przypadku klienta:

```
MqType=2
```

Wyjście zabezpieczeń SSPI

Produkt WebSphere MQ for Windows udostępnia wyjście zabezpieczeń zarówno dla klienta MQI produktu WebSphere MQ , jak i dla serwera WebSphere MQ . Jest to program obsługi wyjścia kanału, który udostępnia uwierzytelnianie dla kanałów produktu WebSphere MQ przy użyciu interfejsu SSPI (Security Services Programming Interface). Interfejs SSPI udostępnia zintegrowane zabezpieczenia systemów Windows .

Pakiety zabezpieczeń są ładowane z pliku security.dll lub z pliku secur32.dll. Te biblioteki DLL są dostarczane wraz z systemem operacyjnym.

Uwierzytelnianie jednokierunkowe jest udostępniane za pomocą usług uwierzytelniania NTLM. Uwierzytelnianie dwukierunkowe jest udostępniane za pomocą usług uwierzytelniania Kerberos .

Program obsługi wyjścia zabezpieczeń jest dostarczany w formacie źródłowym i obiektowym. Można użyć kodu wynikowego, który jest, lub użyć kodu źródłowego jako punktu wyjścia do tworzenia własnych programów obsługi wyjścia użytkownika.

Patrz także “Korzystanie z wyjścia zabezpieczeń SSPI w systemach Windows” na stronie 173.

Wprowadzenie do wyjść bezpieczeństwa

Wyjście zabezpieczeń tworzy bezpieczne połączenie między dwoma programami z wyjściami zabezpieczeń, z których jeden jest przeznaczony dla wysyłającego agenta kanału komunikatów (Message channel agent - MCA), a drugi dla odbierającego agenta MCA.

Program, który inicjuje bezpieczne połączenie, czyli pierwszy program do kontroli po nawiązaniu sesji MCA, jest znany jako *inicjator kontekstu*. Program partnerski jest znany jako *akceptor kontekstu*.

W poniższej tabeli przedstawiono niektóre typy kanałów, które są inicjatorami kontekstu i powiązanymi z nimi akceptorami kontekstu.

Tabela 69. Inicjatory kontekstu i powiązane z nimi akceptory kontekstu

Inicjator kontekstu	**Akceptor kontekstu**
MQCHT_CLNTCONN	MQCHT_SVRCONN
MQCHT_RECEIVER	MQCHT_SENDER
MQCHT_CLUSRCVR	MQCHT_CLUSSDR

Program obsługi wyjścia zabezpieczeń ma dwa punkty wejścia:

- **SCY_NTLM**

  W ten sposób używane są usługi uwierzytelniania NTLM, które zapewniają uwierzytelnianie w jedną stronę. NTLM umożliwia serwerom sprawdzanie tożsamości swoich klientów. Nie pozwala ona klientom zweryfikować tożsamości serwera lub jednego serwera w celu zweryfikowania tożsamości innego serwera. Uwierzytelnianie NTLM zostało zaprojektowane z myślą o środowisku sieciowym, w którym zakłada się, że serwery są prawdziwe.

- **SCY_KERBEROS**

  Korzysta z usług uwierzytelniania wzajemnego Kerberos . Protokół Kerberos nie zakłada, że serwery w środowisku sieciowym są prawdziwe. Strony na obu końcach połączenia sieciowego mogą weryfikować tożsamość drugiej strony. Oznacza to, że serwery mogą weryfikować tożsamość klientów i innych serwerów, a klienty mogą weryfikować tożsamość serwera.

Co robi wyjście zabezpieczeń

W tej sekcji opisano, co robią programy obsługi wyjścia kanału SSPI.

Dostarczone programy obsługi wyjścia kanału udostępniają jednokierunkowe lub dwukierunkowe (wzajemne) uwierzytelnianie systemu partnerskiego w momencie tworzenia sesji. Dla konkretnego kanału, każdy program obsługi wyjścia ma powiązaną *nazwę użytkownika* (podobną do ID użytkownika, patrz “WebSphere MQ -kontrola dostępu i nazwy użytkowników systemu Windows” na stronie 477). Połączenie między dwoma programami obsługi wyjścia jest powiązaniem między tymi dwoma zleceniodawców.

Po nawiązaniu sesji bazowej zostanie nawiązane bezpieczne połączenie między dwoma programami obsługi wyjścia zabezpieczeń (jedną dla wysyłającego agenta MCA i jedną dla odbierającego agenta MCA). Sekwencja operacji jest następująca:

1. Każdy program jest powiązany z konkretnym zleceniodawca, na przykład w wyniku jawnej operacji logowania.
2. Inicjator kontekstu żąda bezpiecznego połączenia z partnerem z pakietu zabezpieczeń (dla Kerberos, nazwanego partnera) i odbiera token (o nazwie token1). Token jest wysyłany, przy użyciu bazowej sesji, która jest już ustanowiona, do programu partnerskiego.
3. Program partnerski (akceptor kontekstu) przekazuje element token1 do pakietu zabezpieczeń, który sprawdza, czy inicjator kontekstu jest autentyczny. W przypadku NTLM połączenie jest teraz nawiązane.
4. W przypadku wyjścia zabezpieczeń dostarczonego przez protokół Kerberos(w przypadku uwierzytelniania wzajemnego) pakiet zabezpieczeń generuje również drugi znacznik (o nazwie token2), który akceptor kontekstu zwraca do inicjatora kontekstu przy użyciu sesji bazowej.
5. Inicjator kontekstu korzysta z token2 w celu sprawdzenia, czy akceptor kontekstu jest autentyczny.
6. Na tym etapie, jeśli obie aplikacje są zadowolone z autentyczności tokenu partnera, nawiąże się bezpieczne (uwierzytelnione) połączenie.

WebSphere MQ -kontrola dostępu i nazwy użytkowników systemu Windows

Kontrola dostępu, którą udostępnia produkt WebSphere MQ , jest oparta na użytkowniku i grupie. Uwierzytelnianie, które udostępnia system Windows , jest oparte na zasadach, takich jak nazwa użytkownika i nazwa servicePrincipal(SPN). W przypadku nazwy servicePrincipalmoże istnieć wiele z nich powiązanych z jednym użytkownikiem.

Wyjście zabezpieczeń SSPI korzysta z odpowiednich nazw użytkowników systemu Windows w celu uwierzytelnienia. Jeśli uwierzytelnianie w systemie Windows zakończy się pomyślnie, wyjście przekazuje identyfikator użytkownika powiązany z elementem głównym systemu Windows do produktu WebSphere MQ w celu kontroli dostępu.

Nazwy użytkowników systemu Windows , które są istotne dla uwierzytelniania, różnią się w zależności od typu używanego uwierzytelniania.

- Dla uwierzytelniania NTLM, nazwą użytkownika Windows dla inicjatora kontekstu jest ID użytkownika powiązany z uruchomionym procesem. Ponieważ to uwierzytelnianie jest jednym ze sposobów, nazwa użytkownika powiązana z akceptorem kontekstu jest nieistotna.
- W przypadku uwierzytelniania Kerberos , w kanałach CLNTCONN, nazwą użytkownika systemu Windows jest ID użytkownika powiązany z uruchomionym procesem. W przeciwnym razie nazwą użytkownika systemu Windows jest nazwa servicePrincipal, która jest tworzona przez dodanie następującego przedrostka do nazwy QueueManager.

```
ibmMQSeries/
```

Korzystanie z usług protokołu LDAP (Lightweight Directory Access Protocol) z produktem WebSphere MQ for Windows

W tym temacie wyjaśniono, czym jest usługa katalogowa i część odtwarza przez protokół dostępu do katalogów (DAP). Wyjaśniono również, w jaki sposób aplikacje WebSphere MQ mogą korzystać z katalogu LDAP (Lightweight Directory Access Protocol) przy użyciu przykładowego programu jako podręcznika.

Uwaga: Przykładowy program jest przeznaczony dla osób, które znają już katalog LDAP.

Poniższe tematy zawierają więcej informacji na temat usług katalogowych, LDAP i korzystania z LDAP za pomocą produktu WebSphere MQ.

- "Usługa katalogowa" na stronie 478
- "protokół LDAP (Lightweight Directory Access Protocol)" na stronie 478
- "Korzystanie z protokołu LDAP w produkcie WebSphere MQ" na stronie 479

Usługa katalogowa

Katalog jest repozytorium informacji o obiektach, które jest zorganizowane w taki sposób, że łatwo jest znaleźć informacje o konkretnym obiekcie.

Typowym przykładem jest katalog telefoniczny, gdzie informacje (adres i numer telefonu) są przechowywane na temat osób i firm. Innym przykładem jest książka adresowa systemu poczty elektronicznej, w której adresy e-mail oraz opcjonalnie inne informacje, takie jak numery telefonów, są przechowywane dla osób.

W systemach komputerowych katalogi mogą przechowywać informacje o zasobach komputera, takich jak drukarki lub dyski współużytkowane. Na przykład, można użyć katalogu, aby dowiedzieć się, gdzie znajduje się najbliższa drukarka kolorowa. W aplikacji WebSphere MQ można użyć katalogu, aby udostępnić powiązanie między usługą aplikacji (taką jak przetwarzanie należności) i kolejką, która ma być używana dla komunikatów wymagających tej usługi (ewentualnie identyfikowanej za pomocą nazwy kolejki i jej nazwy menedżera kolejek hosta).

Katalogi są implementowane jako systemy typu klient-serwer, w których serwer katalogów przechowuje wszystkie informacje i odpowiada na żądania klientów. Klientami mogą być programy interfejsu użytkownika, które udostępniają informacje bezpośrednio użytkownikowi lub programom aplikacji, które muszą znaleźć zasoby w celu zakończenia ich pracy. Usługa katalogowa składa się z serwera katalogów, programów administracyjnych oraz bibliotek i programów klienta, które są niezbędne do skonfigurowania, zaktualizowania i odczytu katalogu.

protokół LDAP (Lightweight Directory Access Protocol)

Istnieje wiele usług katalogowych, takich jak Novell Directory Services, DCE Cell Directory Service, Banyan StreetTalk, Windows Directory Services, X.500, a także usługi książki adresowej powiązane z produktami e-mail. X.500 został zaproponowany jako standard dla usług katalogowych globalnego wg Międzynarodowej Organizacji Normalizacji (ISO). Wymaga on stosu protokołu OSI do jego komunikacji, a w dużej mierze z tego powodu jego użycie zostało ograniczone do dużych organizacji i instytucji akademickich. Serwer katalogów X.500 komunikuje się z klientami za pomocą protokołu dostępu do katalogów (Directory Access Protocol-DAP).

Protokół LDAP (Lightweight Directory Access Protocol) został utworzony jako uproszczona wersja DAP. Łatwiej jest zaimplementować, pomija niektóre z funkcji DAP, a także działa nad TCP/IP. W wyniku tych zmian szybko jest on przyjmowany jako protokół dostępu do katalogów dla większości celów, zastępując wiele wcześniej używanych protokołów zastrzeżonych. Klienci LDAP nadal mogą uzyskiwać dostęp do serwera X.500 za pośrednictwem bramy (X.500 nadal wymaga stosu protokołu OSI) lub w coraz większym stopniu implementacje X.500 zwykle obejmują obsługę rodzimą dla protokołu LDAP, jak również dostęp DAP.

Katalogi LDAP mogą być dystrybuowane i mogą korzystać z replikacji w celu umożliwienia wydajnego dostępu do ich zawartości.

Bardziej kompletny opis katalogu LDAP znajduje się w publikacji *Informacje o LDAP* IBM Redbooks .

Korzystanie z protokołu LDAP w produkcie WebSphere MQ

W konfiguracjach produktu WebSphere MQ informacje definiujące komunikaty i kolejki transmisji są przechowywane lokalnie. Oznacza to, że w sieci WebSphere MQ dystrybuowane są różne definicje bez centralnego katalogu tych informacji, które są dostępne do przeglądania. Zdalne przesyłanie komunikatów między aplikacjami produktu WebSphere MQ jest często realizowane przy użyciu lokalnych definicji kolejek zdalnych. Aplikacja najpierw wysyła wywołanie MQOPEN, używając nazwy podanej w lokalnej definicji kolejki zdalnej. Aby umieścić komunikat w kolejce zdalnej, aplikacja wysyła komunikat MQPUT, określając uchwyt zwrócony z wywołania MQOPEN. Definicja kolejki zdalnej dostarcza nazwę kolejki docelowej, docelowy menedżer kolejek i opcjonalnie kolejkę transmisji. W tej technice aplikacja musi znać w czasie wykonywania nazwę określoną w definicji kolejki lokalnej.

Odmiana w porównaniu z poprzednimi unikaniem używania lokalnych definicji kolejek zdalnych. Aplikacja może określić pełną nazwę kolejki docelowej, która zawiera nazwę zdalnego menedżera kolejek jako część komendy MQOPEN. W związku z tym aplikacja musi znać te dwie nazwy w czasie wykonywania. Lokalny menedżer kolejek musi być poprawnie skonfigurowany z definicją kolejki lokalnej i z odpowiednio nazwaną (lub domyślną) kolejką transmisji i powiązanym kanałem dostarczanym do systemu docelowego.

W przypadku, gdy zarówno menedżerowie kolejek źródłowych, jak i docelowych są zdefiniowani jako członkowie tego samego klastra, można zignorować tę samą kolejkę transmisji i aspekty kanału w poprzednich dwóch scenariuszach. Jeśli docelowa kolejka transmisji jest kolejką klastra, to lokalna definicja kolejki zdalnej również nie jest wymagana. Jednak podobnie jak w poprzednich opisach, aplikacja musi jeszcze znać nazwę kolejki docelowej.

Usługa katalogowa może być używana do usuwania tej zależności aplikacji w nazwach kolejek (lub w połączeniu z nazwami kolejek i menedżerów kolejek). Odwzorowanie między kryteriami aplikacji i nazwami obiektów WebSphere MQ może odbywać się w katalogu i aktualizowane dynamicznie, a także niezależnie od aplikacji. W czasie wykonywania aplikacja WebSphere MQ , która chce wysłać komunikat po raz pierwszy, wysyła zapytanie do katalogu za pomocą kryteriów opartych na aplikacji, na przykład gdzie: service_name = "accounts receivable", pobiera odpowiednie nazwy obiektów WebSphere MQ , a następnie używa tych zwróconych wartości w wywołaniu MQOPEN.

Innym przykładem korzystania z katalogu jest firma, która ma wiele małych składów lub biur, a klienty MQI produktu WebSphere MQ mogą być używane do wysyłania komunikatów do serwerów WebSphere MQ znajdujących się w większych biurach. Klienty muszą znać nazwę komputera hosta, kanał MQI i nazwę kolejki dla każdego serwera, do którego są wysyłane komunikaty. Czasami konieczne może być przeniesienie serwera WebSphere MQ do innego komputera. Każdy klient, który komunikuje się z serwerem, będzie musiał wiedzieć o zmianie. Usługa katalogowa LDAP może być używana do przechowywania nazw komputerów hosta (oraz nazw kanałów i kolejek), a programy klienckie mogą pobierać informacje z katalogu za każdym razem, gdy chcą wysłać komunikat do serwera. W takim przypadku należy zaktualizować tylko katalog, jeśli zmieniono nazwę hosta (lub nazwę kanału lub kolejki).

Wiele miejsc docelowych dla komunikatu aplikacji może być zapisanych w katalogu, przy czym jeden z nich jest zależny od dostępności lub uwag dotyczących współużytkowania obciążenia.

Produkt WebSphere MQ może również używać katalogu LDAP do przechowywania informacji uwierzytelniających do użycia z protokołem SSL (Secure Sockets Layer). Klasy produktu WebSphere MQ dla języka Java mogą także przechowywać informacje w katalogu LDAP.

Przykładowy program LDAP

Przykładowy program jest przeznaczony dla osób zaznajomionych z LDAP i prawdopodobnie korzysta z niego już. Jego celem jest pokazanie, w jaki sposób aplikacje programu WebSphere MQ mogą korzystać z katalogu LDAP.

Budowanie przykładowego programu

Ten program został zbudowany i przetestowany tylko w systemie Windows za pomocą protokołu TCP/IP. Oprócz ogólnych uwag wymienionych w sekcji "Przygotowywanie programów w języku C w systemie Windows" na stronie 471należy zwrócić uwagę na następujące kwestie:

- Ten program jest przeznaczony do uruchamiania jako program kliencki, dlatego powinien być połączony z programem MQIC.LIB .
- Podobnie jak pliki nagłówkowe i biblioteki produktu WebSphere MQ , program ten musi być zbudowany przy użyciu plików nagłówkowych klienta LDAP i bibliotek.

  Na przykład za pomocą klienta IBM eNetwork należy połączyć program za pomocą komendy LIBLDAPSTATICE.LIB i LIBLBERSTATICSSL.LIB .

Konfigurowanie katalogu

Przed uruchomieniem przykładowego programu należy skonfigurować serwer LDAP Directory Server z przykładowymi danymi.

Plik MQuser.ldif, w katalogu `tools\c\samples` , zawiera przykładowe dane w formacie LDIF (LDAP Data Interchange Format). Ten plik można edytować w celu dostosowania go do własnych potrzeb. Zawiera on dane dla fikcyjnej firmy o nazwie MQuser, która zawiera dział transportu składający się z trzech biur. Każdy z tych biur ma komputer, na którym działa serwer WebSphere MQ .

Należy co najmniej edytować trzy wiersze zawierające nazwy hostów komputerów z serwerami WebSphere MQ : wiersze 18, 27 i 36:

```
host: LondonHost
  ...
host: SydneyHost
  ...
host: WashingtonHost
```

Należy zmienić `LondonHost`, `SydneyHost`i `WashingtonHost` na nazwy trzech komputerów, na których działają serwery WebSphere MQ . Można także zmienić nazwy kanału i kolejki, jeśli chcesz (w przykładzie użyto nazw ustawień domyślnych systemu). Możliwe jest również zwiększenie lub zmniejszenie liczby biur w przykładowych danych.

Konfigurowanie serwera IBM Tivoli Directory Server

Informacje na temat instalowania tego katalogu można znaleźć w podręczniku IBM Tivoli Directory Server (ITDS) Administrator's Guide. W temacie `Installing and Configuring Server`praca za pomocą sekcji `Installing Server` i `Basic Server Configuration`. Jeśli to konieczne, zapoznaj się z tematem `Administrator Interface` , aby zapoznać się z pracami interfejsu.

W temacie `Configuring - How Do I`postępuj zgodnie z instrukcjami, aby uruchomić administratora, a następnie postępuj zgodnie z sekcją `Configure Database` i utwórz domyślną bazę danych. Pomiń sekcję `Configure replica` i korzystając z sekcji `Work with Suffixes`, dodaj przyrostek `o=MQuser`.

Przed dodaniem jakichkolwiek pozycji do bazy danych należy rozszerzyć schemat katalogu, dodając definicje atrybutów oraz definicję klasy obiektów. Jest to opisane w publikacji IBM Tivoli Directory Server -Podręcznik administratora w rozdziale `Reference Information` , w sekcji `Directory Schema`. Dołączono dwa pliki przykładowe, które pomogą Ci w tym. Plik `mq.at.conf` zawiera definicje atrybutów, które należy dodać do pliku `?etc?slapd.at.conf`. W tym celu należy dodać przykładowy plik, edytując program `slapd.at.conf` i dodając wiersz:

```
include <pathname>/mq.at.conf
```

Alternatywnie można zmodyfikować plik `slapd.at.conf` i dodać bezpośrednio do niego zawartość przykładowego pliku, to znaczy dodać wiersze:

```
# MQ attribute definitions
attribute mqChannel           ces   mqChannel        1000   normal
```

```
attribute mqQueueManager      ces   mqQueueManager   1000   normal
attribute mqQueue             ces   mqQueue          1000   normal
attribute mqPort              cis   mqPort           64     normal
```

Podobnie w przypadku definicji klasy obiektów można dołączyć przykładowy plik, edytując program `etc?slapd.oc.conf` i dodać wiersz:

```
include <pathname>/mq.oc.conf
```

lub można dodać zawartość przykładowego pliku bezpośrednio do programu `slapd.oc.conf,` , co oznacza, że należy dodać następujące wiersze:

```
# MQ object classdefinition
objectclass mqApplication
      requires
              objectClass,
              cn,
              host,
              mqChannel,
              mqQueue
      allows
              mqQueueManager,
              mqPort,
              description,
              l,
              ou,
              seeAlso
```

Teraz można uruchomić serwer katalogów (Administracja, Serwer, Uruchamianie) i dodać do niego przykładowe wpisy. Aby dodać przykładowe wpisy, przejdź do strony Administrowanie, dodaj wpisy do administratora, wpisz pełną ścieżkę do przykładowego pliku `MQuser.ldif` i kliknij opcję Wyślij.

Serwer katalogów jest teraz uruchomiony i ładowany wraz z danymi odpowiednimi do uruchamiania przykładowego programu.

Konfigurowanie serwera katalogów Netscape

Na stronie Administrowanie serwerem Netscape Server kliknij opcję **Create New Netscape Directory Server**(Utwórz nowy serwer Netscape Directory Server).

Teraz należy przedstawić formularz zawierający informacje konfiguracyjne. Zmień przyrostek katalogu na **o = MQuser** i dodaj hasło dla nieograniczonego użytkownika. Można również zmienić dowolne inne informacje, aby dostosować je do instalacji. Kliknij przycisk **OK**, a katalog powinien zostać utworzony pomyślnie. Kliknij opcję **Return to Server Administration** (Powrót do administrowania serwerem) i uruchom serwer katalogów. Kliknij nazwę katalogu, aby uruchomić serwer Directory Server Administration dla nowego katalogu.

Przed dodaniem jakichkolwiek pozycji do bazy danych należy rozszerzyć schemat katalogu, dodając definicje atrybutów oraz definicję klasy obiektów. Kliknij kartę **Schemat** na stronie Serwer katalogów. Zostanie teraz przedstawiony formularz, który umożliwia dodawanie nowych atrybutów. Dodaj następujące atrybuty (pozostaw pusty identyfikator OID atrybutu dla wszystkich tych atrybutów):

```
Attribute Name              Syntax
--------------              ------
mqChannel                   Case Exact String
mqQueueManager              Case Exact String
mqQueue                     Case Exact String
mqPort                      Integer
```

Dodaj nową klasę objectClass , klikając opcję **Create ObjectClass** w panelu bocznym. Wprowadź wartość **mqApplication** jako nazwę klasy ObjectClass , wybierz element **applicationProcess** jako element nadrzędny ObjectClass i pozostaw wartość pustą **ObjectClass OID** . Następnie należy dodać niektóre atrybuty do klasy objectClass. Jako atrybuty wymagane wybierz opcję **host**, **mqChannel**i **mqQueue** , a następnie wybierz opcję **mqQueueManager** i **mqPort** jako dozwolone atrybuty. Naciśnij przycisk **Create New ObjectClass** (Utwórz nową klasę obiektu), aby utworzyć klasę objectClass.

Aby dodać przykładowe dane, kliknij kartę **Zarządzanie bazą danych** i wybierz opcję **Dodaj pozycje** z panelu bocznego. Wprowadź nazwę ścieżki do przykładowego pliku danych `<pathname>\MQuser.ldif`, wprowadź hasło i kliknij przycisk **OK**.

Przykładowy program działa jako nieautoryzowany użytkownik, a domyślnie Netscape Directory nie zezwala na wyszukiwanie w katalogu przez nieuprawnionych użytkowników. Tę zmianę należy zmienić, klikając kartę **Kontrola dostępu** . Wprowadź hasło dla użytkownika nieograniczonego, a następnie kliknij przycisk **OK** , aby załadować pozycje kontroli dostępu dla katalogu. Te wartości powinny być obecnie puste. Aby utworzyć nowy wpis kontroli dostępu, należy nacisnąć przycisk **Nowy element ACI** . W wyświetlonym polu wprowadzania kliknij opcję **Odmów** (co jest podkreślone), a w wyświetlonym oknie dialogowym zmień wartość na **Zezwalaj**. Dodaj nazwę, na przykład **MQuser-access**, a następnie kliknij **wybierz przyrostek** , aby wybrać opcję **o = użytkownik_MQuser**. Wprowadź wartość **o = użytkownik MQuser** jako element docelowy, wprowadź hasło dla nieograniczonego użytkownika i kliknij przycisk **Wyślij**.

Serwer katalogów jest teraz uruchomiony i ładowany wraz z danymi odpowiednimi do uruchamiania przykładowego programu.

Uruchamianie przykładowego programu

Serwer LDAP Directory Server powinien być uruchomiony i zapełniony danymi przykładowymi. Dane określają trzy komputery hosta, z których wszystkie powinny być uruchomione na serwerach WebSphere MQ . Upewnij się, że domyślny menedżer kolejek jest uruchomiony na każdym komputerze (chyba że zmieniono przykładowe dane w celu określenia innego menedżera kolejek).

Należy również uruchomić program nasłuchujący WebSphere MQ na każdym komputerze; w przykładzie jest używany protokół TCP/IP z domyślnym numerem portu WebSphere MQ , co umożliwia uruchomienie nasłuchiwania za pomocą komendy:

```
runmqlsr -t tcp
```

Aby przetestować przykład, można również uruchomić program w celu odczytania komunikatów przychodzących do każdego serwera WebSphere MQ , na przykład w celu użycia programu przykładowego amqstrg:

```
amqstrg SYSTEM.DEFAULT.LOCAL.QUEUE
```

Program przykładowy wykorzystuje trzy zmienne środowiskowe, jedno wymagane i dwa opcjonalne. Wymagana zmienna to LDAP_BASEDN, które określa podstawową nazwę wyróżniającą dla wyszukiwania katalogu. Aby pracować z przykładowymi danymi, należy ustawić tę wartość na `ou=Transport, o=MQuser`, na przykład w wierszu komend w systemach Windows :

```
set LDAP_BASEDN=ou=Transport, o=MQuser
```

Opcjonalne zmienne to LDAP_HOST i LDAP_VERSION. Zmienna LDAP_HOST określa nazwę hosta, na którym jest uruchomiony serwer LDAP. Jeśli nie jest określony, domyślnie jest on używany na hoście lokalnym. Zmienna LDAP_VERSION określa wersję protokołu LDAP, która ma być używana, i może mieć wartość 2 lub 3. Większość serwerów LDAP obsługuje teraz wersję 3 protokołu; wszystkie te serwery obsługują starszą wersję 2. Ten przykład działa równie dobrze z każdą wersją protokołu, a jeśli nie jest określony, to domyślnie jest to wersja 2.

Można teraz uruchomić przykład, wpisując nazwę programu, po której następuje nazwa aplikacji WebSphere MQ , do której mają być wysyłane komunikaty, w przypadku przykładowych danych, których nazwy aplikacji są następujące: Londyn, Sydney i Waszyngton. Na przykład, aby wysłać wiadomości do aplikacji londyńskiej:

```
amqsldpc London
```

Jeśli nawiązanie połączenia z serwerem WebSphere MQ nie powiedzie się, zostanie wyświetlony odpowiedni komunikat o błędzie. Jeśli połączenie zostanie pomyślnie nawiązane, można rozpocząć wpisywanie komunikatów, przy czym każdy typ wiersza (zakończony przez element < return> lub < enter>) jest wysyłany jako osobny komunikat, a pusty wiersz kończy działanie programu.

Projekt programu

Program składa się z dwóch odrębnych części: pierwsza część używa zmiennych środowiskowych i wartości wiersza komend do odpytywania serwera katalogów LDAP; druga część nawiązuje połączenie WebSphere MQ z wykorzystaniem informacji zwróconych z katalogu i wysyła komunikaty.

Wywołania LDAP używane w pierwszej części programu różnią się nieznacznie w zależności od tego, czy używany jest protokół LDAP w wersji 2, czy 3. Są one szczegółowo opisane w dokumentacji dostarczanej wraz z bibliotekami klienta LDAP. W tej sekcji znajduje się krótki opis.

Pierwsza część programu sprawdza, czy została ona poprawnie wywołana i odczyta zmienne środowiskowe. Następnie nawiązuje połączenie z serwerem katalogów LDAP na podanym hoście:

```
if (ldapVersion == LDAP_VERSION3)
{
  if ((ld = ldap_init(ldapHost, LDAP_PORT)) == NULL)
     ...
}
else
{
  if ((ld = ldap_open(ldapHost, LDAP_PORT)) == NULL )
     ...
}
```

Gdy połączenie zostało nawiązane, program ustawia niektóre opcje na serwerze za pomocą wywołania "ldap_set_option", a następnie uwierzytelnia się na serwerze poprzez powiązanie go z nim:

```
if (ldapVersion == LDAP_VERSION3)
{
  if (ldap_simple_bind_s(ld, bindDN, password) != LDAP_SUCCESS)
     ...
}
else
{
  if (ldap_bind_s(ld, bindDN, password, LDAP_AUTH_SIMPLE) !=
      LDAP_SUCCESS)
     ...
}
```

W przykładowym programie `bindDN` i `password` ustawione są wartości NULL, co oznacza, że program uwierzytelnia się jako anonimowy użytkownik, to znaczy nie ma żadnych specjalnych praw dostępu i może uzyskać dostęp tylko do informacji, które są publicznie dostępne. W praktyce większość organizacji ogranicza dostęp do informacji, które przechowują w katalogach, tak aby tylko autoryzowani użytkownicy mieli do niego dostęp.

Pierwszym parametrem dla wywołania powiązania `ld` jest uchwyt używany do identyfikowania tej konkretnej sesji LDAP przez cały czas trwania programu. Po uwierzytelnieniu program przeszukuje katalog pod kątem wpisów zgodnych z nazwą aplikacji:

```
rc = ldap_search_s(ld,                /* LDAP Handle             */
                   baseDN,            /* base distinguished name */
                   LDAP_SCOPE_ONELEVEL, /* one-level search       */
                   filterPattern,     /* filter search pattern   */
                   attrs,             /* attributes required     */
                   FALSE,             /* NOT attributes only     */
                   &ldapResult);      /* search result           */
```

Jest to proste wywołanie synchroniczne na serwerze, które bezpośrednio zwraca wyniki. Istnieją inne typy wyszukiwania, które są bardziej odpowiednie dla złożonych zapytań lub w przypadku, gdy oczekiwana jest duża liczba wyników. Pierwszym parametrem dla wyszukiwania jest uchwyt `ld` , który identyfikuje sesję. Drugim parametrem jest podstawowa nazwa wyróżniająca, która określa miejsce w katalogu, w którym

ma zostać rozpoczęte wyszukiwanie, a trzecim parametrem jest zasięg wyszukiwania, tj. przeszukiwane są pozycje względne względem punktu początkowego. Te dwa parametry razem definiują, które pozycje w katalogu są przeszukiwane. Następny parametr, `filterPattern` określa to, czego szukamy. Parametr `attrs` zawiera listę atrybutów, które mają zostać odzyskane z obiektu, gdy go znaleziono. Następny atrybut mówi, czy chcemy tylko atrybuty, czy też ich wartości; ustawienie wartości FALSE oznacza, że chcemy wartości atrybutów. Końcowy parametr jest używany do zwrócenia wyniku.

Wynik może zawierać wiele pozycji katalogu, a każdy z nich ma określone atrybuty i ich wartości. Musimy wydobyć z tego wyniku wartości, które chcemy. W tym przykładowym programie spodziewamy się tylko jednego wpisu, więc przyjrzymy się tylko pierwszemu wpisowi w wyniku:

```
ldapEntry = ldap_first_entry(ld, ldapResult);
```

Wywołanie to zwraca uchwyt, który reprezentuje pierwszą pozycję, a my ustawiamy pętlę do wyodrębnienia wszystkich atrybutów z pozycji:

```
for (attribute = ldap_first_attribute(ld, ldapEntry, &ber);
     attribute != NULL;
     attribute = ldap_next_attribute(ld, ldapEntry, ber ))
{
```

Dla każdego z tych atrybutów wyodrębniamy wartości powiązane z tym atrybutem. Ponownie oczekujemy tylko jednej wartości na jeden atrybut, więc używamy tylko pierwszej wartości; określamy, który atrybut mamy i przechowujemy wartość w odpowiedniej zmiennej programowej:

```
values = ldap_get_values(ld, ldapEntry, attribute);
if (values != NULL && values[0] != NULL)
{
  if (stricmp(attribute, MQ_HOST_ATTR) == 0)
  {
    mqHost = strdup(values[0]);
    ...
```

Na koniec przechyliliśmy się przez zwalnianie pamięci (ldap_value_free, ldap_memfree, ldap_msgfree) i zamknięcie sesji przez *unbinding* z serwera:

```
ldap_unbind(ld);
```

Sprawdzamy, czy znaleźliśmy wszystkie wymagane przez nas wartości WebSphere MQ z katalogu, a jeśli tak, to wywołujemy metodę sendMessages(), aby połączyć się z serwerem WebSphere MQ i wysłać komunikaty WebSphere MQ .

Drugą częścią przykładowego programu jest procedura sendMessages() zawierająca wszystkie wywołania programu WebSphere MQ . Jest to wzorowane na przykładowym programie amqsput0 , różnice w tym, że parametry do programu zostały rozszerzone, a zamiast wywołania MQCONN używane jest zmaterializowane tabele MQCONNX.

Tworzenie aplikacji dla produktu IBM WebSphere MQ Telemetry

Aplikacje telemetryczne integrują urządzenia sensowne i sterujące z innymi źródłami informacji dostępnymi w internecie oraz w przedsiębiorstwach.

Tworzenie aplikacji dla produktu IBM WebSphere MQ Telemetry przy użyciu wzorców projektowych, obrobionych przykładów, przykładowych programów, pojęć związanych z programowaniem i informacji referencyjnych. Aby uprościć łączenie wielu małych urządzeń z systemem IBM WebSphere MQ, należy użyć demona IBM WebSphere MQ Telemetry .

Pojęcia pokrewne

WebSphere MQ Telemetry

Koncepcje telemetrii oraz scenariusze monitorowania i kontroli

Zadania pokrewne

Instalowanie produktu WebSphere MQ Telemetry

Administrowanie produktem WebSphere MQ Telemetry

Rozwiązywanie problemów dotyczących produktu WebSphere MQ Telemetry

Odsyłacze pokrewne

Informacje dodatkowe o produkcie WebSphere MQ Telemetry

IBM WebSphere MQ Telemetry programy przykładowe

Sample scripts are provided to demonstrate the basic usage of the MQ Telemetry Transport v3 Client application. Użyj skryptów, aby opublikować komunikat i zasubskrybować temat.

Zanim rozpoczniesz

Uruchom usługę telemetryczną (MQXR), aby uruchomić programy przykładowe.

Identyfikator użytkownika musi należeć do grupy użytkowników mqm.

Najpierw uruchom skrypt SampleMQM , a następnie skrypt MQTTV3Sample , aby wykonać operację publikowania i subskrybowania. Uruchom przykładowy skrypt CleanupMQM , aby usunąć menedżer kolejek utworzony za pomocą skryptu SampleMQM .

Ze względu na to, że skrypt SampleMQM tworzy i używa menedżera kolejek o nazwie QM1, nie należy uruchamiać w systemie niezmienionych w systemie z menedżerem kolejek QM1 . Wszelkie wprowadzone zmiany mogą mieć wpływ na konfigurację istniejącego menedżera kolejek.

O tym zadaniu

- Aplikacja SampleMQM tworzy i uruchamia menedżer kolejek z włączoną obsługą telemetrii o nazwie QM1. Skrypt konfiguruje także domyślną kolejkę transmisji dla QM1, a następnie tworzy i uruchamia domyślny kanał nasłuchujący na porcie 1883. Kanał ten nie uwierzytelnia klientów połączonych z tym kanałem. Kanał ma atrybut Identyfikator użytkownika agenta kanału komunikatów (MCAUSER), ustawiony jako 'gość' w systemach Windows lub 'nobody' w systemach Linux . Klienty połączone z kanałem są traktowane jak użytkownik 'gość' lub użytkownik 'nobody', w zależności od systemu, w którym jest on uruchomiony. Skrypt autoryzuje 'gość' w systemach Windows i 'nobody' w systemach Linux , aby móc publikować i subskrybować dowolny temat na QM1 .
- Aplikacja MQTTV3Sample znajduje się w następującym miejscu:
  - W systemie Windows *MQ_INSTALLATION_PATH*\mqxr\samples

    gdzie *ŚCIEŻKA_INSTALACJI_PRODUKTU_MQ* to miejsce, w którym zainstalowano produkt IBM WebSphere MQ .
  - W systemie Linux *MQ_INSTALLATION_PATH*/mqxr/samples

  Aplikacja MQTTV3Sample działa jako publikator, wysyłając pojedynczy komunikat do tematu na serwerze. Działa on również jako subskrybent, nasłuchuje komunikatów z serwera.
- Przykładowy skrypt CleanupMQM kończy działanie i usuwa QM1 , który został utworzony przez skrypt SampleMQM . Użyj przykładowego skryptu CleanupMQM , jeśli chcesz ponownie uruchomić skrypt SampleMQM lub usuń QM1.

Procedura

1. Aby uruchomić skrypt SampleMQM , wpisz następującą komendę w wierszu komend.
   - W systemie Windowskomenda do uruchomienia skryptu SampleMQM jest następująca:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\SampleMQM.bat
     ```

- W systemach AIX i Linuxkomenda uruchamiania skryptu SampleMQM jest następująca:

  ```
  MQ_INSTALLATION_PATH/mqxr/samples/SampleMQM.sh
  ```

  gdzie *ŚCIEŻKA_INSTALACJI_PRODUKTU_MQ* to miejsce, w którym zainstalowano produkt IBM WebSphere MQ .

  Tworzony jest menedżer kolejek o nazwie MQXR_SAMPLE_QM.

2. Wpisz następującą komendę, aby uruchomić pierwszą część skryptu MQTTV3Sample .
   - W systemie Windows, w jednym wierszu komend wpisz następującą komendę:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\RunMQTTV3Sample.bat -a subscribe
     ```

   - W systemach AIX i Linux, w jednym oknie powłoki, wpisz następującą komendę:

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh -a subscribe
     ```

3. Wpisz następującą komendę, aby uruchomić drugą część skryptu MQTTV3Sample .
   - W systemie Windows, w innym wierszu komend, wpisz następującą komendę:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\RunMQTTV3Sample.bat -m "Hello from an MQTT v3
     application"
     ```

   - W systemach AIX i Linux, w innym oknie powłoki, wpisz następującą komendę:

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh -m "Hello from an MQTT v3 application"
     ```

4. Aby usunąć menedżera kolejek utworzonego za pomocą skryptu SampleMQM , można uruchomić skrypt CleanupMQM za pomocą następującej komendy:
   - W systemie Windowswpisz następującą komendę:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\CleanupMQM.bat
     ```

   - W systemach AIX i Linux w innym oknie powłoki wpisz następującą komendę:

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/CleanupMQM.sh
     ```

Wyniki

Komunikat `Hello from an MQTT v3 application` , który został wpisany w drugim oknie, zostanie opublikowany przez tę aplikację i odebrany przez aplikację w pierwszym oknie. Aplikacja w pierwszym oknie pokaże ją na ekranie.

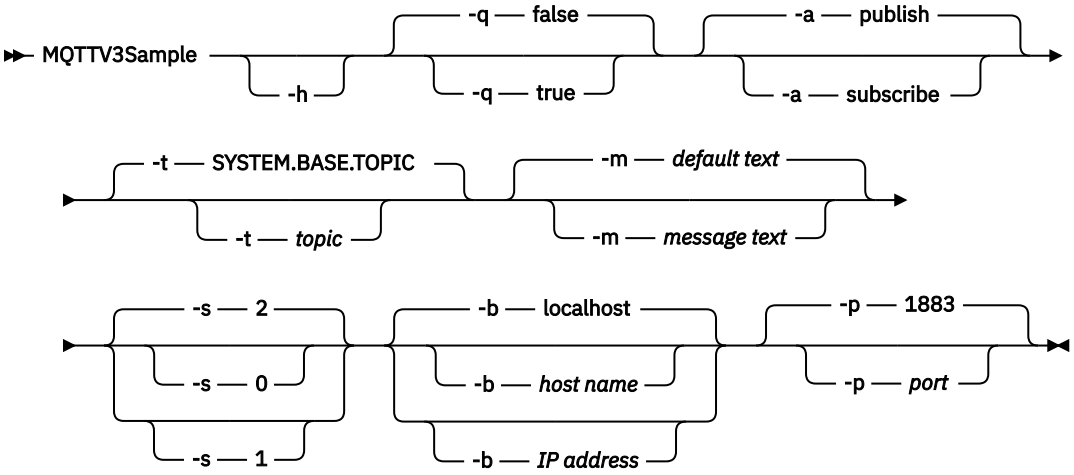
Program MQTTV3Sample

Informacje uzupełniające na temat przykładowej składni i parametrów dla programu MQTTV3Sample .

Przeznaczenie

Program MQTTV3Sample może być używany do publikowania komunikatów i subskrybowania tematu.

MQTTV3Sample syntax



Parametry

-h
Wydrukuj ten tekst pomocy i zakończ

-q
Ustaw tryb cichy, zamiast używać domyślnego trybu false.

-a
Ustaw publikowanie lub subskrybowanie, zamiast zakładać domyślne działanie publikowania.

-t
Publikowanie lub subskrybowanie tematu, zamiast publikowania lub subskrybowania tematu domyślnego

-m
Publish message text instead of sending the default publication text, "Hello from an MQTT v3 application".

-s
Ustaw wartość QoS zamiast domyślnej wartości QoS, 2.

-b
Połącz się z tą nazwą hosta lub adresem IP, a nie nawiąże połączenia z domyślną nazwą hosta localhost.

-p
Użyj tego portu, zamiast używać domyślnego, 1883.

Uruchom program MQTTV3Sample .

Aby zasubskrybować temat w systemie Windows, należy użyć komendy:

```
runMQTTV3Sample -a subscribe
```

Aby opublikować komunikat w systemie Windows, należy użyć następującej komendy:

```
runMQTTV3Sample
```

Więcej informacji na temat uruchamiania przykładowych skryptów zawiera sekcja “IBM WebSphere MQ Telemetry programy przykładowe” na stronie 485.

Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport przy użyciu języka Java

Kroki tworzenia aplikacji klienckiej MQTT są opisane w sposób samouczek. Każda linia kodu jest wyjaśniana. Po zakończeniu zadania zostanie utworzony publikator MQTT. Publikacje można przeglądać za pomocą programu WebSphere MQ Explorer.

Zanim rozpoczniesz

Zainstaluj składnik WebSphere MQ Telemetry na serwerze, na którym jest zainstalowany produkt IBM WebSphere MQ Version 7.1 lub nowszy.

Aplikacja kliencka korzysta z pakietu `com.ibm.mq.micro.client.mqttv3` w pakiecie IBM WebSphere MQ Telemetry Software Development Toolkit (SDK). Pakiet SDK jest częścią instalacji produktu IBM WebSphere MQ Telemetry . Klient łączy się z funkcją IBM WebSphere MQ Telemetry w celu wymiany komunikatów z produktem IBM WebSphere MQ.

Aby administrować produktem IBM WebSphere MQ Telemetry, należy również zainstalować aktualizacje telemetryczne dla produktu IBM WebSphere MQ Explorer Version 7.1 . Aktualizacje są częścią instalacji produktu IBM WebSphere MQ Telemetry .

Klient MQTT, działający w środowisku Java SE, wymaga wersji 6.0 języka Java SE lub nowszej. IBM Java SE v6.0 jest częścią instalacji produktu IBM WebSphere MQ Version 7.1 . Znajduje się on w katalogu *WebSphere MQ installation directory*`\java\jre` .

O tym zadaniu

Przykładem jest aplikacja publikowania, PubSync. Produkt PubSync publikuje `Hello World` w temacie `MQTT Examples`i oczekuje na potwierdzenie, że publikacja została dostarczona do menedżera kolejek.

Ustawiając trwałą subskrypcję programu `MQTT Examples` , można sprawdzić, czy aplikacja działa.

W ramach tej procedury stosowana jest Eclipse do tworzenia, budowania i uruchamiania klienta. Produkt Eclipse można pobrać z serwisu WWW projektu Eclipse pod adresem www.eclipse.org.

Aby utworzyć aplikację, można utworzyć pliki Java, a następnie skompilować i uruchomić je za pomocą wiersza komend.

W nowym katalogu utwórz ścieżkę katalogu `.\com\ibm\mq\id`. Utwórz dwa pliki Java, `Example.java` i PubSync`.java`. Skopiuj kod z programu “Przykładowy kod” na stronie 492 do plików Java.

Skompiluj kod Java za pomocą komendy,

```
javac -cp jar_dir\com.mq.micro.client.mqttv3.jar
       com.ibm.mq.id.PubSync.java com.ibm.mq.id.Example.java
```

Uruchom program PubSync przy użyciu komendy,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubSync
```

Procedura

1. Utwórz projekt Java w środowisku Eclipse.
   a) **Plik > Nowy > Projekt Java** i wpisz nazwę projektu. Kliknij przycisk **Dalej**.

      Sprawdź, czy środowisko JRE jest w poprawnej lub nowszej wersji. Środowisko Java SE musi być w wersji 6.0 lub nowszej.

   b) Na stronie Ustawienia Java kliknij opcję **Biblioteki > Dodaj zewnętrzne pliki JAR ...**
   c) Przejdź do katalogu, w którym zainstalowano folder pakietu SDK produktu WebSphere MQ Telemetry. Odszukaj folder SDK`\clients\java` i wybierz wszystkie pliki `.jar` > **> Otwórz > Zakończ**.

2. Zainstaluj dokumentację Javadoc klienta MQTT.

   W przypadku zainstalowania dokumentacji Javadoc klienta MQTT edytor Java zapewnia asystę przy użyciu klas MQTT v3 .

   a) W projekcie Java otwórz okno **Eksplorator pakietów > Przywoływane biblioteki**. Kliknij prawym przyciskiem myszy opcję `com.ibm.micro.client.mqttv3.jar` **> Właściwości**.

   b) W nawigatorze właściwości kliknij opcję **Położenie dokumentacji Javadoc**.

   c) Na stronie Położenie dokumentacji Javadoc kliknij opcję **Adres URL dokumentacji Javadoc > Przeglądaj ...** i znajdź folder *`WMQ Installation directory`*`\mqxr\SDK\clients\java\doc\javadoc` **> OK**.

   d) Kliknij przycisk **Sprawdź poprawność ... > OK** .

   Zostanie wyświetlona prośba o otwarcie przeglądarki w celu wyświetlenia dokumentacji.

3. Utwórz klasę PubSyncprzy użyciu kreatora klas Java.

   a) Kliknij prawym przyciskiem myszy projekt Java, który został utworzony > **Nowy** > **Klasa**.

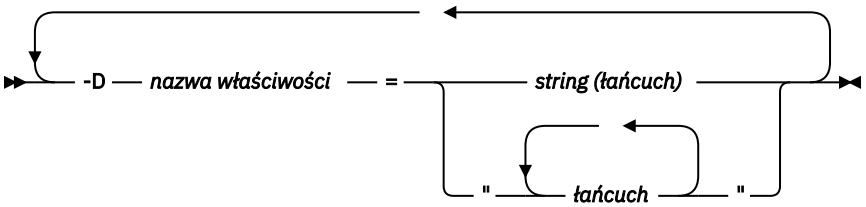
   b) Wpisz nazwę pakietu, `com.ibm.mq.id`

   c) Wpisz nazwę klasy, PubSync

   d) Sprawdź pole kodu pośredniczącego metody, **public static void main (String [] args)**

4. Utwórz plik `Example.java` w pakiecie `com.ibm.mq.id`. Skopiuj kod w programie Rysunek 89 na stronie 493 do pliku.

   Wszystkie parametry użyte w przykładach są ustawiane jako właściwości. Wartości można przesłonić, zmieniając wartości domyślne w programie `Example.java`lub podając właściwości jako opcje w wierszu komend Java przy użyciu parametru `-D` :



   Identyfikator klienta użyty w tym przykładzie, a także przykłady “Tworzenie asynchronicznego publikatora dla produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 493 , jest nazwą użytkownika z przyrostkiem określonym przez łańcuch losowy.

5. Wykonaj kroki, aby utworzyć kod, lub skopiuj kod z programu Rysunek 88 na stronie 492.

   Kroki, które należy wykonać, wyjaśniają kod w produkcie `Pubsync.java`.

6. Utwórz blok `try-catch` .

```
try { ...
 } catch (Exception e) {
  e.printStackTrace();
}
```

   Klient MQTT zgłasza `MqttException`, `MqttPersistenceException` lub `MqttSecurityException`. `MqttPersistenceException` i `MqttSecurityException` są podklasami produktu `MqttException`.

   Użyj metody `MqttException.getReasonCode` , aby dowiedzieć się, dlaczego wystąpił wyjątek. Jeśli zostanie zgłoszony wyjątek `MqttPersistenceException` lub `MqttSecurityException` , należy użyć metody `getCause` , aby zwrócić bazowy wyjątek throwable.

7. Utwórz nową instancję produktu `MqttClient` .

```
MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
```

Należy udostępnić klientowi adres serwera, który jest używany później w celu nawiązania połączenia z produktem WebSphere MQ. Ustaw identyfikator klienta, aby nazwać klienta.

- Opcjonalnie można udostępnić implementację interfejsu `MqttClientPersistence` w celu zastąpienia implementacji domyślnej. Domyślna implementacja produktu `MqttPersistence` przechowuje komunikaty QoS 1 i 2 oczekujące na dostarczenie jako pliki; patrz sekcja “Trwałość komunikatu w klientach MQTT” na stronie 547.
- Domyślny port TCP/IP IBM WebSphere MQ dla MQTT to 1883. Dla SSL jest to 8883. W tym przykładzie adres domyślny jest ustawiony na wartość `tcp://localhost:1883`.
- Zwykle ważne jest, aby być w stanie zidentyfikować konkretnego klienta fizycznego przy użyciu identyfikatora klienta. Identyfikator klienta musi być unikalny dla wszystkich klientów łączy się z serwerem; patrz “Identyfikator klienta” na stronie 543. Użycie tego samego identyfikatora klienta co w poprzedniej instancji wskazuje, że obecna instancja jest instancją tego samego klienta. Jeśli identyfikator klienta zostanie zduplikowany w dwóch uruchomionych klientach, w obu klientach zostanie zgłoszony wyjątek, a jeden klient kończy działanie.
- Długość identyfikatora klienta jest ograniczona do 23 bajtów. Jeśli długość zostanie przekroczona, zgłaszany jest wyjątek. Identyfikator klienta musi zawierać tylko znaki dozwolone w nazwie menedżera kolejek, na przykład brak łączników lub spacji.
- Dopóki nie zostanie wywołana metoda `MqttClient.connect` , przetwarzanie komunikatów nie jest wymagane.

Za pomocą obiektu klienta można publikować i subskrybować tematy oraz odzyskiwać informacje o publikacjach, które nie zostały jeszcze dostarczone.

8. Utwórz temat, który ma zostać opublikowany.

```
MqttTopic topic = client.getTopic(Example.topicString);
```

Łańcuch tematu jest ograniczony do 64 kB, co przekracza maksymalną długość łańcucha tematu IBM WebSphere MQ . W przeciwnym razie łańcuch tematu będzie zgodny z tymi samymi regułami, co łańcuchy tematów produktu WebSphere MQ . Patrz sekcja Łańcuchy tematów. W przykładzie przedstawiono łańcuch tematu `MQTT Examples`.

9. Utwórz komunikat publikacji.

```
MqttMessage message = new MqttMessage(Example.publication.getBytes());
```

Łańcuch `"Hello World"` jest przekształcany w tablicę bajtów i używany do tworzenia partycji `MqttMessage`.

- Ładunek komunikatu MQTT jest zawsze tablicą bajtów. Metoda `getBytes` przekształca obiekt łańcuchowy na UTF-8. `MqttMessage` ma wygodną metodę `toString` , aby zwrócić ładunek komunikatu w postaci łańcucha. Jest on równoważny `new string(message.getPayload)`
- Komunikat publikacji jest wysyłany do menedżera kolejek z nagłówkiem RFH2 , a dane komunikatu są wysyłane jako komunikat `jms-bytes` .
- Obiekt komunikatu ma jakość usługi i zachowane atrybuty. Jakość usługi (QoS) określa, jak niezawodnie jest przesyłany komunikat między klientem MQTT a menedżerem kolejek. Patrz sekcja “Jakość usług świadczonych przez klienta MQTT” na stronie 551. Zachowany atrybut określa, czy publikacja jest przechowywana przez menedżera kolejek dla przyszłych subskrybentów. Jeśli publikacja nie została zachowana, jest ona wysyłana tylko do bieżących subskrybentów; patrz “Zachowane publikacje i klienty MQTT” na stronie 553. Domyślne ustawienia produktu `MqttMessage` to: "Komunikaty są dostarczane co najmniej raz i nie są zachowywane."

10. Połącz się z serwerem.

```
client.connect();
```

Przykład łączy się z serwerem przy użyciu domyślnych opcji połączenia. Po nawiązaniu połączenia można rozpocząć publikowanie. Domyślne opcje połączenia to:

- Mały komunikat "keep-alive" jest wysyłany co 15 sekund, aby zapobiec zamknięciu połączenia TCP/IP.
- Sesja jest uruchamiana bez sprawdzania, czy zostały ukończenie poprzednich publikacji.
- Odstęp czasu między ponowną próbą wysłania komunikatu wynosi 15 sekund.
- Dla połączenia nie zostanie utworzony ostatni komunikat testamentu i testament.
- Do utworzenia połączenia używana jest standardowa fabryka SocketFactory .

Zmień opcje połączenia, tworząc obiekt `ConnectionOptions` i przekazując go jako dodatkowy parametr do `client.connect`.

11. Publikuj.

```
MqttDeliveryToken token = topic.publish(message);
```

W tym przykładzie jest wysyłany publikacja "Hello World" w temacie "Przykłady MQTT" do menedżera kolejek.

- Po powrocie z metody `publish` komunikat jest bezpiecznie przesyłany do klienta MQTT, ale nie został jeszcze przesłany na serwer. Jeśli komunikat ma wartość QoS 1 lub 2, komunikat jest przechowywany lokalnie, w przypadku, gdy klient nie powiedzie się przed zakończeniem dostarczania.
- Produkt `publish` zwraca znacznik dostarczania, który jest używany do sprawdzania, czy potwierdzenie zostało jeszcze odebrane z serwera.

12. Poczekaj na potwierdzenie z serwera.

```
token.waitForCompletion(Example.timeout);
```

Przykład PubSync czeka na potwierdzenie z serwera, który potwierdza, że wiadomość została dostarczona.

- Bez limitu czasu, klient będzie czekał bezterminowo. Zadanie “Tworzenie asynchronicznego publikatora dla produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 493 przedstawia sposób odbierania potwierdzeń bez oczekiwania przy użyciu obiektu wywołania zwrotnego.

13. Odłącz klienta od serwera.

```
client.disconnect();
```

Klient rozłącza się z serwerem i czeka na wszystkie metody `MqttCallback` , które są wykonywane do końca. Następnie czeka aż 30 sekund, aby zakończyć wszystkie pozostałe prace. Jako dodatkowy parametr można określić limit czasu wygaszania.

14. Zapisz zmiany w produkcie PubSync `.java` i `Example.Java`

Środowisko Eclipse automatycznie kompiluje język Java. Teraz można zobaczyć wyniki, uruchamiając program.

Wyniki

Aby wyświetlić publikacje za pomocą produktu WebSphere MQ, należy utworzyć temat, kolejkę i trwałą subskrypcję, a wszystko to wywołanie funkcji `"MQTTExampleTopic"` przy użyciu skryptu w produkcie Rysunek 87 na stronie 492. Uruchom klienta, aby opublikować w temacie `MQTT Examples` , a następnie uruchom program przykładowy **amqsbcg** , aby przeglądać publikacje w kolejce produktu `MQTTExamples` .

1. Uruchom menedżer kolejek i uruchom jego usługę telemetryczną (MQXR). Upewnij się, że adres TCP/IP i port skonfigurowany dla kanału telemetrycznego są zgodne z wartościami używannymi w aplikacji MQTT.
2. Skonfiguruj trwałą subskrypcję, tworząc skrypt komend `mqttexamples.txt` i uruchamiając go za pomocą programu **runmqsc:** .

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Rysunek 87. mqttExampleTopic.txt

Aby uruchomić skrypt w systemie Windows, wpisz komendę:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Uruchom klienta jako aplikację Java z poziomu środowiska Eclipselub za pomocą środowiska Java w oknie komend:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Uwaga: Okno komend musi być otwarte w katalogu, w którym znajduje się ścieżka com\ibm\mq\id.

4. Przejrzyj wyniki za pomocą programu WebSphere MQ Explorer lub uruchom następującą komendę:

```
amqsbcg MQTTExampleQueue queue manager name
```

Przykładowy kod

Plik PubSync.java jest pełną listą kodu opisanego w sekcji Procedura. Zmodyfikuj klasę Example w produkcie Rysunek 89 na stronie 493 w taki sposób, aby przesłonić parametry domyślne używane w pliku PubSync.java.

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubSync {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      client.connect();
      System.out.println("Waiting for up to " + Example.sleepTimeout / 1000
          + " seconds for publication of \"" + message.toString()
          + "\" with QoS = " + message.getQos());
      System.out.println("On topic \"" + topic.getName()
          + "\" for client instance: \"" + client.getClientId()
          + "\" on address " + client.getServerURI() + "\"");
      MqttDeliveryToken token = topic.publish(message);
      token.waitForCompletion(Example.sleepTimeout);
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
      client.disconnect();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Rysunek 88. PubSync.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean          retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Rysunek 89. Example.java

Pojęcia pokrewne

Aplikacje publikowania/subskrybowania MQTT

Tworzenie asynchronicznego publikatora dla produktu MQ Telemetry Transport przy użyciu języka Java

W ramach tego zadania użytkownik śledzi kurs, aby zmodyfikować pierwszą aplikację publikatora. Modyfikacje umożliwiają aplikacji wysyłanie publikacji bez oczekiwania na potwierdzenie dostarczenia. Potwierdzenie dostarczenia jest odbierane przez klasę wywołania zwrotnego, która została utworzona.

Zanim rozpoczniesz

Zainstaluj składnik WebSphere MQ Telemetry na serwerze, na którym jest zainstalowany produkt IBM WebSphere MQ Version 7.1 lub nowszy.

Aplikacja kliencka korzysta z pakietu `com.ibm.mq.micro.client.mqttv3` w pakiecie IBM WebSphere MQ Telemetry Software Development Toolkit (SDK). Pakiet SDK jest częścią instalacji produktu IBM WebSphere MQ Telemetry . Klient łączy się z funkcją IBM WebSphere MQ Telemetry w celu wymiany komunikatów z produktem IBM WebSphere MQ.

Aby administrować produktem IBM WebSphere MQ Telemetry, należy również zainstalować aktualizacje telemetryczne dla produktu IBM WebSphere MQ Explorer Version 7.1 . Aktualizacje są częścią instalacji produktu IBM WebSphere MQ Telemetry .

Klient MQTT, działający w środowisku Java SE, wymaga wersji 6.0 języka Java SE lub nowszej. IBM Java SE v6.0 jest częścią instalacji produktu IBM WebSphere MQ Version 7.1 . Znajduje się on w katalogu *WebSphere MQ installation directory*\java\jre .

O tym zadaniu

Przykładem jest aplikacja publikowania, PubAsync. Produkt PubAsync publikuje `Hello World` w temacie `MQTT Examples`, nie czekając na potwierdzenie, że publikacja została dostarczona do menedżera kolejek. Potwierdzenie dostarczenia jest odbierane w klasie wywołania zwrotnego `CallBack`.

Ustawiając trwałą subskrypcję programu `MQTT Examples` , można sprawdzić, czy aplikacja działa.

W ramach tej procedury stosowana jest Eclipse do tworzenia, budowania i uruchamiania klienta. Produkt Eclipse można pobrać z serwisu WWW projektu Eclipse pod adresem www.eclipse.org.

Kroki opisane w sekcji Procedura modyfikują aplikację PubSync `.java` w produkcie “Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 488.

Alternatywnie można skopiować kod “Przykładowy kod” na stronie 496do nowego katalogu `.\com\ibm\mq\id`. Utwórz trzy pliki Java, `Example.java`, `CallBack.java`i PubAsync `.java`. Skompiluj przykłady za pomocą komendy,

```
javac -cp jar_dir\com.mq.micro.client.mqttv3.jar
       com.ibm.mq.id.PubAsync.java com.ibm.mq.id.CallBack.java com.ibm.mq.id.Example.java
```

Uruchom program PubAsync przy użyciu komendy,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubAsync.class
```

Procedura

1. W pakiecie `com.ibm.mq.id` utwórz plik `CallBack.java`. Skopiuj kod w programie Rysunek 92 na stronie 497 do pliku.

   Produkt `Callback.java` implementuje interfejs `MqttCallBack` . W tym przykładzie dodatkowy konstruktor inicjuje wywołanie zwrotne z niektórymi danymi instancji.
2. W pakiecie `com.ibm.mq.id` kliknij prawym przyciskiem myszy PubSync `.java` i skopiuj go. Wklej go do tego samego pakietu, zmieniając jego nazwę na PubAsync.
3. Tuż przed wierszem `client.connect();` kodu, utwórz instancję klasy `CallBack` , przekazując identyfikator klienta.

   ```
   CallBack callback = new CallBack(Example.clientId);
   client.setCallback(callback);
   ```

   - Klasa `CallBack` implementuje interfejs `MqttCallBack`. Wymagana jest jedna instancja wywołania zwrotnego, na identyfikator klienta. W tym przykładzie konstruktor przekazuje identyfikator klienta, który ma zostać zapisany jako dane instancji. Jest on używany w wywołaniu zwrotnym w celu określenia, która instancja wywołania zwrotnego została uruchomiona.
   - W klasie zwrotnej należy zaimplementować trzy metody:

     `public void messageArrived(MqttTopic topic, MqttMessage message)`
     Otrzymuje publikację, która została zasubskrybowana.

     `public void connectionLost(Throwable cause)`
     Wywoływana, gdy połączenie zostanie utracone.

public void deliveryComplete(MqttDeliveryToken token))
Wywoływana po odebraniu znacznika dostarczania dla komunikatu QoS 1 lub 2, który został opublikowany.

- Wywołanie zwrotne jest aktywowane przez produkt `MqttClient.connect`.

4. Rozłącz klienta
   a) Usuń instrukcję zawierającą wyrażenie `token.waitForCompletion` .

   Wątek główny jest kontynuowany bez oczekiwania na dostarczenie publikacji.

   b) Sprawdź, czy klient jest już odłączony.

   Klient MQTT rozłącza się po wystąpieniu błędu zwróconego do metody `lostConnection` w produkcie `MqttCallback`klub aplikacja kliencka może odłączyć się od tej metody. Sprawdź, czy istnieje otwarte połączenie.

   c) Należy użyć stałej `Example.quiesceTimeout`, aby ustawić maksymalny czas wyciszenia klienta.

```
if (client.isConnected())
    client.disconnect(Example.quiesceTimeout);
```

Klient kończy działanie w przypadku, gdy połączenie następujących trzech warunków jest prawdziwe:

a. Wywołanie zwrotne zostało wywołane dla wszystkich komunikatów, które zostały opublikowane w tej sesji, lub jeśli sesja została zrestartowana, w poprzednich sesjach.

b. Komunikaty są w trakcie lotu, a interwał wyciszania utracił ważność. Domyślnie odstęp czasu wygaszania wynosi 30 sekund. Można zmienić limit czasu wygaszania, przekazując liczbę milisekund oczekiwania na wartość parametru `client.disconnect`.

c. Program `client.disconnect` został wywołany po opublikowaniu niektórych komunikatów i umieszczonym w kolejce przez klienta, ale przed przesłaniem komunikatów. Komunikaty w kolejce nie są jeszcze w trakcie lotu. Jeśli sesja jest restartowana, komunikaty są ponownie wysyłane po restarcie sesji.

Wyniki

Aby wyświetlić publikacje za pomocą produktu WebSphere MQ, należy utworzyć temat, kolejkę i trwałą subskrypcję, a wszystko to wywołanie funkcji `"MQTTExampleTopic"` przy użyciu skryptu w produkcie Rysunek 90 na stronie 495. Uruchom klienta, aby opublikować w temacie `MQTT Examples` , a następnie uruchom program przykładowy **amqsbcg** , aby przeglądać publikacje w kolejce produktu `MQTTExamples` .

1. Uruchom menedżer kolejek i uruchom jego usługę telemetryczną (MQXR). Upewnij się, że adres TCP/IP i port skonfigurowany dla kanału telemetrycznego są zgodne z wartościami używannymi w aplikacji MQTT.
2. Skonfiguruj trwałą subskrypcję, tworząc skrypt komend `mqttexamples.txt` i uruchamiając go za pomocą programu **runmqsc:** .

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Rysunek 90. mqttExampleTopic.txt

Aby uruchomić skrypt w systemie Windows, wpisz komendę:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Uruchom klienta jako aplikację Java z poziomu środowiska Eclipselub za pomocą środowiska Java w oknie komend:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Uwaga: Okno komend musi być otwarte w katalogu, w którym znajduje się ścieżka `com\ibm\mq\id`.

4. Przejrzyj wyniki za pomocą programu WebSphere MQ Explorer lub uruchom następującą komendę:

```
amqsbcg MQTTExampleQueue queue manager name
```

Przykładowy kod

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubAsync {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      CallBack callback = new CallBack(Example.clientId);
      client.setCallback(callback);
      client.connect();
      System.out.println("Publishing \"" + message.toString()
          + "\" on topic \"" + topic.getName() + "\" with QoS = "
          + message.getQos());
      System.out.println("For client instance \"" + client.getClientId()
          + "\" on address " + client.getServerURI() + "\"");
      MqttDeliveryToken token = topic.publish(message);
      System.out.println("With delivery token \"" + token.hashCode()
          + " delivered: " + token.isComplete());
      if (client.isConnected())
          client.disconnect(Example.quiesceTimeout);
      System.out.println("Disconnected: delivery token \"" + token.hashCode()
          + "\" received: " + token.isComplete());
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Rysunek 91. PubAsync.java

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class CallBack implements MqttCallback {
  private String instanceData = "";
  public CallBack(String instance) {
    instanceData = instance;
  }
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\" for instance \""
          + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void connectionLost(Throwable cause) {
    System.out.println("Connection lost on instance \"" + instanceData
        + "\" with cause \"" + cause.getMessage() + "\" Reason code "
        + ((MqttException)cause).getReasonCode() + "\" Cause \""
        + ((MqttException)cause).getCause() +  "\"");
    cause.printStackTrace();
  }
  public void deliveryComplete(MqttDeliveryToken token) {
    try {
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" received by instance \"" + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Rysunek 92. CallBack.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Rysunek 93. Example.java

Tworzenie odtwarzalnego asynchronicznego publikatora dla produktu MQ Telemetry Transport przy użyciu języka Java

W ramach tego zadania użytkownik śledzi kurs w celu zmodyfikowania aplikacji publikatora asynchronicznego. Modyfikacje umożliwiają aplikacji zakończenie dostarczania publikacji, które nie zostały potwierdzone podczas ostatniego uruchomienia klienta.

Zanim rozpoczniesz

Zainstaluj składnik WebSphere MQ Telemetry na serwerze, na którym jest zainstalowany produkt IBM WebSphere MQ Version 7.1 lub nowszy.

Aplikacja kliencka korzysta z pakietu `com.ibm.mq.micro.client.mqttv3` w pakiecie IBM WebSphere MQ Telemetry Software Development Toolkit (SDK). Pakiet SDK jest częścią instalacji produktu IBM WebSphere MQ Telemetry . Klient łączy się z funkcją IBM WebSphere MQ Telemetry w celu wymiany komunikatów z produktem IBM WebSphere MQ.

Aby administrować produktem IBM WebSphere MQ Telemetry, należy również zainstalować aktualizacje telemetryczne dla produktu IBM WebSphere MQ Explorer Version 7.1 . Aktualizacje są częścią instalacji produktu IBM WebSphere MQ Telemetry .

Klient MQTT, działający w środowisku Java SE, wymaga wersji 6.0 języka Java SE lub nowszej. IBM Java SE v6.0 jest częścią instalacji produktu IBM WebSphere MQ Version 7.1 . Znajduje się on w katalogu *WebSphere MQ installation directory*\java\jre .

O tym zadaniu

Przykładem jest aplikacja publikowania, PubAsyncRestartable. Produkt PubAsyncRestartable publikuje Hello World w temacie MQTT Examples, bez czekania na potwierdzenie, że publikacja została dostarczona do menedżera kolejek. Potwierdzenie dostarczenia jest odbierane w klasie wywołania zwrotnego CallBack. Wszystkie znaczniki dostarczania dla publikacji, które nie zostały zakończone w poprzedniej instancji, mogą zostać sprawdzone. Są one również przetwarzane przez klasę wywołania zwrotnego.

Ustawiając trwałą subskrypcję programu MQTT Examples , można sprawdzić, czy aplikacja działa.

W ramach tej procedury stosowana jest Eclipse do tworzenia, budowania i uruchamiania klienta. Produkt Eclipse można pobrać z serwisu WWW projektu Eclipse pod adresem www.eclipse.org.

Kroki opisane w sekcji Procedura modyfikują aplikację PubAsync.java w produkcie “Tworzenie asynchronicznego publikatora dla produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 493.

Alternatywnie można skopiować kod “Przykładowy kod” na stronie 502do nowego katalogu .\com\ibm\mq\id. Utwórz trzy pliki Java, Example.java, CallBack.javai PubAsyncRestartable.java. Skompiluj przykłady za pomocą komendy,

```
javac -cp jar_dir\com.mq.micro.client.mqttv3.jar
       com.ibm.mq.id.PubAsyncRestartable.java com.ibm.mq.id.CallBack.java
com.ibm.mq.id.Example.java
```

Uruchom program PubAsyncRestartable przy użyciu komendy,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubAsyncRestartable.class
```

Procedura

1. W pakiecie com.ibm.mq.id kliknij prawym przyciskiem myszy PubAsync.java i skopiuj go. Wklej go do tego samego pakietu, zmieniając jego nazwę na PubAsyncRestartable.
2. Utwórz identyfikator klienta wielokrotnego użytku.

   ```
   Example.clientId = String.format(
       "%-23.23s",
       (System.getProperty("user.name") + "_" + (System.getProperty(
           "clientId", "PubAsyncRestartable."))).trim()).replace('-', '_');
   ```

   Rysunek 94. Identyfikator klienta wielokrotnego użytku

   Aplikacje w produkcie “Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 488 i “Tworzenie asynchronicznego publikatora dla produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 493 korzystały z nowego identyfikatora klienta dla każdego połączenia klienckiego. W przypadku wznawialnego publikatora lub subskrybenta należy za każdym razem używać tego samego identyfikatora klienta, ale różne klienty muszą używać różnych identyfikatorów. Patrz “Identyfikator klienta” na stronie 543. Identyfikator klienta wielokrotnego użytku jest tworzony na podstawie nazwy użytkownika i nazwy klasy. Jego długość jest ograniczona do 23 bajtów. Muszą mieć tylko znaki, które są poprawne w nazwach obiektów menedżera kolejek. Kod usuwa wszystkie łączniki, które mogły zostać wstawione.

3. Wartość QoS komunikatu jest ustawiona na wartość 2, a nie wartość domyślna (1), aby uniknąć duplikowania komunikatów.

```
message.setQos(Example.QoS);
```

   Należy albo zmienić wartość parametru `Example.QoS` na 2, albo przekazać właściwość QoS jako argument za pomocą opcji `-DQoS=2` w wierszu komend Java.
4. Utwórz obiekt `MqttConnectOptions` i ustaw jego atrybut `cleanSession` na wartość `false`.
   a) Utwórz obiekt `MqttConnectOptions` .

```
MqttConnectOptions conOptions = new MqttConnectOptions();
```

      `conOptions` jest parametrem opcji w konstruktorze `MqttClient` .
   b) Ustaw atrybut clearSession .

```
conOptions.setCleanSession(Example.cleanSession);
```

      Domyślnie parametr `Example.cleanSession` jest ustawiony na wartość `true`, co jest zgodne z domyślnym ustawieniem `MqttConnectionOptions.cleanSession`.

   Po zrestartowaniu programu `PubAsyncRestartable` może on rozpoczynać się od "czystej sesji", a następnie wyczyścić wszystkie oczekujące tokeny dostarczania dla komunikatów o wartości QoS 1 lub 2.

   Ustaw wartość `Example.cleanSession` na `false` , aby zachować wszystkie oczekujące tokeny dostarczania. Tokeny są przetwarzane przez klasę `MqttCallBack` , gdy klient jest ponownie połączony.
5. Jeśli sesja jest restartowana, pobierz wszystkie oczekujące tokeny dostarczania i wydrukuj ich zawartość.

```
if (!conOptions.isCleanSession()) {
        MqttDeliveryToken tokens[] = client.getPendingDeliveryTokens();
        System.out.println("Starting a previous session for instance \""
            + client.getClientId() + "\" with " + tokens.length
            + " delivery tokens pending");
        for (int i = 0; i < tokens.length; i++) {
          System.out.println("Message \"" + tokens[i].getMessage().toString()
              + "\" with QoS=" + tokens[i].getMessage().getQos()
              + " recovered by instance \"" + client.getClientId()
              + "\" and assigned delivery token \"" + tokens[i].hashCode()
              + "\"");
        }
      } else
        System.out.println("Starting a clean session for instance "
            + client.getClientId());
```

6. Przekaz parametr `conOptions` do konstruktora `MqttClient` .

```
client.connect(conOptions);
```

7. W przypadku rozłączania należy ustawić maksymalny przedział czasu odłączenia.

```
client.disconnect(Example.timeout);
```

   Aby możliwe było wyświetlanie oczekujących tokenów dostarczania będących w trakcie przetwarzania, poprzednia instancja musi kończyć się bez zakończenia dostarczania. Aby uruchomić przykład z możliwością niepotwierdzania publikacji przed zakończeniem `PubAsyncRestartable` , należy ustawić wartość `Example.timeout` na 0.

Wyniki

Aby wyświetlić publikacje za pomocą produktu WebSphere MQ, należy utworzyć temat, kolejkę i trwałą subskrypcję, a wszystko to wywołanie funkcji `"MQTTExampleTopic"` przy użyciu skryptu w produkcie

Rysunek 95 na stronie 501. Uruchom klienta, aby opublikować w temacie MQTT Examples , a następnie uruchom program przykładowy **amqsbcg** , aby przeglądać publikacje w kolejce produktu MQTTExamples .

1. Uruchom menedżer kolejek i uruchom jego usługę telemetryczną (MQXR). Upewnij się, że adres TCP/IP i port skonfigurowany dla kanału telemetrycznego są zgodne z wartościami używannymi w aplikacji MQTT.
2. Skonfiguruj trwałą subskrypcję, tworząc skrypt komend `mqttexamples.txt` i uruchamiając go za pomocą programu **runmqsc:** .

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Rysunek 95. mqttExampleTopic.txt

Aby uruchomić skrypt w systemie Windows, wpisz komendę:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Uruchom klienta jako aplikację Java z poziomu środowiska Eclipselub za pomocą środowiska Java w oknie komend:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Uwaga: Okno komend musi być otwarte w katalogu, w którym znajduje się ścieżka com\ibm\mq\id.

4. Przejrzyj wyniki za pomocą programu WebSphere MQ Explorer lub uruchom następującą komendę:

```
amqsbcg MQTTExampleQueue queue manager name
```

Przykładowy kod

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.MqttClient;
import com.ibm.micro.client.mqttv3.MqttConnectOptions;
import com.ibm.micro.client.mqttv3.MqttDeliveryToken;
import com.ibm.micro.client.mqttv3.MqttMessage;
import com.ibm.micro.client.mqttv3.MqttTopic;
public class PubAsyncRestartable {
  public static void main(String[] args) {
    Example.clientId = String.format(
        "%-23.23s",
        (System.getProperty("user.name") + "_" + (System.getProperty(
            "clientId", "PubAsyncRestartable."))).trim()).replace('-', '_');
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      CallBack callback = new CallBack(Example.clientId);
      client.setCallback(callback);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setCleanSession(Example.cleanSession);
      if (!conOptions.isCleanSession()) {
        MqttDeliveryToken tokens[] = client.getPendingDeliveryTokens();
        System.out.println("Starting a previous session for instance \""
            + client.getClientId() + "\" with " + tokens.length
            + " delivery tokens pending");
        for (int i = 0; i < tokens.length; i++) {
          System.out.println("Message \"" + tokens[i].getMessage().toString()
              + "\" with QoS=" + tokens[i].getMessage().getQos()
              + " recovered by instance \"" + client.getClientId()
              + "\" and assigned delivery token \"" + tokens[i].hashCode()
              + "\"");
        }
      } else
        System.out.println("Starting a clean session for instance \""
            + client.getClientId() + "\"");
      client.connect(conOptions);
      System.out.println("Publishing \"" + message.toString()
          + "\" on topic \"" + topic.getName() + "\" with QoS = "
          + message.getQos());
      System.out.println("For client instance \"" + client.getClientId()
          + "\" on address " + client.getServerURI() + "\"");
      MqttDeliveryToken token = topic.publish(message);
      System.out.println("With delivery token \"" + token.hashCode()
          + " delivered: " + token.isComplete());
      if (client.isConnected())
          client.disconnect(Example.quiesceTimeout);
      System.out.println("Disconnected: delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Rysunek 96. PubAsyncRestartable.java

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class CallBack implements MqttCallback {
  private String instanceData = "";
  public CallBack(String instance) {
    instanceData = instance;
  }
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\" for instance \""
          + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void connectionLost(Throwable cause) {
    System.out.println("Connection lost on instance \"" + instanceData
        + "\" with cause \"" + cause.getMessage() + "\" Reason code "
        + ((MqttException)cause).getReasonCode() + "\" Cause \""
        + ((MqttException)cause).getCause() +  "\"");
    cause.printStackTrace();
  }
  public void deliveryComplete(MqttDeliveryToken token) {
    try {
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" received by instance \"" + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Rysunek 97. CallBack.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Rysunek 98. Example.java

Tworzenie subskrybenta dla produktu MQ Telemetry Transport przy użyciu języka Java

W ramach tego zadania użytkownik śledzi kurs w celu utworzenia aplikacji subskrybenta. Subskrybent tworzy subskrypcję tematu i otrzymuje publikacje dotyczące subskrypcji.

Zanim rozpoczniesz

Zainstaluj składnik WebSphere MQ Telemetry na serwerze, na którym jest zainstalowany produkt IBM WebSphere MQ Version 7.1 lub nowszy.

Aplikacja kliencka korzysta z pakietu `com.ibm.mq.micro.client.mqttv3` w pakiecie IBM WebSphere MQ Telemetry Software Development Toolkit (SDK). Pakiet SDK jest częścią instalacji produktu IBM WebSphere MQ Telemetry . Klient łączy się z funkcją IBM WebSphere MQ Telemetry w celu wymiany komunikatów z produktem IBM WebSphere MQ.

Aby administrować produktem IBM WebSphere MQ Telemetry, należy również zainstalować aktualizacje telemetryczne dla produktu IBM WebSphere MQ Explorer Version 7.1 . Aktualizacje są częścią instalacji produktu IBM WebSphere MQ Telemetry .

Klient MQTT, działający w środowisku Java SE, wymaga wersji 6.0 języka Java SE lub nowszej. IBM Java SE v6.0 jest częścią instalacji produktu IBM WebSphere MQ Version 7.1 . Znajduje się on w katalogu *WebSphere MQ installation directory*\java\jre .

O tym zadaniu

Przykładem jest aplikacja subskrybenta, Subscribe. Produkt Subscribe tworzy temat subskrypcji, MQTT Examplesi oczekuje na publikacje w subskrypcji przez 30 sekund.

Subskrybent może utworzyć subskrypcję i czekać na publikacje. Może również odbierać publikacje wysłane wcześniej do subskrypcji utworzonej wcześniej dla tego samego identyfikatora klienta. Atrybut boolowski MqttConnectionOptions.cleanSession określa, czy publikacje wysłane wcześniej są odbierane, czy też nie. Patrz “Subskrypcje” na stronie 554.

Za pomocą programów przykładowych publikowania można tworzyć publikacje lub korzystać z programu WebSphere MQ Explorer w celu utworzenia testowej publikacji w temacie MQTT Examples .

W ramach tej procedury stosowana jest Eclipse do tworzenia, budowania i uruchamiania klienta. Produkt Eclipse można pobrać z serwisu WWW projektu Eclipse pod adresem www.eclipse.org.

W instrukcjach w sekcji Procedura założono, że pakiet com.ibm.mq.id został już utworzony w jednym z wcześniejszych zadań, a następnie skopiowany w klasach Example.java i Callback.java .

Procedura

1. Utwórz klasę Subscribe w pakiecie com.ibm.mq.id.
2. Utwórz identyfikator klienta wielokrotnego użytku.

```
Example.clientId = String.format(
    "%-23.23s",
    (System.getProperty("user.name") + "_" + (System.getProperty(
        "clientId", "Subscribe."))).trim()).replace('-', '_');
```

Rysunek 99. Identyfikator klienta wielokrotnego użytku

Aplikacje w produkcie “Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 488 i “Tworzenie asynchronicznego publikatora dla produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 493 korzystały z nowego identyfikatora klienta dla każdego połączenia klienckiego. W przypadku wznawialnego publikatora lub subskrybenta należy za każdym razem używać tego samego identyfikatora klienta, ale różne klienty muszą używać różnych identyfikatorów. Patrz “Identyfikator klienta” na stronie 543. Identyfikator klienta wielokrotnego użytku jest tworzony na podstawie nazwy użytkownika i nazwy klasy. Jego długość jest ograniczona do 23 bajtów. Muszą mieć tylko znaki, które są poprawne w nazwach obiektów menedżera kolejek. Kod usuwa wszystkie łączniki, które mogły zostać wstawione.

3. Utwórz blok try-catch .

```
try { ...
 } catch (Exception e) {
   e.printStackTrace();
}
```

Klient MQTT zgłasza MqttException, MqttPersistenceException lub MqttSecurityException. MqttPersistenceException i MqttSecurityException są podklasami produktu MqttException.

Użyj metody MqttException.getReasonCode , aby dowiedzieć się, dlaczego wystąpił wyjątek. Jeśli zostanie zgłoszony wyjątek MqttPersistenceException lub MqttSecurityException , należy użyć metody getCause , aby zwrócić bazowy wyjątek throwable.

4. Utwórz nową instancję produktu MqttClient .

```
MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
```

Należy udostępnić klientowi adres serwera, który jest używany później w celu nawiązania połączenia z produktem WebSphere MQ. Ustaw identyfikator klienta, aby nazwać klienta.

- Opcjonalnie można udostępnić implementację interfejsu `MqttClientPersistence` w celu zastąpienia implementacji domyślnej. Domyślna implementacja produktu `MqttPersistence` przechowuje komunikaty QoS 1 i 2 oczekujące na dostarczenie jako pliki; patrz sekcja "Trwałość komunikatu w klientach MQTT" na stronie 547.
- Domyślny port TCP/IP IBM WebSphere MQ dla MQTT to 1883. Dla SSL jest to 8883. W tym przykładzie adres domyślny jest ustawiony na wartość `tcp://localhost:1883`.
- Zwykle ważne jest, aby być w stanie zidentyfikować konkretnego klienta fizycznego przy użyciu identyfikatora klienta. Identyfikator klienta musi być unikalny dla wszystkich klientów łączy się z serwerem; patrz "Identyfikator klienta" na stronie 543. Użycie tego samego identyfikatora klienta co w poprzedniej instancji wskazuje, że obecna instancja jest instancją tego samego klienta. Jeśli identyfikator klienta zostanie zduplikowany w dwóch uruchomionych klientach, w obu klientach zostanie zgłoszony wyjątek, a jeden klient kończy działanie.
- Długość identyfikatora klienta jest ograniczona do 23 bajtów. Jeśli długość zostanie przekroczona, zgłaszany jest wyjątek. Identyfikator klienta musi zawierać tylko znaki dozwolone w nazwie menedżera kolejek, na przykład brak łączników lub spacji.
- Dopóki nie zostanie wywołana metoda `MqttClient.connect` , przetwarzanie komunikatów nie jest wymagane.

Za pomocą obiektu klienta można publikować i subskrybować tematy oraz odzyskiwać informacje o publikacjach, które nie zostały jeszcze dostarczone.

5. Tuż przed wierszem `client.connect();` kodu, utwórz instancję klasy `CallBack` , przekazując identyfikator klienta.

```
CallBack callback = new CallBack(Example.clientId);
client.setCallback(callback);
```

- Klasa `CallBack` implementuje interfejs `MqttCallBack`. Wymagana jest jedna instancja wywołania zwrotnego, na identyfikator klienta. W tym przykładzie konstruktor przekazuje identyfikator klienta, który ma zostać zapisany jako dane instancji. Jest on używany w wywołaniu zwrotnym w celu określenia, która instancja wywołania zwrotnego została uruchomiona.
- W klasie zwrotnej należy zaimplementować trzy metody:

  `public void messageArrived(MqttTopic topic, MqttMessage message)`
  Otrzymuje publikację, która została zasubskrybowana.

  `public void connectionLost(Throwable cause)`
  Wywoływana, gdy połączenie zostanie utracone.

  `public void deliveryComplete(MqttDeliveryToken token))`
  Wywoływana po odebraniu znacznika dostarczania dla komunikatu QoS 1 lub 2, który został opublikowany.
- Wywołanie zwrotne jest aktywowane przez produkt `MqttClient.connect`.

6. Utwórz obiekt `MqttConnectOptions` i ustaw jego atrybut `cleanSession` .
   a) Utwórz obiekt `MqttConnectOptions` .

```
MqttConnectOptions conOptions = new MqttConnectOptions();
```

   `conOptions` jest parametrem opcji w konstruktorze `MqttClient` .

   b) Ustaw atrybut clearSession .

```
conOptions.setCleanSession(Example.cleanSession);
```

   Domyślnie parametr `Example.cleanSession` jest ustawiony na wartość `true`, co jest zgodne z domyślnym ustawieniem `MqttConnectionOptions.cleanSession`.

W przypadku korzystania z domyślnej klasy `MqttConnectOptions` lub w przypadku ustawienia wartości `true` na potrzeby atrybutu `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia z klientem wszystkie stare subskrypcje klienta zostaną usunięte po nawiązaniu połączenia z klientem. Wszystkie nowe subskrypcje dokonane przez klienta podczas sesji zostaną usunięte po rozłączeniu.

W przypadku ustawienia wartości `false` na potrzeby atrybutu `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia wszystkie subskrypcje tworzone przez klienta będą dodawane do wszystkich subskrypcji klienta, które istniały przed nawiązaniem połączenia. Wszystkie subskrypcje pozostaną aktywne po rozłączeniu połączenia z klientem.

Innym podejściem umożliwiającym zrozumienie wpływu atrybutu `cleanSession` na subskrypcje jest uznanie go za atrybut modalny. W domyślnym trybie atrybutu `cleanSession=true` klient tworzy subskrypcje i odbiera publikacje tylko w zasięgu sesji. W alternatywnym trybie atrybutu `cleanSession=false` subskrypcje są trwałe. Można nawiązywać połączenia z klientem i je rozłączać, a jego subskrypcje pozostają aktywne. Po ponownym nawiązaniu połączenia z klientem odbiera on wszystkie niedostarczone publikacje. Po nawiązaniu połączenia klient może modyfikować zestaw subskrypcji aktywnych na jego potrzeby.

Tryb atrybutu `cleanSession` należy ustawić przed nawiązaniem połączenia. Ten tryb będzie obowiązywać przez cały czas trwania sesji. Aby zmienić ustawienie trybu, należy rozłączyć połączenie z klientem, a następnie ponownie je nawiązać. W przypadku zmiany używanego trybu atrybutu z `cleanSession=false` na `cleanSession=true` wszystkie wcześniejsze subskrypcje klienta oraz wszystkie publikacje, które nie zostały odebrane, zostaną usunięte.

7. Przekaz parametr `conOptions` do konstruktora `MqttClient` .

```
client.connect(conOptions);
```

8. Utwórz subskrypcję.

```
       client.subscribe(Example.topicString, Example.QoS);
```

W tym przykładzie używana jest metoda `MqttClient.subscribe` , która przekazuje jeden filtr tematów za pomocą opcji `QoS` . Metoda `MqttClient.subscribe` ma cztery sygnatury i można przekazywać tablice filtrów subskrypcji, jak również jeden filtr.

W tym przykładzie używany jest łańcuch tematu używany przez przykłady publikowania jako filtr tematów, a więc otrzymuje wszystkie utworzone przez siebie publikacje.

Za każdym razem, gdy uruchamiasz przykład, `subscribe.java`, tworzy subskrypcję. Jeśli wartość Example.topicStringnie zostanie zmieniona, ponownie zostanie ponownie utworzona ta sama subskrypcja. Jeśli subskrypcja zostanie ponownie utworzona, nie spowoduje to utworzenia dwóch identycznych subskrypcji. Klient nie otrzymuje zduplikowanych kopii publikacji, które są zgodne z identyczną subskrypcją.

Subskrypcje są opisane w sekcji “Subskrypcje” na stronie 554, a filtry w produkcie “Łańcuchy tematów i filtry tematów w klientach MQTT” na stronie 556.

9. Poczekaj na nadejście niektórych publikacji, a następnie odłącz od klienta.

```
       Thread.sleep(Example.sleepTimeout);
       client.disconnect();
```

Publikacje są odbierane przez implementację metody `MqttCallback.messageArrived` .

Aplikacja subskrybuj nie opublikowała żadnych komunikatów, a więc nie czeka na żadne tokeny dostarczania. `client.disconnect` ma miejsce bez żadnych opóźnień.

Przykładowy kod

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.MqttClient;
import com.ibm.micro.client.mqttv3.MqttConnectOptions;
public class Subscribe {
  public static void main(String[] args) {
    Example.clientId = String.format(
        "%-23.23s",
        (System.getProperty("user.name") + "_" + System.getProperty("clientId",
            "Subscribe."))).trim();
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      CallBack callback = new CallBack(Example.clientId);
      client.setCallback(callback);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setCleanSession(Example.cleanSession);
      client.connect(conOptions);
      System.out.println("Subscribing to topic \"" + Example.topicString
          + "\" for client instance \"" + client.getClientId()
          + "\" using QoS " + Example.QoS + ". Clean session is "
          + Example.cleanSession);
      client.subscribe(Example.topicString, Example.QoS);
      System.out.println("Going to sleep for " + Example.sleepTimeout / 1000
          + " seconds");
      Thread.sleep(Example.sleepTimeout);
      client.disconnect();
      System.out.println("Finished");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Rysunek 100. Subscribe.java

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class CallBack implements MqttCallback {
  private String instanceData = "";
  public CallBack(String instance) {
    instanceData = instance;
  }
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\" for instance \""
          + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void connectionLost(Throwable cause) {
    System.out.println("Connection lost on instance \"" + instanceData
        + "\" with cause \"" + cause.getMessage() + "\" Reason code "
        + ((MqttException)cause).getReasonCode() + "\" Cause \""
        + ((MqttException)cause).getCause() +  "\"");
    cause.printStackTrace();
  }
  public void deliveryComplete(MqttDeliveryToken token) {
    try {
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" received by instance \"" + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Rysunek 101. CallBack.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Rysunek 102. Example.java

Pojęcia pokrewne

Aplikacje publikowania/subskrybowania MQTT

Uwierzytelnianie klienta Java MQTT za pomocą usługi JAAS

Dowiedz się, jak uwierzytelnić klienta za pomocą usługi JAAS. Zmodyfikuj przykładowy program `JAASLoginModule.java` i przykładowy program Java `PubSync.java`. Skonfiguruj kanał pomiarowy w taki sposób, aby wymagał uwierzytelniania JAAS , a następnie uruchom zmodyfikowany publikator, sprawdzając jego `nazwę użytkownika` i `hasło` przy użyciu usługi JAAS.

Zanim rozpoczniesz

Zakłada się, że przed wykonaniem tego zadania zostały zainstalowane pliki JAR klienta MQTT v3 , dokumentacja Javadoc, środowisko Eclipse, skonfigurowane kanały telemetryczne i kod PubSync.java . Istnieje obszar roboczy Eclipse , który zawiera działającą wersję pliku PubSync.java.

Zadanie jest napisane dla systemu Windows. Zmień ścieżki katalogów dla produktu Linux.

O tym zadaniu

Zadanie jest oparte na modyfikacji przykładowej klasy JAASLoginModule w programie *WMQ Installation directory*\mqxr\samples\JAASLoginModule.java w celu utworzenia MyLogin.java. W zadaniu modyfikujesz także przykładowy kod, PubSync.java w programie “Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 488 , aby ustawić nazwę użytkownika i hasło. W ramach testu program MyLogin.java losowo akceptuje lub odrzuca nazwę użytkownika i hasło.

Kroki w zadaniu są zapisywane jako ćwiczenie programistyczne. Konieczne jest dostosowanie procedury w celu przeprowadzenia rzeczywistego uwierzytelniania w środowisku produkcyjnym.

W typowym wyjaśnieniu, w jaki sposób należy programować uwierzytelnianie JAAS , zakłada się, że moduł logowania uwierzytelnia kontekst, który załadował JAAS. Gdy usługa telemetryczna (MQXR) wywołuje usługę JAAS, kontekst, który załadował usługę JAAS , jest usługą telemetryczną (MQXR). Nie istnieje punkt w uwierzytelnianiu kontekstu usługi telemetrycznej (MQXR); zawsze jest to mqm. Zamiast tego usługa telemetryczna (MQXR) konfiguruje klienta nazwa_użytkownika i hasło , aby były dostępne dla klasy modułu logowania. Wartości nazwa_użytkownika i hasło są przekazywane do modułu logowania za pomocą dwóch wywołań zwrotnych.

```
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);
callbackHandler.handle(callbacks);
String username =
      ((javax.security.auth.callback.NameCallback) callbacks[0]).getName();
char[] password =
      ((javax.security.auth.callback.PasswordCallback) callbacks[1]).getPassword();
```

Nazwa użytkownika i hasło klienta są jedynymi informacjami na temat klienta, który jest dostępny dla modułu logowania.

Procedura

1. Utwórz dwa pakiety, samples i security.jaas w tym samym projekcie Java, co PubSync.java.

   Pakiet samples jest używany tylko do celów referencyjnych. Wprowadź zmiany w kodzie w pakiecie security.jaas .
2. Zaimportuj JAASLoginModule.java i JAASPrincipal.java do obu pakietów.

   Jeśli to konieczne, refaktoryzuj instrukcje pakietu w źródle Java, aby wyeliminować błędy kompilacji.
3. Refaktoryzuj nazwę klasy, JAASLoginModule, w pakiecie security.jaas do MyLogin
4. W programie MyLogin.javanależy zastąpić część kodu w metodzie login , aby wyświetlić pracę modułu.
   a) Zastąp kod:

   ```
   // Accept everything.
   if (true)
     loggedIn = true;
   else
     throw new javax.security.auth.login.FailedLoginException("Login failed");
   ```

   b) Z kodem:

   ```
   // login half the users randomly
   PrintWriter pw = new PrintWriter(new FileWriter(System.getProperty("user.dir")
           + "\\MyLogin.log", true));
   pw.println("Called JAASLogin.login at "
           + System.getProperty("publication", "Hello World "
           + String.format("%tc", System.currentTimeMillis())));
   if (Math.random() < 0.5)
     loggedIn = true;
   pw.println("Username: \"" + username + "\", Password: \""
       + String.valueOf(password) + "\" loggedIn: " + loggedIn);
   ```

```
pw.close();
if (!loggedIn)
  throw new javax.security.auth.login.FailedLoginException("Login failed");
principal= new JAASPrincipal(username);
```

Kompletne źródło dla produktu `MyLogin.java` znajduje się w sekcji Rysunek 105 na stronie 513. Źródło produktu `JAASPrincipal.javaz` nazwą pakietu refaktoryzowanego do produktu `security.jaas` znajduje się w katalogu Rysunek 106 na stronie 514.

5. Ustaw ścieżkę klasy w programie `service.env` , tak aby wskazywała katalog zawierający ścieżkę do produktu `security/jaas/MyLogin.class` i `security/jaas/JAASPrincipal.class`.

```
CLASSPATH=C:\WMQTelemetryApps\MQTTSecureExamples\bin
```

Informacje na temat przekazywania `service.env` klasy do usługi produktu WebSphere MQ zawiera sekcja Konfigurowanie JAAS (JAAS) kanału telemetrycznego .

6. Dodaj sekcję modułu logowania do produktu `jaas.config`.

```
MyLoginExample {
  security.jaas.MyLogin required debug=true;
  };
```

Informacje na temat używania produktu `jaas.config` do definiowania modułu logowania JAAS zawiera sekcja Konfiguracja kanału JAAS kanału telemetrycznego .

7. Dodaj kanał pomiarowy przy użyciu kreatora **Nowy kanał pomiarowy** w programie WebSphere MQ Explorer, konfigurując kanał w taki sposób, aby wymagał uwierzytelniania JAAS . Zapoznaj się z sekcją `MyLoginExample` .

Na przykład można dostosować typ informacji wpisanych do kreatora z tej sekcji w pliku `mqxr_win.properties` . Jeśli pracujesz w Linux , plik nosi nazwę `mqxr_unix.properties`. Nie należy bezpośrednio edytować pliku właściwości telemetrycznych. W tym celu należy użyć kreatora.

```
com.ibm.mq.MQXR.channel/JAASMCAUser: \
com.ibm.mq.MQXR.Port=1884;\
com.ibm.mq.MQXR.JAASConfig=MyLoginExample;\
com.ibm.mq.MQXR.UserName=Admin;\
com.ibm.mq.MQXR.StartWithMQXRService=true
```

Uwaga: W przypadku zmodyfikowania dowolnego z parametrów kanału telemetrycznego lub zmodyfikowania klasy `security.jaas.Mylogin` należy zatrzymać i zrestartować usługę telemetryczną (MQXR). Dopiero po zrestartowaniu usługi wprowadzone zmiany zostaną uwzględnione.

8. Utwórz kopię produktu PubSync `.java` w pakiecie `com.ibm.mq.id` i nazwij kopię `PubSyncJAAS.java`.

Sekcja “Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 488 zawiera opis kroków tworzenia pliku PubSync.java w pakiecie produktu `com.ibm.mq.id` .

9. Ustaw `MqttConnectOptions.username` i `MqttConnectOptions.password` w programie `PubSyncJAAS.java` , a następnie przekaz `MqttConnectOptions` jako parametr `MqttClient.connect`.

```
MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setUserName(Example.username);
conOptions.setPassword(Example.password);
client.connect(conOptions);
```

Zapoznaj się z kurlityzowanym kodem w usłudze PubSyncJAAS.Java przy użyciu stałych ustawionych w pliku Example.java.

10. Ustaw wartość `Example.TCPAddress` na adres gniazda kanału pomiarowego, który został skonfigurowany do użycia konfiguracji JAAS `MyLoginExample`. Na przykład: 1884 jako numer portu.
11. Uruchom PubSyncJAAS kilka razy, aby zobaczyć, czy klient się loguje i czy został zaakceptowany lub odrzucony.

Za każdym razem, gdy próba logowania zostanie odrzucona, zgłaszany jest wyjątek.

Wyniki

W programie Rysunek 103 na stronie 512 wyświetlane są wyniki działania PubSyncJAAS.Java dwukrotnie. Rekordy dziennika są wyświetlane w programie Rysunek 104 na stronie 512.

```
Waiting for up to 10 seconds for publication of "Hello World Fri Jun 04 08:31:05 BST 2010" with
QoS = 1
On topic "MQTT Example" for client instance: "Admin_61c57a18_4bf7_40d" on address tcp://
localhost:1884"
With username "Admin" and password "Password"
Client exception caught
Client is not connected (32104)
    at
com.ibm.micro.client.mqttv3.internal.ExceptionHelper.createMqttException(ExceptionHelper.java:33
)
    at com.ibm.micro.client.mqttv3.internal.ClientComms.internalSend(ClientComms.java:88)
    at com.ibm.micro.client.mqttv3.internal.ClientComms.sendNoWait(ClientComms.java:105)
    at com.ibm.micro.client.mqttv3.MqttTopic.publish(MqttTopic.java:68)
    at com.ibm.mq.id.PubSync.main(PubSync.java:24)
```

```
Waiting for up to 10 seconds for publication of "Hello World Fri Jun 04 08:31:40 BST 2010" with
QoS = 1
On topic "MQTT Example" for client instance: "Admin_1d1599a0_50f5_4ea" on address tcp://
localhost:1884"
With username "Admin" and password "Password"
Delivery token "1731749688" has been received: true
```

Rysunek 103. Dane wyjściowe konsoli z programu PubSyncJAAS.java

Plik dziennika MyLogin.log jest przechowywany w produkcie *WMQ Data directory*, na przykład C:\IBM\MQ\Data\MyLogin.log:

```
Called JAASLogin.login at Hello World Fri Jun 04 08:31:05 BST 2010
Username: "Admin", Password: "Password" loggedIn: false
Called JAASLogin.login at Hello World Fri Jun 04 08:31:40 BST 2010
Username: "Admin", Password: "Password" loggedIn: true
```

Rysunek 104. MyLogin.log

Przykłady

Kursywą kodu w produkcie Rysunek 105 na stronie 513 jest modyfikacja przykładu JAASLoginModule.java.

```
package security.jaas;
import java.io.FileWriter;
import java.io.PrintWriter;

public class JAASLogin implements javax.security.auth.spi.LoginModule {
  private javax.security.auth.Subject                  subject;
  private javax.security.auth.callback.CallbackHandler callbackHandler;
  JAASPrincipal                                        principal;
  boolean                                              loggedIn = false;
  public void initialize(javax.security.auth.Subject subject,
      javax.security.auth.callback.CallbackHandler callbackHandler,
      java.util.Map<String, ?> sharedState, java.util.Map<String, ?> options) {
    this.subject = subject;
    this.callbackHandler = callbackHandler;
  }
  public boolean login() throws javax.security.auth.login.LoginException {
    try {
      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);

      callbackHandler.handle(callbacks);
      String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
          .getName();
      char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
          .getPassword();
      // login half the users randomly
      PrintWriter pw = new PrintWriter(new FileWriter(System
          .getProperty("user.dir")
          + "\\mylogin.log", true));
      pw.println("Called JAASLogin.login at "
          + System.getProperty("publication", "Hello World "
              + String.format("%tc", System.currentTimeMillis())));
      if (Math.random() < 0.5)
        loggedIn = true;
      pw.println("Username: \"" + username + "\", Password: \""
          + String.valueOf(password) + "\" loggedIn: " + loggedIn);
      pw.close();
      if (!loggedIn)
        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;
  }
  public boolean abort() throws javax.security.auth.login.LoginException {
    logout();
    return true;
  }
  public boolean commit() throws javax.security.auth.login.LoginException {
    if (loggedIn) {
      if (!subject.getPrincipals().contains(principal))
        subject.getPrincipals().add(principal);
    }
    return true;
  }
  public boolean logout() throws javax.security.auth.login.LoginException {
    subject.getPrincipals().remove(principal);
    principal = null;
    loggedIn = false;
    return true;
  }
}
```

Rysunek 105. MyLogin.java

Rysunek 106 na stronie 514 jest przykładowym kodem JAASLoginPrincipal.java, kopiowanym do pakietu security.jaas. Celem produktu JAASLoginPrincipal jest zaimplementowanie interfejsu

`java.security.Principal` w celu zachowania rekordu użytkowników, którzy zostali pomyślnie zalogowani przez produkt `MyLogin`.

```
package security.jaas;
public class JAASPrincipal implements java.security.Principal,
    java.io.Serializable {
  private static final long serialVersionUID = 1L;
  String                    name;
  public JAASPrincipal(String name) {
    this.name = name;
  }
  public String getName() {
    return name;
  }
  public String toString() {
    return (name);
  }
  public boolean equals(Object object) {
    if (object != null && object instanceof JAASPrincipal
        && name.equals(((JAASPrincipal) object).getName()))
      return true;
    else
      return false;
  }
  public int hashCode() {
    return name.hashCode();
  }
}
```

Rysunek 106. JAASLoginPrincipal.java

Kod w pliku PubSync.java , który jest modyfikowany w celu dodania `nazwy użytkownika` i `hasła` , jest kursywą w produkcie Rysunek 107 na stronie 514.

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubSyncSSL {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.SSLAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setUserName(Example.username);
      conOptions.setPassword(Example.password);
      client.connect(conOptions);
      System.out.println("Waiting for up to " + Example.sleepTimeout / 1000
              + " seconds for publication of \"" + message.toString()
              + "\" with QoS = " + message.getQos());
      System.out.println("On topic \"" + topic.getName() + "\" for client instance: \""
              + client.getClientId() + "\" on address " + client.getServerURI() + "\"");
      System.out.println("With username \"" + conOptions.getUserName()
              + "\" and password \"" + String.valueOf(conOptions.getPassword()) + "\"");
      MqttDeliveryToken token = topic.publish(message);
      token.waitForCompletion(Example.sleepTimeout);
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
      client.disconnect();
    } catch (Exception e) {
      System.out.println("Client exception caught");
      e.printStackTrace();
    }
  }
}
```

Rysunek 107. PubSyncJAAS.Java

Zmodyfikuj stałe w pliku Example.java w taki sposób, aby były zgodne z konfiguracją. Zignoruj ustawienia SSL dla tego przykładu.

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Rysunek 108. Example.java

Uwierzytelnianie połączenia telemetrycznego SSL przy użyciu samopodpisanych certyfikatów

Użyj samopodpisanych certyfikatów wygenerowanych przy użyciu produktu **Keytool** do uwierzytelniania połączenia SSL. Istnieje możliwość uwierzytelniania kanału pomiarowego lub kanału telemetrycznego oraz klientów dołączających do niego. Komunikaty przesyłane w połączeniu są szyfrowane.

Zanim rozpoczniesz

Przed uruchomieniem zadania “Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 488 należy wykonać zadanie PubSync.java pracujące z niezabezpieczonym połączeniem TCP/IP. W ramach tej czynności użytkownik modyfikuje PubSync.java , aby pracować z połączeniem SSL.

O tym zadaniu

Kroki w zadaniu są zapisywane jako ćwiczenie programistyczne. Konieczne jest dostosowanie procedury w celu przeprowadzenia rzeczywistego uwierzytelniania w środowisku produkcyjnym.

Zadanie jest napisane dla systemu Windows. Zmień ścieżki katalogów dla produktu Linux.

Procedura

1. Wykonaj zadanie “Modyfikowanie pliku PubSync.java w celu użycia protokołu SSL” na stronie 516, aby zmodyfikować plik PubSync.java w taki sposób, aby używany był protokół SSL.
2. Skonfiguruj kanał pomiarowy, a następnie utwórz magazyny kluczy w celu użycia protokołu SSL.

   Uwierzytelnij tylko kanał telemetryczny lub kanał i klienty, które łączą się z nim:

   - Wykonaj zadanie “Uwierzytelnianie kanału telemetrycznego” na stronie 517, aby połączyć się z protokołem SSL, uwierzytelniając kanał pomiarowy.
   - Wykonaj zadanie “Uwierzytelnianie kanału telemetrycznego i klientów” na stronie 518, aby połączyć się z protokołem SSL, uwierzytelniając kanał pomiarowy i klienty, które łączą się z nim.
3. Zatrzymaj i zrestartuj usługę telemetryczną (MQXR), aby odebrać zmiany w konfiguracjach kanałów telemetrycznych.
4. Uruchom program kliencki, aby sprawdzić, czy konfiguracja działa.

Modyfikowanie pliku PubSync.java w celu użycia protokołu SSL

Zmodyfikuj pierwszy przykład programu publikatora, aby połączyć się z kanałem pomiarowym przy użyciu protokołu SSL. Ustaw właściwości SSL używane przez zmodyfikowany program.

Zanim rozpoczniesz

Zakłada się, że przed wykonaniem tego zadania zostały zainstalowane pliki JAR klienta MQTT v3 , dokumentacja Javadoc, środowisko Eclipse, skonfigurowane kanały telemetryczne i kod PubSync.java . Istnieje obszar roboczy Eclipse , który zawiera działającą wersję pliku PubSync.java.

O tym zadaniu

Zadanie korzysta z klienta publikatora, PubSync.java, utworzonego w produkcie “Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 488 jako bazy danych. Do korzystania z protokołu SSL niezbędne są tylko małe modyfikacje; patrz Rysunek 109 na stronie 517 i Rysunek 110 na stronie 517.

Procedura

1. Utwórz kopię produktu PubSync.java w pakiecie com.ibm.mq.id i nazwij kopię PubSyncSSL.java.

   Sekcja “Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport przy użyciu języka Java” na stronie 488 zawiera opis kroków tworzenia pliku PubSync.java w pakiecie produktu com.ibm.mq.id .
2. Ustaw Example.SSLAddress na adres gniazda kanału telemetrycznego, który został skonfigurowany do użycia w konfiguracji SSL.
3. Zmień parametr adresu gniazda konstruktora klienta, aby był używany produkt Example.SSLAddress.

   ```
   MqttClient client = new MqttClient(Example.SSLAddress, Example.clientId);
   ```
4. Ustaw MqttConnectOptions.SSLProperties w PubSyncSSL.javai przekaz MqttConnectOptions jako parametr MqttClient.connect.

   ```
   MqttConnectOptions conOptions = new MqttConnectOptions();
   conOptions.setSSLProperties(Example.getSSLSettings());
   client.connect(conOptions);
   ```

   Zapoznaj się z kursywą kodu PubSyncSSL.Java , używając stałych ustawionych w Example.java.

Przykłady

Modyfikacje metody PubSync.java w celu dodania protokołu SSL są wyświetlane w języku Rysunek 109 na stronie 517 kursywą.

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubSyncSSL {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.SSLAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setSSLProperties(Example.getSSLSettings());
      client.connect(conOptions);
      System.out.println("Waiting for up to " + Example.sleepTimeout / 1000
              + " seconds for publication of \"" + message.toString()
              + "\" with QoS = " + message.getQos());
      System.out.println("On topic \"" + topic.getName() + "\" for client instance: \""
              + client.getClientId() + "\" on address " + client.getServerURI() + "\"");
      System.out.println("SSL Properties" + conOptions.getSSLProperties());
      MqttDeliveryToken token = topic.publish(message);
      token.waitForCompletion(Example.sleepTimeout);
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
      client.disconnect();
    } catch (Exception e) {
      System.out.println("Client exception caught");
      e.printStackTrace();
    }
  }
}
```

Rysunek 109. PubSyncSSL.Java

Modyfikacje wprowadzone w pliku Example.java są wyświetlane w produkcie Rysunek 110 na stronie 517.

```
  public static final String          SSLAddress =
     System.getProperty("SSLAddress", "ssl://localhost:8883");

  public static final Properties getSSLSettings() {
     final Properties properties = new Properties();
     properties.setProperty("com.ibm.ssl.keyStore", "C:\\Certificates\\SSClientKey.jks");
     properties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
     properties.setProperty("com.ibm.ssl.keyStorePassword", "password");
     properties.setProperty("com.ibm.ssl.trustStore", "C:\\Certificates\\SSClientTrust.jks");
     properties.setProperty("com.ibm.ssl.trustStoreType", "JKS");
     properties.setProperty("com.ibm.ssl.trustStorePassword", "password");
     return properties;
```

Rysunek 110. Modyfikacje produktu Example.java

Uwierzytelnianie kanału telemetrycznego

Klienty uwierzytelniają kanał pomiarowy w celu zaszyfrowania treści komunikatów przepływających przez kanał oraz w celu zapewnienia, że klient łączy się z poprawnym kanałem telemetryczny. Serwer nie uwierzytelnia klienta.

O tym zadaniu

Do tworzenia i zarządzania samopodpisanymi certyfikatami można użyć wielu edytorów magazynów kluczy. Zadanie korzysta z komendy wiersza komend **keytool** , która jest częścią środowiska JRE. Za pomocą narzędzia GUI **iKeyman**, który jest dostarczany razem z produktem WebSphere MQ , można przeglądać magazyny kluczy i generować klucze. Uruchom program **iKeyman** przy użyciu komendy **strmqikm**.

Procedura

1. Utwórz kanał pomiarowy, SSLSSOptClients , który wymaga połączenia SSL przy użyciu kreatora **Nowy kanał pomiarowy** . Kanał akceptuje anonimowych klientów.

   Dostosuj konfigurację kanału z następującej sekcji konfiguracji. Nie należy bezpośrednio edytować pliku właściwości telemetrycznych. W tym celu należy użyć kreatora.

   ```
   com.ibm.mq.MQXR.channel/SSLSSOptClients: \
   com.ibm.mq.MQXR.Port=8883;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\SSServerOptKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=OPTIONAL;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Wygeneruj klucze dla klienta w celu uwierzytelnienia kanału telemetrycznego.

   a) Wygeneruj samopodpisaną parę kluczy dla kanału telemetrycznego w nowym magazynie kluczy SSServerOptKey.jks:

   ```
   Keytool -genkey -noprompt -alias SSServerPrivate
           -dname "CN=mqttserver.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore SSServerOptKey.jks -storepass password -keypass password
   ```

   b) Wyeksportuj jego certyfikat publiczny jako plik ASCII za pomocą opcji -rfc :

   ```
   Keytool -export -noprompt -alias SSServerPrivate -file SSServerPublic.cer
           -keystore SSServerOptKey.jks -storepass password -rfc
   ```

   Jeśli czynność jest uruchamiana w systemie Windows, kliknij dwukrotnie opcję SSServerPublic.cer , aby sprawdzić jego zawartość.

   c) Zaimportuj certyfikat publiczny do nowego magazynu zaufanych certyfikatów klienta, SSClientTrust.jks:

   ```
   Keytool -import -noprompt -alias SSServerPublic -file SSServerPublic.cer
           -keystore SSClientTrust.jks -storepass password
   ```

   d) Utwórz pusty magazyn kluczy klienta SSClientKey.jks.

   Produkt **Keytool** nie ma komendy, aby utworzyć pusty magazyn kluczy. Dostępne są dwie opcje:

   i) Uruchom program **strmqikm**i utwórz magazyn kluczy SSClientKey.jks, ale nie dodaj żadnych kluczy.

   ii) Wykonaj krok 3a w sekcji “Uwierzytelnianie kanału telemetrycznego i klientów” na stronie 518, ale nie używaj jeszcze kluczy.

Uwierzytelnianie kanału telemetrycznego i klientów

Klienty uwierzytelniają kanał pomiarowy, a kanał pomiarowy uwierzytelnia klientów przyłączonych do niego. Komunikaty przesyłane w kanale są szyfrowane.

O tym zadaniu

Do tworzenia i zarządzania samopodpisanymi certyfikatami można użyć wielu edytorów magazynów kluczy. Zadanie korzysta z komendy wiersza komend **keytool** , która jest częścią środowiska JRE. Za pomocą narzędzia GUI **iKeyman**, który jest dostarczany razem z produktem WebSphere MQ , można przeglądać magazyny kluczy i generować klucze. Uruchom program **iKeyman** przy użyciu komendy **strmqikm**.

Kanał telemetryczny jest skonfigurowany z innym plikiem kluczy do zadania “Uwierzytelnianie kanału telemetrycznego” na stronie 517. Można użyć tego samego magazynu kluczy i pominąć krok “2” na stronie 519 w celu dodania kluczy do magazynu kluczy.

Procedura

1. Utwórz kanał pomiarowy, SSLSSReqClients , który wymaga połączenia SSL przy użyciu kreatora **Nowy kanał pomiarowy** . Kanał akceptuje tylko uwierzytelnionych klientów.

   Dostosuj konfigurację kanału z następującej sekcji konfiguracji:

   ```
   com.ibm.mq.MQXR.channel/SSLSSReqClients: \
   com.ibm.mq.MQXR.Port=8884;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\SSServerReqKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=REQUIRED;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Wygeneruj klucze dla klienta w celu uwierzytelnienia kanału telemetrycznego.

   a) Wygeneruj samopodpisaną parę kluczy dla kanału telemetrycznego w nowym magazynie kluczy SSServerReqKey.jks:

   ```
   Keytool -genkey -noprompt -alias SSServerPrivate
           -dname "CN=mqttserver.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore SSServerReqKey.jks -storepass password -keypass password
   ```

   b) Wyeksportuj jego certyfikat publiczny jako plik ASCII za pomocą opcji -rfc :

   ```
   Keytool -export -noprompt -alias SSServerPrivate -file SSServerPublic.cer
           -keystore SSServerReqKey.jks -storepass password -rfc
   ```

   Jeśli czynność jest uruchamiana w systemie Windows, kliknij dwukrotnie opcję SSServerPublic.cer , aby sprawdzić jego zawartość.

   c) Zaimportuj certyfikat publiczny do magazynu zaufanych certyfikatów nowego klienta, SSClientTrust.jks:

   ```
   Keytool -import -noprompt -alias SSServerPublic -file SSServerPublic.cer
           -keystore SSClientTrust.jks -storepass password
   ```

3. Wygeneruj klucze dla kanału telemetrycznego w celu uwierzytelnienia klienta.

   a) Wygeneruj samopodpisaną parę kluczy dla klienta w nowym magazynie kluczy, SSClientKey.jks:

   ```
   Keytool -genkey -noprompt -alias SSClientPrivate
           -dname "CN=mqttclient.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore SSClientKey.jks -storepass password -keypass password
   ```

   b) Wyeksportuj jego certyfikat publiczny jako plik ASCII za pomocą opcji -rfc :

   ```
   Keytool -export -noprompt -alias SSClientPrivate -file SSClientPublic.cer
           -keystore SSClientKey.jks -storepass password -rfc
   ```

   Jeśli czynność jest uruchamiana w systemie Windows, kliknij dwukrotnie opcję SSClientPublic.cer , aby sprawdzić jego zawartość.

   c) Zaimportuj certyfikat publiczny do magazynu kluczy serwera SSServerReqKey.jks:

   ```
   Keytool -import -noprompt -alias SSClientPublic -file SSClientPublic.cer
           -keystore SSServerReqKey.jks -storepass password
   ```

   Kanały telemetryczne korzystają z tego samego sklepu zarówno dla kluczy prywatnych, jak i zaufanych certyfikatów.

Uwierzytelnianie połączenia telemetrycznego SSL przy użyciu łańcucha certyfikatów

Użyj podpisanych certyfikatów uzyskanych z ośrodka certyfikacji lub z zaimplementowania własnej procedury certyfikacji, aby uwierzytelnić połączenie SSL. Istnieje możliwość uwierzytelniania kanału pomiarowego lub kanału telemetrycznego oraz klientów dołączających do niego. Komunikaty przesyłane w połączeniu są szyfrowane.

Zanim rozpoczniesz

Przed rozpoczęciem należy wykonać zadanie "Uwierzytelnianie połączenia telemetrycznego SSL przy użyciu samopodpisanych certyfikatów" na stronie 515 , aby program `PubSyncSSL.Java` pracował z zabezpieczonym połączeniem TCP/IP przy użyciu samopodpisanych certyfikatów.

O tym zadaniu

W tym zadaniu należy zmodyfikować zadania "Uwierzytelnianie kanału telemetrycznego" na stronie 517i "Uwierzytelnianie kanału telemetrycznego i klientów" na stronie 518 w programie "Uwierzytelnianie połączenia telemetrycznego SSL przy użyciu samopodpisanych certyfikatów" na stronie 515, aby pracować z kluczami certyfikowanymi przez łańcuch certyfikatów.

Certyfikaty dla tego zadania można uzyskać z ośrodka certyfikacji lub można skorzystać z serwisów WWW, takich jak http://www.openca.org/ , aby uzyskać certyfikaty. Komercyjne organy certyfikatowe zazwyczaj dostarczają próbne certyfikaty na krótki okres bez żadnych opłat. To zadanie zostało przetestowane przy użyciu komercyjnych certyfikatów uzyskanych.

Inną opcją jest budowanie własnego procesu certyfikacji i uruchamianie go na własnych komputerach, przy użyciu narzędzi ze stron internetowych, takich jak https://www.openssl.org/.

Magazyny zaufanych certyfikatów środowiska JRE `cacerts` nie są używane w tym zadaniu. Magazyn zaufanych certyfikatów środowiska JRE `cacerts` można używać na kliencie w zadaniu "Uwierzytelnianie kanału telemetrycznego" na stronie 520zamiast używać określonego magazynu zaufanych certyfikatów. Łańcuch certyfikatów może być podpisany przez dobrze znany ośrodek certyfikacji, który ma już certyfikat główny w sklepie `cacerts` na kliencie. W takim przypadku nie należy określać magazynu zaufanych certyfikatów na kliencie. Upewnij się, że na kliencie jest zainstalowanych wiele środowisk JRE, które zarządzają poprawnym sklepem `cacerts` .

Procedura

1. Jeśli czynność ta nie została jeszcze wykonana, należy wykonać czynność "Modyfikowanie pliku PubSync.java w celu użycia protokołu SSL" na stronie 516w celu zmodyfikowania pliku PubSync.java w taki sposób, aby używała protokołu SSL.
2. Skonfiguruj kanał pomiarowy, a następnie utwórz magazyny kluczy w celu użycia protokołu SSL.

   Uwierzytelnij tylko kanał telemetryczny lub kanał i klienty, które łączą się z nim:

   - Wykonaj zadanie "Uwierzytelnianie kanału telemetrycznego" na stronie 520, aby połączyć się z protokołem SSL, uwierzytelniając kanał pomiarowy.
   - Wykonaj zadanie "Uwierzytelnianie kanału telemetrycznego i klientów" na stronie 522, aby połączyć się z protokołem SSL, uwierzytelniając kanał pomiarowy i klienty, które łączą się z nim.
3. Zatrzymaj i zrestartuj usługę telemetryczną (MQXR), aby odebrać zmiany w konfiguracjach kanałów telemetrycznych.
4. Uruchom program kliencki, aby sprawdzić, czy konfiguracja działa.

Uwierzytelnianie kanału telemetrycznego

Klienty uwierzytelniają kanał pomiarowy w celu zaszyfrowania treści komunikatów przepływających przez kanał oraz zapewnienia, że klient łączy się z poprawnym kanałem telemetryczny. Serwer nie uwierzytelnia klienta.

O tym zadaniu

Do tworzenia i zarządzania certyfikatami można użyć wielu edytorów magazynów kluczy. Zadanie korzysta z komendy wiersza komend **keytool** , która jest częścią środowiska JRE. Za pomocą narzędzia GUI **iKeyman**, który jest dostarczany razem z produktem WebSphere MQ , można przeglądać magazyny kluczy i generować klucze. Uruchom program **iKeyman** przy użyciu komendy **strmqikm**.

Procedura

1. Utwórz kanał pomiarowy, SSLCAOptClients , który wymaga połączenia SSL przy użyciu kreatora **Nowy kanał pomiarowy** . Kanał akceptuje anonimowych klientów.

   Dostosuj konfigurację kanału z następującej sekcji konfiguracji. Nie należy bezpośrednio edytować pliku właściwości telemetrycznych. W tym celu należy użyć kreatora.

   ```
   com.ibm.mq.MQXR.channel/SSLCAOptClients: \
   com.ibm.mq.MQXR.Port=8885;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\CAServerOptKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=OPTIONAL;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Wygeneruj klucz podpisany przez ośrodek CA dla klienta w celu uwierzytelnienia kanału telemetrycznego.

   a) Wygeneruj samopodpisaną parę kluczy dla kanału telemetrycznego w nowym magazynie kluczy SSServerOptKey.jks:

   ```
   Keytool -genkey -noprompt -alias CAServerPrivate -keyalg RSA
           -dname "CN=mqttserverOpt.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore CAServerOptKey.jks -storepass password -keypass password
   ```

   Algorytm klucza jest ustawiony na wartość RSA, ponieważ niektóre uprawnienia do certyfikatów wymagają tego algorytmu. Nazwa zwykła certyfikatu musi być unikalna, niektóre uprawnienia do certyfikatów nie wydają kluczy o identycznych nazwach wspólnych.

   b) Tworzenie żądania podpisania certyfikatu (CSR) jako pliku ASCII

   ```
   Keytool -certreq -noprompt -alias CAServer -file CAServerOptKey.csr
           -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore CAServerOptKey.jks -storepass password -keypass password
   ```

   c) Uruchom oprogramowanie ośrodka certyfikacji lub zaloguj się do ich serwisu WWW. Wkleja zawartość pliku CAServerOptKey.csr , gdy zostanie wyświetlone pytanie o plik CSR.

   d) Ośrodek certyfikacji zwraca jeden lub dwa certyfikaty, a podpisany plik odpowiedzi jako pliki ASCII. Wklej zawartość do dwóch lub trzech plików:

   certyfikat główny
   Wklej do CARoot.cer

   certyfikat pośredni
   Wklej do CAInter.cer

   Plik odpowiedzi podpisany przez serwer
   Wklej do CAServerOpt.rsp

   Baza certyfikatów środowiska JRE nie jest używana w tym zadaniu. Jeśli odebrano jeden certyfikat główny i podpisaną odpowiedź z ośrodka CA, należy użyć certyfikatu głównego i podpisanej odpowiedzi w następujących krokach. Jeśli został odebrany certyfikat główny i certyfikat pośredni, należy użyć certyfikatu pośredniego i podpisanej odpowiedzi.

   e) Odbierz podpisaną odpowiedź serwera do magazynu kluczy serwera, z którego wysłano żądanie certyfikatu.

Otrzymywanie odpowiedzi modyfikuje samopodpisany certyfikat, tak aby był podpisany przez ośrodek CA. Jeśli zajrzyj do certyfikatu w magazynie kluczy przed i po odebraniu odpowiedzi, osoba podpisująca zmieni się. Jeśli nie, zostanie zgłoszony przez narzędzie do zarządzania kluczami. Przed użyciem certyfikatu należy go sprawdzić, a następnie sprawdzić, czy osoba podpisująca jest teraz ośrodkiem CA.

```
Keytool -import -noprompt -alias CAServer -file CAServerOpt.rsp
        -keystore CAServerOptKey.jks -storepass password
```

W niektórych kluczach oprogramowania do zarządzania, takich jak **iKeyman**, użytkownik otrzymuje pliki odpowiedzi, a nie pliki odpowiedzi.

f) Zaimportuj certyfikat CA do magazynu zaufanych certyfikatów klienta.

Zaimportuj certyfikat pośredni, jeśli odebrano dwa certyfikaty z ośrodka CA, lub certyfikat główny, jeśli odebrano tylko jeden certyfikat.

Albo:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAClientTrust.jks -storepass password
```

lub:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAClientTrust.jks -storepass password
```

Uwierzytelnianie kanału telemetrycznego i klientów

Klienty uwierzytelniają kanał pomiarowy, a kanał pomiarowy uwierzytelnia klientów przyłączonych do niego. Komunikaty przesyłane w kanale są szyfrowane.

O tym zadaniu

Do tworzenia i zarządzania certyfikatami można użyć wielu edytorów magazynów kluczy. Zadanie korzysta z komendy wiersza komend **keytool** , która jest częścią środowiska JRE. Za pomocą narzędzia GUI **iKeyman**, który jest dostarczany razem z produktem WebSphere MQ , można przeglądać magazyny kluczy i generować klucze. Uruchom program **iKeyman** przy użyciu komendy **strmqikm**.

Kanał pomiarowy jest skonfigurowany z innym plikiem kluczy do tego, który znajduje się w zadaniu, “Uwierzytelnianie kanału telemetrycznego” na stronie 520. Można użyć tego samego magazynu kluczy i pominąć krok “2” na stronie 522 w celu dodania kluczy do magazynu kluczy.

Procedura

1. Utwórz kanał pomiarowy, SSLCAReqClients , który wymaga połączenia SSL przy użyciu kreatora **Nowy kanał pomiarowy** . Kanał akceptuje tylko uwierzytelnionych klientów.

   Dostosuj konfigurację kanału z następującej sekcji konfiguracji. Nie należy bezpośrednio edytować pliku właściwości telemetrycznych. W tym celu należy użyć kreatora.

   ```
   com.ibm.mq.MQXR.channel/SSLCAReqClients: \
   com.ibm.mq.MQXR.Port=8886;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\CAServerReqKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=REQUIRED;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Wygeneruj klucz podpisany przez ośrodek CA dla klienta w celu uwierzytelnienia kanału telemetrycznego.

   a) Wygeneruj samopodpisaną parę kluczy dla kanału telemetrycznego w nowym magazynie kluczy CAServerReqKey.jks:

```
Keytool -genkey -noprompt -alias CAServerPrivate -keyalg RSA
        -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
        -keystore CAServerReqKey.jks -storepass password -keypass password
```

Algorytm klucza jest ustawiony na wartość RSA, ponieważ niektóre uprawnienia do certyfikatów wymagają tego algorytmu. Nazwa zwykła certyfikatu musi być unikalna, niektóre uprawnienia do certyfikatów nie wydają kluczy o identycznych nazwach wspólnych.

b) Tworzenie żądania podpisania certyfikatu (CSR) jako pliku ASCII

```
Keytool -certreq -noprompt -alias CAServer -file CAServerReqKey.csr
        -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
        -keystore CAServerReqKey.jks -storepass password -keypass password
```

c) Uruchom oprogramowanie ośrodka certyfikacji lub zaloguj się do ich serwisu WWW. Wkleja zawartość pliku `CAServerReqKey.csr` , gdy zostanie wyświetlone pytanie o plik CSR.

d) Ośrodek certyfikacji zwraca jeden lub dwa certyfikaty, a podpisany plik odpowiedzi jako pliki ASCII. Wklej zawartość do dwóch lub trzech plików:

certyfikat główny
Wklej do `CARoot.cer`

certyfikat pośredni
Wklej do `CAInter.cer`

Plik odpowiedzi podpisany przez serwer
Wklej do `CAServerReq.rsp`

Baza certyfikatów środowiska JRE nie jest używana w tym zadaniu. Jeśli odebrano jeden certyfikat główny i podpisaną odpowiedź z ośrodka CA, należy użyć certyfikatu głównego i podpisanej odpowiedzi w następujących krokach. Jeśli został odebrany certyfikat główny i certyfikat pośredni, należy użyć certyfikatu pośredniego i podpisanej odpowiedzi.

e) Odbierz podpisaną odpowiedź serwera do magazynu kluczy serwera, z którego wysłano żądanie certyfikatu.

Otrzymywanie odpowiedzi modyfikuje samopodpisany certyfikat, tak aby był podpisany przez ośrodek CA. Jeśli zajrzyj do certyfikatu w magazynie kluczy przed i po odebraniu odpowiedzi, osoba podpisująca zmieni się. Jeśli nie, a błąd jest zgłaszany przez narzędzie do zarządzania kluczami. Przed użyciem certyfikatu należy go sprawdzić, a następnie sprawdzić, czy osoba podpisująca jest teraz ośrodkiem CA.

```
Keytool -import -noprompt -alias CAServer -file CAServerReq.rsp
        -keystore CAServerReqKey.jks -storepass password
```

W niektórych kluczach oprogramowania do zarządzania, takich jak **iKeyman**, użytkownik otrzymuje pliki odpowiedzi, a nie pliki odpowiedzi.

f) Zaimportuj certyfikat CA do magazynu zaufanych certyfikatów klienta.

Zaimportuj certyfikat pośredni, jeśli odebrano dwa certyfikaty z ośrodka CA, lub certyfikat główny, jeśli odebrano tylko jeden certyfikat.

Albo:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAClientTrust.jks -storepass password
```

lub:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAClientTrust.jks -storepass password
```

3. Wygeneruj klucz podpisany przez ośrodek CA dla kanału telemetrycznego w celu uwierzytelnienia klientów.

a) Wygeneruj samopodpisaną parę kluczy dla klientów w nowym magazynie kluczy, `CAClientKey.jks`:

```
Keytool -genkey -noprompt -alias CAClientPrivate -keyalg RSA
        -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
        -keystore CAClientKey.jks -storepass password -keypass password
```

Algorytm klucza jest ustawiony na wartość RSA, ponieważ niektóre uprawnienia do certyfikatów wymagają tego algorytmu. Nazwa zwykła certyfikatu musi być unikalna, niektóre uprawnienia do certyfikatów nie wydają kluczy o identycznych nazwach wspólnych.

b) Tworzenie żądania podpisania certyfikatu (CSR) jako pliku ASCII

```
Keytool -certreq -noprompt -alias CAClient -file CAClientKey.csr
        -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
        -keystore CAClientKey.jks -storepass password -keypass password
```

c) Uruchom oprogramowanie ośrodka certyfikacji lub zaloguj się do ich serwisu WWW. Wkleja zawartość pliku `CAClientKey.csr` , gdy zostanie wyświetlone pytanie o plik CSR.

d) Ośrodek certyfikacji zwraca jeden lub dwa certyfikaty, a podpisany plik odpowiedzi jako pliki ASCII. Wklej zawartość do dwóch lub trzech plików:

certyfikat główny
Wklej do `CARoot.cer`

certyfikat pośredni
Wklej do `CAInter.cer`

Plik odpowiedzi podpisanej przez klienta
Wklej do `CAClient.rsp`

Baza certyfikatów środowiska JRE nie jest używana w tym zadaniu. Jeśli odebrano jeden certyfikat główny i podpisaną odpowiedź z ośrodka CA, należy użyć certyfikatu głównego i podpisanej odpowiedzi w następujących krokach. Jeśli został odebrany certyfikat główny i certyfikat pośredni, należy użyć certyfikatu pośredniego i podpisanej odpowiedzi.

e) Odbierz podpisaną odpowiedź klienta do magazynu kluczy klienta, z którego wystawiłeś żądanie certyfikatu.

Otrzymywanie odpowiedzi modyfikuje samopodpisany certyfikat, tak aby był podpisany przez ośrodek CA. Jeśli zajrzyj do certyfikatu w magazynie kluczy przed i po odebraniu odpowiedzi, osoba podpisująca zmieni się. Jeśli nie, a błąd jest zgłaszany przez narzędzie do zarządzania kluczami. Przed użyciem certyfikatu należy go sprawdzić, a następnie sprawdzić, czy osoba podpisująca jest teraz ośrodkiem CA.

```
Keytool -import -noprompt -alias CAClient -file CAClient.rsp
        -keystore CAClientKey.jks -storepass password
```

W niektórych kluczach oprogramowania do zarządzania, takich jak **iKeyman**, użytkownik otrzymuje pliki odpowiedzi, a nie pliki odpowiedzi.

f) Zaimportuj certyfikat CA do magazynu kluczy serwera.

Zaimportuj certyfikat pośredni, jeśli odebrano dwa certyfikaty z ośrodka CA, lub certyfikat główny, jeśli odebrano tylko jeden certyfikat.

Albo:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAServerReqKey.jks -storepass password
```

lub:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAServerReqKey.jks -storepass password
```

Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport za pomocą języka C

Kroki służące do tworzenia aplikacji publikatora klienta MQTT są opisane w sposób samouczek. Każdy wiersz kodu C jest wyjaśniony. Po zakończeniu zadania zostanie utworzony publikator MQTT.

Zanim rozpoczniesz

Aplikacja kliencka opracowała biblioteki klienta MQTT v3 C klienta MQTT. Aplikacja nawiązuje połączenie z demonem WebSphere MQ Telemetry dla urządzeń w celu publikowania komunikatów. W sekcji Tworzenie pierwszego publikatora znajduje się przykład klienta komuniującego się z produktem WebSphere MQ Telemetry.

O tym zadaniu

Przykładem jest aplikacja publikowania, `pubsync.c`. Program `pubsync.c` publikuje komunikat z ładunkiem `Hello World!` do tematu `MQTT Examplei` oczekuje na potwierdzenie, że publikacja została dostarczona do demona.

W przypadku uproszczenia kody powrotu z niektórych używanych funkcji nie są testowane pod kątem poprawnego zakończenia. Kody powrotu można sprawdzić w kodzie produkcyjnym, aby upewnić się, że program zachowuje się zgodnie z oczekiwaniami. Jeśli wystąpi nieoczekiwany błąd, należy podjąć odpowiednie działania.

Po ustawieniu subskrybenta na `MQTT Example` można sprawdzić, czy aplikacja działa.

Aby utworzyć, zbudować i uruchomić klienta, należy użyć wybranego środowiska programistycznego C. Jeśli wolisz, możesz skopiować kod bezpośrednio z przykładów.

Procedura

1. Utwórz nowy, pusty plik źródłowy, `pubsync.c`
2. Utwórz plik `settings.h`. Skopiuj kod na rysunku 2 do pliku.

   Wszystkie parametry używane w programie są zdefiniowane w programie `settings.h`. Wartości można przesłonić, zmieniając wartości w pliku.
3. Kroki, które należy wykonać, wyjaśniają kod. Postępuj zgodnie z instrukcjami lub skopiuj kod z Rysunek 1 do `pubsync.c`.
4. Dodaj do pliku nagłówkowego instrukcje dotyczące wymaganych bibliotek standardowych oraz plików `MQTTClient.h` i `settings.h` .

   ```
   #include "stdio.h"
   #include "stdlib.h"
   #include "string.h"
   #include "MQTTClient.h"
   #include "settings.h"
   ```
5. Uruchom definicję funkcji `main()` .

   ```
   int main(int argc, char* argv[])
   {
   ```
6. Zdefiniuj zmienne lokalne używane w programie.

   ```
   MQTTClient client;
   MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
   MQTTClient_message pubmsg;
   MQTTClient_deliveryToken token;
   int rc;
   ```

   Uwaga: Opcje połączenia są wymagane przez funkcję `MQTTClient_connect` . `MQTTClient_connectOptions_initializer` zawiera opcje domyślne.
7. Utwórz klienta.

```
MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
```

- & `client` jest wskaźnikiem do uchwytu dla nowo utworzonego klienta. Gdy ta funkcja zwraca kod powrotu 0, zawiera uchwyt do nowego klienta. W przykładzie założono sukces. Przetestuj kod błędu, aby uzyskać poprawne zakończenie w kodzie produkcyjnym.
- ADDRESS to identyfikator URI portu MQTT, który jest monitorowany przez demona dla przychodzących żądań połączeń klientów.
- `CLIENTID` to nazwa używana do identyfikowania klienta dla demona. Każdy aktywny klient musi mieć unikalną nazwę. Jeśli identyfikator klienta zostanie zduplikowany w dwóch uruchomionych klientach, w obu klientach zostanie zgłoszony wyjątek, a jeden klient kończy działanie. Nazwa jest używana przez demon do rozpoznania, że klient tjhat ponownie łączy się po rozłączeniu, patrz Identyfikator klienta.
- `MQTTCLIENT_PERSISTENCE_NONE` określa, że stan klienta jest wstrzymany w pamięci i jest tracony, jeśli wystąpi awaria systemu. Parametr `MQTTCLIENT_PERSISTENCE_DEFAULT` określa trwałość opartą na systemie plików, która zapewnia pewną ochronę przed awariami. W przypadku bardziej wyspecjalizowanych aplikacji można użyć parametru `MQTTCLIENT_PERSISTENCE_USER`, który udostępnia interfejs przeznaczony do implementowania własnego mechanizmu trwałości. Więcej szczegółowych informacji na ten temat zawiera dokumentacja interfejsu API produktu `MQTTClientPersistence.h`. To, czy wymagana jest trwałość, to pytanie o projekt aplikacji. Więcej informacji na ten temat zawiera sekcja Trwałość komunikatów.
- Domyślnym portem TCP/IP demona dla MQTT jest 1883. W tym przykładzie adres domyślny jest ustawiony na wartość `tcp://localhost:1883`.
- Do momentu wywołania funkcji `MQTTClient_connect` przetwarzanie komunikatów nie jest wymagane.

8. Połącz klienta z demonem.

```
if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
     printf("Failed to connect, return code %d\n", rc);
     exit(-1);
}
```

- Wywoływana jest funkcja `MQTTClient_connect` , przekazując uchwyt klienta i wskaźnik do opcji połączenia jako argumenty.
- Kod powrotu z wywołania `MQTTClient_connect` jest testowany w celu upewnia się, że żądanie nawiązania połączenia powiodło się.
- Jeśli działanie produktu `MQTTClient_connect` nie powiedzie się, program kończy działanie z kodem błędu -1.
- Po nawiązaniu połączenia z aplikacją można rozpocząć publikowanie i subskrybowanie.
- Mały komunikat "keep-alive" jest wysyłany co 20 sekund, aby zapobiec zamknięciu połączenia TCP/IP. Ta opcja jest ustawiana przez produkt `conn_opts.keepAliveInterval`.
- Sesja jest uruchamiana bez sprawdzania, czy zakończenie komunikatów inflight pozostało z poprzedniego połączenia, ponieważ parametr `conn_opts.cleansession` ma wartość true. Więcej informacji na ten temat zawiera sekcja Czyszczenie sesji.
- Dla połączenia nie zostanie utworzony ostatni komunikat testamentu i testament. Więcej informacji na ten temat zawiera sekcja Ostatnia wola i testament.

9. Zapełnij strukturę *MQTTClient_message* danymi, aby zdefiniować ładunek komunikatu i jego atrybuty.

```
pubmsg.payload = PAYLOAD;
pubmsg.payloadlen = strlen(PAYLOAD);
pubmsg.qos = QOS;
pubmsg.retained = 0;
```

- PAYLOAD to nasza treść wiadomości.
- W przykładzie użyto ładunku łańcucha, ale ładunki MQTT są tablicami bajtowymi. Długość łańcucha jest wymagana do określenia wielkości ładunku.

- W przykładzie publikowany jest komunikat QoS=1 , tak więc należy ustawić odpowiednio wartość
- Zachowany atrybut jest ustawiony na false (0), ponieważ komunikat nie jest zachowywany przez demona. Więcej informacji na ten temat zawiera sekcja Zachowane publikacje.

10. Opublikuj komunikat.

```
MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
```

- Funkcja publikowania określa klienta, temat i ładunek, który ma zostać wysłany do demona.
- `TOPIC` (temat) jest zdefiniowany w produkcie `settings.h` jako `MQTT Example`.
- Funkcja jest również przekazywana wskaźnikiem do obiektu *MQTTClient_deliveryToken*. Ten wskaźnik jest zapełniany leksem reprezentującym komunikat po powrocie do funkcji.
- Komunikat jest teraz bezpiecznie przesyłany do klienta MQTT, ale nie został jeszcze przesłany do demona. Jeśli komunikat ma wartość QoS=1 lub 2, to komunikat jest przechowywany lokalnie, na wypadek, gdyby klient nie zakończył się przed zakończeniem dostarczania.
- Ta funkcja zwraca kod błędu, który można przetestować pod kątem poprawnego zakończenia w kodzie produkcyjnym.

11. Poczekaj na potwierdzenie z serwera.

```
rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
```

- Przykład `pubsync.c` czeka na potwierdzenie z serwera, który potwierdza, że wiadomość została dostarczona.
- Argumenty klienta i tokenu identyfikują konkretny komunikat, który program oczekuje na zakończenie.
- Parametr `TIMEOUT` określa, jak długo program oczekuje na zakończenie dostarczania komunikatu. Zadanie Tworzenie publikatora asynchronicznego dla produktu MQ Telemetry Transport przy użyciu języka C przedstawia sposób odbierania potwierdzeń bez oczekiwania przy użyciu funkcji zwrotnych.
- Ta funkcja zwraca kod błędu, który może zostać przetestowany pod kątem poprawnego zakończenia w kodzie produkcyjnym.

12. Odłącz klienta od demona.

```
MQTTClient_disconnect(client, 10000);
```

- Klient rozłącza się z serwerem i oczekuje na zakończenie wszystkich funkcji zwrotnych (nieużywanych w tym przykładzie) dla komunikatów inflight.
- Drugi argument określa limit czasu wyciszania w milisekundach. Ten przykład czeka nawet 10 sekund na zakończenie wszystkich innych zadań, które muszą wykonać przed rozłączeniu.
- Ta funkcja zwraca kod błędu, który musi zostać przetestowany pod kątem poprawnego zakończenia w kodzie produkcyjnym.

13. Zwolniaj pamięć używaną przez klienta i zakończyj działanie programu.

```
MQTTClient_destroy(&client);
}
```

Wyniki

Aby wyświetlić publikacje wysłane przez tego klienta, należy utworzyć subskrybent w temacie `MQTT Example` . Więcej szczegółowych informacji na ten temat zawiera sekcja Tworzenie subskrybenta dla produktu MQ Telemetry Transport przy użyciu języka C .

Przykład

Rysunek 1 jest pełną listą kodu opisanego w sekcji Procedura. Plik `settings.h` na Rysunku 2 pozwala na zmianę domyślnych parametrów używanych w programie `pubsync.c`.

```
#include "stdio.h"
#include "stdlib.h"
#include "MQTTClient.h"
#include "settings.h"

int pubsync_main(int argc, char* argv[]) {
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_message pubmsg;
    MQTTClient_deliveryToken token;
    int rc;

    MQTTClient_create(&client, ADDRESS, CLIENTID,  MQTTCLIENT_PERSISTENCE_NONE, NULL);
    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
          printf("Failed to connect, return code %d\n", rc);
         exit(-1);
    }
    pubmsg.payload = PAYLOAD;
    pubmsg.payloadlen = strlen(PAYLOAD);
    pubmsg.qos = QOS;
    pubmsg.retained = 0;
    MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
    printf("Waiting for up to %d seconds for publication of %s\n"
          "on topic %s for client with ClientID: %s\n",
          TIMEOUT/1000, PAYLOAD, TOPIC, CLIENTID);
    rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
    printf("Message with delivery token %d delivered\n", token);
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
}
```

Rysunek 111. pubsync.c

```
#define ADDRESS   "tcp://localhost:1883"
#define CLIENTID  "ExampleClientPub"
#define TOPIC     "MQTT Example"
#define PAYLOAD   "Hello World!"
#define QOS      1
#define TIMEOUT   10000L
```

Rysunek 112. settings.h

Tworzenie publikatora asynchronicznego dla produktu MQ Telemetry Transport przy użyciu języka C

Kroki tworzenia aplikacji asynchronicznej publikatora klienta MQTT są opisane w trybie kursu. Każdy wiersz kodu C jest wyjaśniony. Po zakończeniu zadania zostanie utworzony asynchroniczny publikator MQTT.

W ramach tego zadania użytkownik śledzi kurs, aby zmodyfikować pierwszą aplikację publikatora. Modyfikacje umożliwiają aplikacji wysyłanie publikacji bez oczekiwania na potwierdzenie dostarczenia. Potwierdzenie dostarczenia jest odbierane przez funkcję zwrotną, którą tworzysz.

Zanim rozpoczniesz

Aplikacja kliencka opracowała biblioteki klienta MQTT v3 C klienta MQTT. Aplikacja nawiązuje połączenie z demonem WebSphere MQ Telemetry dla urządzeń w celu publikowania komunikatów. W sekcji Tworzenie pierwszego publikatora znajduje się przykład klienta komuniującego się z produktem WebSphere MQ Telemetry.

O tym zadaniu

Przykładem jest aplikacja publikowania, `pubasync.c`. Program `pubasync.c` publikuje komunikat z ładunkiem `Hello World!` do tematu `MQTT Example`, nie czekając na potwierdzenie, że publikacja została dostarczona do demona. Potwierdzenie dostarczenia jest odbierane w funkcji zwrotnej, `MQTTClient_deliveryComplete`.

W przypadku uproszczenia kody powrotu z niektórych używanych funkcji nie są testowane pod kątem poprawnego zakończenia. Kody powrotu można sprawdzić w kodzie produkcyjnym, aby upewnić się, że program zachowuje się zgodnie z oczekiwaniami. Jeśli wystąpi nieoczekiwany błąd, należy podjąć odpowiednie działania.

Po ustawieniu subskrybenta na `MQTT Example` można sprawdzić, czy aplikacja działa.

Aby utworzyć, zbudować i uruchomić klienta, należy użyć wybranego środowiska programistycznego C.

Kroki opisane w sekcji Procedura modyfikują aplikację `pubsync.c` z programu "Tworzenie pierwszej aplikacji publikatora produktu MQ Telemetry Transport za pomocą języka C" na stronie 525. Jeśli wolisz, możesz skopiować kod bezpośrednio z przykładów.

Procedura

1. Utwórz nowy, pusty plik źródłowy `callback.h`.
2. Skopiuj kod na Rysunku 2 do pliku.
   - Program `callback.h` deklaruje trzy metody wywołania zwrotnego wymagane dla operacji klienta asynchronicznego.
   - Zadeklarowana jest również zmienna *deliveredtoken*. Dostęp do tego programu jest uzyskiwany przez program główny i wywołanie zwrotne na różnych wątkach wykonania. W związku z tym uznaje się ją za ulotną. W przypadku korzystania z wywołań zwrotnych należy upewnić się, że dostęp do odpowiednich zmiennych jest możliwy w sposób bezpieczny dla wątku.
3. Utwórz nowy, pusty plik źródłowy `callback.c`.
4. Skopiuj kod z Rysunku 3 do pliku.
   - Produkt `callback.c` implementuje trzy metody wywołania zwrotnego używane przez klienta dla operacji asynchronicznej, `delivered`, `msgarrvd`i `connlost`.
5. Dodaj instrukcję include dla `callback.h` po innych dołączonych do produktu `pubasync.c`.

   ```
   #include "callback.h"
   ```

6. Skopiuj zawartość pliku `pubsync.c` do nowego pliku, `pubasync.c`.
7. Tuż przed wywołaniem funkcji `MQTTClient_connect` w produkcie `pubasync.c`należy ustawić metody wywołania zwrotnego dla klienta.

   ```
   MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);
   ```

   - Należy określić trzy funkcje wywołania zwrotnego. Te funkcje są implementowane w produkcie `callback.c`.
   - `MQTTClient_messageArrived` jest wywoływana po wysłaniu komunikatu do klienta ze względu na zgodną subskrypcję. Musi ona zwrócić wartość true, gdy odebrany komunikat został pomyślnie odebrany przez aplikację kliencką. Zwracanie wartości false wskazuje klientowi, że w aplikacji wystąpił problem z odebraniem komunikatu.
   - Program `MQTTClient_connectionLost` jest wywoływany, gdy klient utraci połączenie z serwerem.
   - Serwer `MQTTClient_deliveryComplete` jest wywoływany po dotarciu komunikatu QoS1 lub QoS2 przez serwer i potwierdzony przez niego. Nie jest wywoływana dla komunikatów QoS0 . W tym przykładzie funkcja zapisuje znacznik z dostarczonej wiadomości w elemencie *deliveredtoken* w celu wskazania, że komunikat został dostarczony.
   - Program `MQTTClient_setCallbacks` musi być wywoływany, gdy klient jest odłączony od serwera.
   - Drugi argument umożliwia przekazywanie informacji kontekstowych do funkcji zwrotnych. Ta opcja nie jest używana w przykładzie, więc jest ustawiona na NULL.

8. Bezpośrednio przed wywołaniem funkcji `MQTTClient_publishMessage`, usuń zaznaczenie pola *deliveredtoken*. `MQTTClient_deliveryComplete` do ustawiania elementu *deliveredtoken* po odebraniu znacznika.

   ```
   deliveredtoken = 0;
   ```

9. Usuń wywołanie `MQTTClient_waitForCompletion` i `printf` po nim, a następnie zastąp pętlą oczekując na zgodność oryginalnego znacznika i znacznika odebranego w wywołaniu zwrotnym.

   ```
   while(deliveredtoken != token);
   ```

   Jest to przykład i nie radzi sobie z wieloma sytuacjami, które muszą być zakwaterowane w projektowaniu kodu produkcyjnego. Do takich sytuacji należą:

   - Jeśli dostarczenie sprawy nie zostanie zakończone, można zaimplementować limit czasu
   - Wiele komunikatów może być rozświetlaniem. Przykładowy program umożliwia sprawdzanie tylko jednego znacznika dostarczania w danym momencie.

10. Odłącz klienta od demona.

    ```
    MQTTClient_disconnect(client, 10000);
    ```

    - Klient rozłącza się z serwerem i czeka na zakończenie wszystkich funkcji wywołania zwrotnego dla komunikatów inflight.
    - Drugi argument określa limit czasu wyciszania w milisekundach. Ten przykład czeka nawet 10 sekund na zakończenie wszystkich innych zadań, które muszą wykonać przed rozłączeniu.
    - Ta funkcja zwraca kod błędu, który musi zostać przetestowany pod kątem poprawnego zakończenia w kodzie produkcyjnym.

11. Zwolniaj pamięć używaną przez klienta i zakończyj działanie programu.

    ```
    MQTTClient_destroy(&client);
    }
    ```

Wyniki

Aby wyświetlić publikację wysłanej przez tego klienta, należy utworzyć subskrybent w temacie `MQTT Example` . Więcej szczegółowych informacji na ten temat zawiera sekcja Tworzenie subskrybenta dla produktu MQ Telemetry Transport .

Przykład

`pubasync.c`, `callbacks.c` i `callbacks.h` są pełnymi listingami kodu opisanymi w procedurze.

```
#include "stdio.h"
#include "stdlib.h"
#include "MQTTClient.h"
#include "settings.h"
#include "callback.h"

int main(int argc, char* argv[]) {
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_message pubmsg;
    MQTTClient_deliveryToken token;
    int rc;

    MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);
    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
        printf("Failed to connect, return code %d\n", rc);
        exit(-1);
    }
    pubmsg.payload = PAYLOAD;
    pubmsg.payloadlen = strlen(PAYLOAD);
    pubmsg.qos = QOS;
    pubmsg.retained = 0;
    deliveredtoken = 0;
    MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
    printf("Waiting for publication of %s\n"
           "on topic %s for client with ClientID: %s\n", PAYLOAD, TOPIC, CLIENTID);
    while(deliveredtoken != token);
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
}
```

Rysunek 113. pubasync.c

```
MQTTClient_deliveryComplete delivered;
MQTTClient_messageArrived msgarrvd;
MQTTClient_connectionLost connlost;

extern volatile MQTTClient_deliveryToken deliveredtoken;
```

Rysunek 114. callback.h

```
#include "MQTTClient.h"

volatile MQTTClient_deliveryToken deliveredtoken;

void delivered(void *context, MQTTClient_deliveryToken dt)
{
    printf("Message with token value %d delivery confirmed\n", dt);
    deliveredtoken = dt;
}

int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message)
{
    int i;
    char* payloadptr;

    printf("Message arrived\n");
    printf("     topic: %s\n", topicName);
    printf("   message: ");

    payloadptr = message->payload;
    for(i=0; i<message->payloadlen; i++) {
        putchar(*payloadptr++);
    }
    putchar('\n');
    MQTTClient_freeMessage(&message);
    free(topicName);
    return 1;
}

void connlost(void *context, char *cause)
{
    printf("\nConnection lost\n");
    printf("     cause: %s\n", cause);
}
```

Rysunek 115. `callback.c`

```
#define ADDRESS    "tcp://localhost:1883"
#define CLIENTID   "ExampleClientPub"
#define TOPIC      "MQTT Example"
#define PAYLOAD    "Hello World!"
#define QOS        1
#define TIMEOUT    10000L
```

Rysunek 116. `settings.h`

Tworzenie subskrybenta dla produktu MQ Telemetry Transport przy użyciu języka C

Kroki tworzenia aplikacji subskrybenta klienta MQTT są opisane w kursie mody. Każdy wiersz kodu C jest wyjaśniony. Po zakończeniu zadania zostanie utworzony subskrybent MQTT.

Zanim rozpoczniesz

Aplikacja kliencka opracowała biblioteki klienta MQTT v3 C klienta MQTT. Aplikacja nawiązuje połączenie z demonem WebSphere MQ Telemetry dla urządzeń w celu publikowania komunikatów. W sekcji Tworzenie pierwszego publikatora znajduje się przykład klienta komuniującego się z produktem WebSphere MQ Telemetry.

O tym zadaniu

Przykładem jest aplikacja subskrybenta, `subscribe.c`. Program `subscribe.c` subskrybuje temat `MQTT Example` i czeka na publikacje zgodne z subskrypcją do momentu zakończenia programu przez użytkownika.

Subskrybent tworzy subskrypcję tematu i oczekuje na komunikaty, które są zgodne z tematem subskrypcji. Komunikaty publikowane w czasie, gdy klient jest odłączony i które są zgodne z subskrypcją utworzoną wcześniej przez klienta, można odebrać po ponownym nawiązaniu połączenia przez klienta. Usługa telemetryczna WebSphere MQ (MQXR) lub demon dla urządzeń rozpoznaje klienta, który

wcześniej był połączony za pomocą identyfikatora klienta. Więcej informacji na ten temat zawiera sekcja Identyfikator klienta. Atrybut boolowski `MQTTClient_connectOptions.cleansession` określa, czy publikacje wysłane wcześniej są odbierane, czy nie. Szczegółowe informacje na ten temat zawiera sekcja “Czyste sesje” na stronie 541.

W przypadku uproszczenia kody powrotu z niektórych używanych funkcji nie są testowane pod kątem poprawnego zakończenia. Kody powrotu mogą być sprawdzane w kodzie produkcyjnym w celu zapewnienia, że program zachowuje się zgodnie z oczekiwaniami. Jeśli wystąpi nieoczekiwany błąd, można podjąć odpowiednie działania.

Wcześniej opisanych programów przykładowych publikowania można użyć do wysyłania zgodnych publikacji do demona WebSphere MQ Telemetry dla urządzeń. Alternatywnie można użyć programu WebSphere MQ explorer w celu utworzenia publikacji testowych w temacie `MQTT Example` , jeśli klient ma zostać podłączony do kanału WebSphere MQ Telemetry .

W instrukcjach w sekcji Procedura założono, że w jednym z wcześniejszych zadań zostały już utworzone pliki `callback.c, callback.h`i `settings.h` .

Aby utworzyć, zbudować i uruchomić klienta, należy użyć wybranego środowiska programistycznego C. Jeśli wolisz, możesz skopiować kod bezpośrednio z przykładów.

Procedura

1. Utwórz kopię produktu `settings.h` dla tego przykładu, a następnie zmień instrukcję definiowania *CLIENTID* na następujące:

   ```
   #define CLIENTID "ExampleClientSub"
   ```

   - Jeśli dwa klienty o tym samym identyfikatorze próbują połączyć się z jednym serwerem, jeden z nich zostanie wymuszony rozłączony. Zwykle nowa próba nawiązania połączenia powiedzie się, a starsze połączenie zostanie odłączone.
   - Zmiana parametru `ClientID` umożliwia korzystanie z wcześniej opracowanych przykładów publikowania w celu wysyłania komunikatów do tego subskrybenta.
2. Utwórz nowy, pusty plik źródłowy `subscribe.c`.
3. Kroki, które należy wykonać, wyjaśniają kod. Postępuj zgodnie z instrukcjami lub skopiuj kod z Rysunek 117 na stronie 536 do pliku `subscribe.c`.
4. Dodaj do pliku nagłówkowego instrukcje dotyczące wymaganych bibliotek standardowych oraz plików `MQTTClient.h` i `settings.h` .

   ```
   #include "stdio.h"
   #include "stdlib.h"
   #include "MQTTClient.h"
   #include "settings.h"
   ```
5. Uruchom definicję funkcji `main()` .

   ```
   int main(int argc, char* argv[]) {
   ```
6. Zdefiniuj zmienne lokalne używane w programie.

   ```
   MQTTClient client;
   MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
   MQTTClient_deliveryToken token;
   int rc;
   ```

   Opcje połączenia są wymagane przez funkcję `MQTTClient_connect` . `MQTTClient_connectOptions_initializer` zawiera opcje domyślne.
7. Utwórz klienta.

   ```
   MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
   ```

- `& client` jest wskaźnikiem do uchwytu dla nowo utworzonego klienta. Gdy ta funkcja zwraca kod powrotu 0, wskaźnik zawiera uchwyt do nowego klienta. W przykładzie założono sukces. Kod błędu może zostać przetestowany pod kątem poprawnego zakończenia w kodzie produkcyjnym.
- ADDRESS to identyfikator URI portu MQTT, który jest monitorowany przez demona dla przychodzących żądań połączeń klientów.
- `CLIENTID` to nazwa używana do identyfikowania klienta dla demona. Każdy aktywny klient musi mieć unikalną nazwę. Jeśli identyfikator klienta zostanie zduplikowany w dwóch uruchomionych klientach, w obu klientach zostanie zgłoszony wyjątek, a jeden klient kończy działanie. Nazwa jest używana przez demon do rozpoznania, że klient ponownie łączy się po rozłączeniu, patrz Identyfikator klienta.
- `MQTTCLIENT_PERSISTENCE_NONE` określa, że stan klienta jest wstrzymany w pamięci i jest tracony, jeśli wystąpi awaria systemu. Parametr `MQTTCLIENT_PERSISTENCE#_DEFAULT` określa trwałość opartą na systemie plików, która zapewnia pewną ochronę przed awariami. W przypadku bardziej wyspecjalizowanych aplikacji można użyć parametru `MQTTCLIENT_PERSISTENCE_USER`, który udostępnia interfejs przeznaczony do implementowania własnego mechanizmu trwałości. To, czy wymagana jest trwałość, to pytanie o projekt aplikacji. Więcej informacji na ten temat zawiera sekcja Trwałość komunikatów.
- Domyślnym portem TCP/IP demona dla MQTT jest 1883. W tym przykładzie adres domyślny jest ustawiony na wartość `tcp://localhost:1883`.
- Do momentu wywołania funkcji `MQTTClient_connect` przetwarzanie komunikatów nie jest wymagane.

8. Połącz klienta z demonem

```
if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
    printf("Failed to connect, return code %d\n", rc);
    exit(-1);
}
```

   - Wywoływana jest funkcja `MQTTClient_connect` , przekazując uchwyt klienta i wskaźnik do opcji połączenia jako argumenty.
   - Kod powrotu z wywołania `MQTTClient_connect` jest testowany w celu upewnia się, że żądanie nawiązania połączenia powiodło się.
   - Jeśli wywołanie połączenia nie powiedzie się, program kończy działanie z kodem błędu -1.
   - Po nawiązaniu połączenia aplikacja może rozpocząć publikowanie i subskrybowanie.
   - Mały komunikat "keep-alive" jest wysyłany co 20 sekund, aby zapobiec zamknięciu połączenia TCP/IP. Ta opcja jest ustawiana przez produkt `conn_opts.keepAliveInterval`.
   - Sesja jest uruchamiana bez sprawdzania, czy zakończenie komunikatów inflight pozostało z poprzedniego połączenia, ponieważ parametr *conn_opts.cleansession* ma wartość true. Więcej informacji na ten temat zawiera sekcja Czyszczenie sesji.
   - Dla połączenia nie zostanie utworzony ostatni komunikat testamentu i testament. Więcej informacji na ten temat zawiera sekcja Ostatnia wola i testament .

9. Zasubskrybuj temat.

```
MQTTClient_subscribe(client, TOPIC, QOS);
```

   - Użyj funkcji `MQTTClient_subscribe` , aby zasubskrybować aplikację kliencką do wybranego tematu. Nazwa tematu może zawierać znaki wieloznaczne. Szczegółowe informacje na ten temat zawiera sekcja “Łańcuchy tematów i filtry tematów w klientach MQTT” na stronie 556.
   - Ustawienie QoS określa maksymalną jakość usługi, która jest stosowana do komunikatów wysyłanych do tego subskrybenta. Serwer wysyła komunikaty przy niższej wartości tego ustawienia i ustawieniu QoS dla oryginalnego komunikatu.
   - Ta funkcja zwraca kod błędu, który może zostać przetestowany pod kątem poprawnego zakończenia w kodzie produkcyjnym.

10. Zaczekaj w pętli do momentu, gdy użytkownik wprowadzi znak 'Q' z klawiatury.

```
do {
    ch = getchar();
    } while(ch!='Q' && ch != 'q');
```

Program czeka na nadejście komunikatów. W tym przykładzie cała obsługa komunikatów ma miejsce w funkcji zwrotnej `MQTTClient_messageArrived`. Szczegółowe informacje na ten temat zawiera sekcja “Odbieranie komunikatów” na stronie 535.

11. Odłącz klienta od demona.

```
MQTTClient_disconnect(client, 10000);
```

- Klient rozłącza się z serwerem i oczekuje na zakończenie wszystkich funkcji zwrotnych (nieużywanych w tym przykładzie) dla komunikatów inflight.
- Drugi argument określa limit czasu wyciszania w milisekundach. Ten przykład oczekuje nawet 10 sekund na zakończenie wszystkich innych prac, które musi wykonać przed rozłączeniu.
- Ta funkcja zwraca kod błędu, który może zostać przetestowany pod kątem poprawnego zakończenia w kodzie produkcyjnym.

12. Zwolniaj pamięć używaną przez klienta i zakończyj działanie programu.

```
MQTTClient_destroy(&client);
}
```

Odbieranie komunikatów

O tym zadaniu

Gdy komunikaty są przesyłane z serwera, uruchamiana jest funkcja `MQTTClient_messageArrived` . Kroki, które należy wykonać, wyjaśniają kod.

Procedura

1. Uruchom definicję funkcji zwrotnej. Ta definicja musi być zgodna z szablonem funkcji `MQTTClient_messageArrived` .

```
int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message) {
```

- `context` zapewnia dostęp do kontekstu przekazanego do biblioteki klienta po wywołaniu funkcji `MQTTClient_setCallbacks` . Ta funkcja nie jest używana w przykładzie.
- `topicName` jest wskaźnikiem do tematu, do którego został opublikowany odebrany komunikat. Jeśli zasubskrybowano przy użyciu znaków wieloznacznych, ten parametr identyfikuje konkretny temat używany dla komunikatu.
- `topicLen` to długość łańcucha tematu. Ta opcja jest dostępna dla użytkowników, którzy muszą osadzać znaki NULL w łańcuchach tematów.
- `komunikat` jest wskaźnikiem do struktury `MQTTClient_message` zawierającej ładunek komunikatu i atrybuty.

2. Zdefiniuj używane zmienne lokalne.

```
int i;
char* payloadptr;
```

Te zmienne są używane w przykładzie do wypisania ładunku przez iterowanie przez iterację.

3. Wydrukuj wiadomość, wyświetlając temat i ładunek komunikatu

```
printf("Message arrived\n");
printf("     topic: %s\n",topicName);
printf("   message: ");
payloadptr = message->payload;
for(i=0; i<message->payloadlen; i++){
    putchar(*payloadptr++);
```

```
    }
putchar('\n');
```

- W przykładzie założono, że odebrany ładunek jest sekwencją znaków drukowalnych.
- Ładunek MQTT jest tablicą bajtów. Aplikacja jest odpowiedzialna za interpretowanie ich znaczenia.

4. Zwolń pamięć używaną do przechowywania komunikatu.

```
MQTTClient_freeMessage(&message);
MQTTClient_free(topicName);
```

- W tym przykładzie cała obsługa komunikatów ma miejsce w funkcji zwrotnej.
- Upewnij się, że funkcje wywołania zwrotnego są krótkie i jak najszybciej, zwracaj sterowanie do wywołującego wątku.
- Wskaźnik komunikatu jest przekazywany do obsługi w głównej części programu.
- Program główny musi zwolnić pamięć używaną przez komunikat po zakończeniu przetwarzania. `MQTTClient_freeMessage()` to wygodna funkcja, która zwraca dwa bloki pamięci używane do przechowywania struktury *MQTTClient_message* i ładunku komunikatu z powrotem do systemu. Pamięć przydzielona do partycji `topicName` musi być zwolniona oddzielnie, jak to pokazano na rysunku.

5. Zwraca wartość true, gdy wywołanie zwrotne zakończyło się pomyślnie.

```
    return 1;
}
```

- Zwrócenie wartości true wskazuje, że biblioteka klienta może traktować komunikat jako pomyślnie dostarczony.
- Jeśli funkcja zwrotna nie może poprawnie przetworzyć komunikatu, zwracana jest wartość false. Na przykład, jeśli wywołanie zwrotne powoduje umieszczanie komunikatów w kolejce dla głównego programu do przetworzenia, a kolejka jest pełna, to zwrócenie wartości false byłoby właściwe.
- W przypadku komunikatów QoS1 i QoS2 zwracanie wartości false wskazuje, że komunikat nie został dostarczony, a dalsze próby dostarczenia go są wykonywane.

Przykładowy kod

```
#include "stdio.h"
#include "stdlib.h"
#include "MQTTClient.h"
#include "settings.h"
#include "callback.h"

int main(int argc, char* argv[]) {
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    int rc;
    int ch;

    MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);

    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);

    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
        printf("Failed to connect, return code %d\n", rc);
        exit(-1);
        }
    printf("Subscribing to topic %s\nfor client %s using QoS%d\n\n"
           "Press Q<Enter> to quit\n\n", TOPIC, CLIENTID, QOS);

    MQTTClient_subscribe(client, TOPIC, QOS);
    do {
        ch = getchar();
        } while(ch!='Q' && ch != 'q');
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
}
```

Rysunek 117. `subscriber.c`

```
#include "MQTTClient.h"

volatile MQTTClient_deliveryToken deliveredtoken;

void delivered(void *context, MQTTClient_deliveryToken dt) {
    printf("Message with token value %d delivery confirmed\n", dt);
    deliveredtoken = dt;
}

int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message) {
    int i;
    char* payloadptr;

    printf("Message arrived\n");
    printf("     topic: %s\n", topicName);
    printf("   message: ");

    payloadptr = message->payload;
    for(i=0; i<message->payloadlen; i++) {
        putchar(*payloadptr++);
        }
    putchar('\n');
    MQTTClient_freeMessage(&message);
    MQTTClient_free(topicName);
    return 1;
}
void connlost(void *context, char *cause) {
    printf("\nConnection lost\n");
    printf("     cause: %s\n", cause);
}
```

Rysunek 118. `callback.h`

```
#define ADDRESS    "tcp://localhost:1883"
#define CLIENTID   "ExampleClientSub"
#define TOPIC      "MQTT Example"
#define PAYLOAD    "Hello World!"
#define QOS        1
#define TIMEOUT    10000L
```

Rysunek 119. `settings.h`

Pojęcia związane z programowaniem klienta MQTT

Pojęcia opisane w tej sekcji ułatwiają zrozumienie bibliotek klienta Java, JavaScript i C dla wersji 3.1 produktu MQTT protocol. Pojęcia uzupełniają dokumentację API dołączoną do bibliotek klienta.

Produkt `com.ibm.micro.client.mqttv3` zawiera klasy udostępniające metody publiczne dla bibliotek klienta dla protokołu MQTT w wersji 3.1 . Wersja pakietu `com.ibm.micro.client.mqttv3` oraz towarzyszące pakiety, które implementują protokół dla produktu Java SE i ME, są dostarczane wraz z instalacją produktu IBM WebSphere MQ Telemetry. Aby uzyskać najnowszą wersję bibliotek klienta MQTT (Java, JavaScript oraz aby wyświetlić lub pobrać dokumentację interfejsu API, należy zapoznać się z informacjami w sekcji "Skorowidz programistyczny klienta MQTT".

Aby utworzyć i uruchomić klienta MQTT , należy skopiować lub zainstalować te pakiety na urządzeniu klienckim. Nie ma potrzeby instalowania odrębnego środowiska wykonawczego klienta.

Warunki licencjonowania dla klientów są powiązane z serwerem, z którym łączą się klienty.

Biblioteki klienta produktu MQTT są implementacjami referencyjnymi w wersji 3.1 produktu MQTT protocol. Istnieje możliwość zaimplementowania własnych klientów w różnych językach odpowiednich dla różnych platform urządzeń. Więcej informacji na ten temat zawiera sekcja MQ Telemetry Transport format and protocol.

Dokumentacja interfejsu API nie zawiera żadnych założeń dotyczących serwera MQTT , z którym klient jest połączony. Zachowanie klienta może się nieznacznie różnić w przypadku połączenia z różnymi serwerami. W poniższych opisach opisano zachowanie klienta podczas połączenia z usługą telemetryczną IBM WebSphere MQ .

Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT

W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Wywołania zwrotne

Interfejs `MqttCallback` ma trzy metody wywołania zwrotnego. Patrz przykładowa implementacja w pliku Callback.java.

`connectionLost(java.lang.Throwable cause)`

Program `connectionLost` jest wywoływany po wystąpieniu błędu komunikacyjnego w celu usunięcia połączenia. Jest ona również wywoływana, jeśli serwer usunie połączenie w wyniku błędu na serwerze po nawiązaniu połączenia. Błędy serwera są rejestrowane w dzienniku błędów menedżera kolejek. Serwer usuwa połączenie z klientem, a klient wywołuje `MqttCallback.connectionLost`.

Jedynymi błędami zdalnymi zgłaszanych jako wyjątki w tym samym wątku, co aplikacja kliencka, są wyjątki od `MqttClient.connect`. Błędy wykryte przez serwer po nawiązaniu połączenia są zgłaszane z powrotem do metody wywołania zwrotnego `MqttCallback.connectionLost` jako `throwables`.

Typowe błędy serwera, których wynikiem jest `connectionLost` , to błędy autoryzacji. Na przykład serwer telemetryczny podejmuje próbę opublikowania w temacie w imieniu klienta, który nie ma uprawnień do publikowania w tym temacie. Wszystkie wyniki w kodzie warunku `MQCC_FAIL` zwracane do serwera telemetrycznego mogą spowodować usunięcie połączenia.

`deliveryComplete(MqttDeliveryToken token)`

Program `deliveryComplete` jest wywoływany przez klienta MQTT w celu przekazania tokenu dostarczania z powrotem do aplikacji klienckiej. Patrz “Znaczniki dostawy” na stronie 544. Za pomocą znacznika dostarczania wywołanie zwrotne może uzyskać dostęp do opublikowanego komunikatu przy użyciu metody `token.getMessage`.

Gdy wywołanie zwrotne aplikacji zwraca sterowanie do klienta MQTT po wywołaniu przez metodę `deliveryComplete` , dostarczenie jest zakończone. Do czasu zakończenia dostarczania komunikaty z funkcją QoS 1 lub 2 są zachowywane przez klasę trwałości.

Wywołanie funkcji `deliveryComplete` jest punktem synchronizacji między aplikacją a klasą trwałości. Metoda `deliveryComplete` nigdy nie jest wywoływana dwukrotnie dla tego samego komunikatu.

Gdy wywołanie zwrotne aplikacji zwraca wartość z `deliveryComplete` do klienta MQTT , klient wywołuje `MqttClientPersistence.remove` dla komunikatów z funkcją QoS 1 lub 2. Program `MqttClientPersistence.remove` usuwa lokalnie zapisaną kopię opublikowanego komunikatu.

Z perspektywy przetwarzania transakcji wywołanie funkcji `deliveryComplete` jest transakcją jednofazową, która zatwierdza dostawę. Jeśli przetwarzanie nie powiedzie się podczas wywołania zwrotnego, po ponownym uruchomieniu klienta `MqttClientPersistence.remove` zostanie ponownie wywołana w celu usunięcia lokalnej kopii opublikowanego komunikatu. Wywołanie zwrotne nie jest ponownie wywoływane. Jeśli do przechowywania dziennika dostarczonych komunikatów używana jest procedura zwrotna, nie można zsynchronizować dziennika z klientem MQTT . Jeśli dziennik ma być niezawodnie przechowywany, należy zaktualizować dziennik w klasie `MqttClientPersistence` .

Znacznik dostawy i komunikat są przywoływane przez główny wątek aplikacji i klienta MQTT . Klient MQTT wyrejestrowuje obiekt `MqttMessage` po zakończeniu dostarczania, a także obiekt znacznika dostarczania, gdy klient rozłącza się. Obiekt `MqttMessage` może być czyszczony po zakończeniu dostarczania, jeśli aplikacja kliencka wyłuskuje go. Znacznik dostarczania może być czyszczony po rozłączeniu sesji.

Atrybuty `MqttDeliveryToken` i `MqttMessage` można pobrać po opublikowaniu komunikatu. Jeśli po opublikowaniu komunikatu zostanie podjęta próba ustawienia atrybutów `MqttMessage` , wynik nie zostanie zdefiniowany.

Klient MQTT kontynuuje przetwarzanie potwierdzeń dostarczenia, jeśli klient ponownie nawiąże połączenie z poprzednią sesją przy użyciu tego samego `ClientIdentifier`. Patrz "Czyste sesje" na stronie 541. Aplikacja kliencka MQTT musi ustawić wartość `MqttClient.CleanSession` na `false` dla poprzedniej sesji, a następnie ustawić ją na wartość `false` w nowej sesji. Klient MQTT tworzy nowe znaczniki dostarczania i obiekty komunikatów w nowej sesji dla oczekujących dostaw. Odtwarza obiekty za pomocą klasy `MqttClientPersistence` . Jeśli klient aplikacji nadal ma odwołania do starych tokenów dostawy i komunikatów, wyłuskuj je. Wywołanie zwrotne aplikacji jest wywoływane w nowej sesji dla wszystkich dostaw zainicjowanych w poprzedniej sesji i zakończonych w tej sesji.

Wywołanie zwrotne aplikacji jest wywoływane po nawiązaniu połączenia przez klient aplikacji po zakończeniu dostarczania oczekującego. Zanim klient aplikacji połączy się, będzie mógł pobrać oczekujące dostawy za pomocą metody `MqttClient.getPendingDeliveryTokens` .

Zauważ, że aplikacja kliencka pierwotnie utworzyła opublikowany obiekt komunikatu i jego tablicę bajtów ładunku. Klient MQTT odwołuje się do tych obiektów. Obiekt komunikatu zwrócony przez znacznik dostarczania w metodzie `token.getMessage` nie musi być tym samym obiektem komunikatu tworzonym przez klienta. Jeśli nowa instancja klienta MQTT ponownie odtwarza znacznik dostarczania, klasa `MqttClientPersistence` odtwarza obiekt `MqttMessage` . For consistency `token.getMessage` returns `null` if `token.isCompleted` is `true`, regardless of whether the message object was created by the application client or the `MqttClientPersistence` class.

`messageArrived(MqttTopic topic, MqttMessage message)`

Program `messageArrived` jest wywoływany po nadejściu publikacji dla klienta, który jest zgodny z tematem subskrypcji. `temat` to temat publikowania, a nie filtr subskrypcji. Dwa mogą być różne, jeśli filtr zawiera znaki wieloznaczne.

Jeśli temat pasuje do wielu subskrypcji utworzonych przez klienta, klient otrzymuje wiele kopii tej publikacji. Jeśli klient publikuje w temacie, do którego również subskrybuje, otrzymuje kopię własnej publikacji.

Jeśli komunikat jest wysyłany z usługą QoS 1 lub 2, komunikat jest zapisywany przez klasę `MqttClientPersistence` przed wywołaniami `messageArrived`klienta MQTT . `messageArrived` zachowuje się jak `deliveryComplete`: jest wywoływana tylko raz dla publikacji, a lokalna kopia publikacji jest usuwana przez program `MqttClientPersistence.remove` , gdy program `messageArrived` powróci do klienta MQTT . Klient MQTT usuwa odwołania do tematu i komunikatu, gdy program `messageArrived` powróci do klienta MQTT . Obiekty tematów i komunikatów są czyszczeniem pamięci, jeśli klient aplikacji nie ma odwołania do odwołania do obiektów.

Synchronizacja aplikacji zwrotnych, wielowątkowych i aplikacji klienckich

Klient MQTT wywołuje metodę wywołania zwrotnego w osobnym wątku do głównego wątku aplikacji. Aplikacja kliencka nie tworzy wątku dla wywołania zwrotnego, jest on tworzony przez klienta MQTT .

Klient MQTT synchronizuje metody wywołania zwrotnego. W danej chwili działa tylko jedna instancja metody wywołania zwrotnego. Synchronizacja ułatwia aktualizację obiektu, który utalentowany jest, które publikacje zostały dostarczone. Jedna instancja serwera `MqttCallback.deliveryComplete` jest uruchamiana jednocześnie, a więc można bezpiecznie aktualizować ją bez dalszej synchronizacji. Jest to również przypadek, że tylko jedna publikacja dociera w danym momencie. Kod w metodzie `messageArrived` może aktualizować obiekt bez synchronizacji. Jeśli odwołujesz się do "tally" lub obiektu, który jest aktualizowany, w innym wątku, zsynchronizuj ten sam obiekt lub obiekt.

Znacznik dostarczania udostępnia mechanizm synchronizacji między głównym wątkiem aplikacji a publikacją. Metoda `token.waitForCompletion` oczekuje na zakończenie dostarczania konkretnej publikacji lub do momentu utraty ważności opcjonalnego limitu czasu. Produkt `token.waitForCompletion` może być używany w kilku prostych sposobach przetwarzania jednej publikacji w danym momencie:

1. Wstrzymanie klienta aplikacji do czasu dostarczenia publikacji jest zakończone; patrz Rysunek 88 na stronie 492.

2. Aby przeprowadzić synchronizację z metodą `MqttCallback.deliveryComplete` . Tylko w przypadku, gdy program `MqttCallback.deliveryComplete` powróci do programu MQTT , zostanie wznowiony `token.waitForCompletion` . Za pomocą tego mechanizmu można synchronizować działający kod w produkcie `MqttCallback.deliveryComplete` , zanim kod zostanie uruchomiony w głównym wątku aplikacji.

Co, jeśli chcesz opublikować bez czekania na każdą publikację, która ma być dostarczona, ale chcesz potwierdzić, kiedy wszystkie publikacje zostały dostarczone? W przypadku publikowania w pojedynczym wątku ostatnia publikacja, która ma zostać wysłana, jest również ostatnią dostarczoną.

Synchronizacja żądań wysłanych do serwera

W sekcji Tabela 70 na stronie 540 opisano metody w kliencie Java produktu MQTT , które wysyłają żądanie do serwera. Jeśli klient aplikacji nie ustawi nieokreślonego limitu czasu, klient nigdy nie będzie czekał bezterminowo na serwer. Jeśli klient zawiesi się, jest to problem z programowaniem aplikacji lub defekt w kliencie MQTT .

Tabela 70. Zachowanie synchronizacji metod, których wynikiem są żądania do serwera

Metoda	Synchronizacja	Limit czasu
`MqttClient.Connect`	Oczekuje na nawiązanie połączenia z serwerem.	Wartość domyślna to 30 sekund lub wartość ustawiona przez parametr, a następnie zgłasza wyjątek.
`MqttClient.Disconnect`	Oczekuje, że klient MQTT zakończy pracę, którą musi wykonać, a sesja TCP/IP zostanie rozłączona.	
`MqttClient.Subscribe`	Oczekuje na zakończenie działania metody Subskrybuj lub UnSubscribe .	
`MqttClient.UnSubscribe`		
`MqttClient.Publish`	Powraca natychmiast do wątku aplikacji po przekazaniu żądania do klienta MQTT .	Brak.
`MqttDeliveryToken.waitForCompletion`	Oczekuje na zwrócenie znacznika dostarczania.	Nieokreślony lub jako parametr ustawiony jako parametr.

Pojęcia pokrewne

Czyste sesje
Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Identyfikator klienta

Znaczniki dostawy

Publikacja ostatniego testamentu i testamentu
Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Trwałość komunikatu w klientach MQTT

Publikacje

Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

Jakość usług świadczonych przez klienta MQTT
Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Zachowane publikacje i klienty MQTT
Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Subskrypcje
Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Łańcuchy tematów i filtry tematów w klientach MQTT
Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Czyste sesje

Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Po połączeniu aplikacji klienckiej MQTT za pomocą metody `MqttClient.connect` klient identyfikuje połączenie przy użyciu identyfikatora klienta i adresu serwera. Serwer sprawdza, czy informacje o sesji zostały zeskładowane z poprzedniego połączenia z serwerem. Jeśli poprzednia sesja nadal istnieje, a następnie `cleanSession=true`, to poprzednie informacje o sesji na kliencie i serwerze zostaną wyczyszczone. Jeśli `cleanSession=false` poprzednia sesja została wznowiona. Jeśli żadna wcześniejsza sesja nie istnieje, zostanie uruchomiona nowa sesja.

Uwaga: Administrator produktu WebSphere MQ może wymusić zamknięcie otwartej sesji i usunięcie wszystkich informacji o sesji. Jeśli klient ponownie otworzy sesję z programem `cleanSession=false`, uruchamiana jest nowa sesja.

Publikacje

Jeśli przed nawiązaniem połączenia z klientem zostanie użyta wartość domyślna `MqttConnectOptions`lub zostanie ustawiona wartość `MqttConnectOptions.cleanSession` na `true` , wszystkie oczekujące dostawy publikacji dla klienta zostaną usunięte po nawiązaniu połączenia przez klienta.

Ustawienie sesji czyszczenia nie ma wpływu na publikacje wysłane z produktem `QoS=0`. W przypadku produktów `QoS=1` i `QoS=2`użycie produktu `cleanSession=true` może spowodować utratę publikacji.

Subskrypcje

W przypadku korzystania z domyślnej klasy `MqttConnectOptions` lub w przypadku ustawienia wartości `true` na potrzeby atrybutu `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia z klientem wszystkie stare subskrypcje klienta zostaną usunięte po nawiązaniu połączenia z klientem. Wszystkie nowe subskrypcje dokonane przez klienta podczas sesji zostaną usunięte po rozłączeniu.

W przypadku ustawienia wartości `false` na potrzeby atrybutu `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia wszystkie subskrypcje tworzone przez klienta będą dodawane do wszystkich subskrypcji klienta, które istniały przed nawiązaniem połączenia. Wszystkie subskrypcje pozostaną aktywne po rozłączeniu połączenia z klientem.

Innym podejściem umożliwiającym zrozumienie wpływu atrybutu `cleanSession` na subskrypcje jest uznanie go za atrybut modalny. W domyślnym trybie atrybutu `cleanSession=true` klient tworzy subskrypcje i odbiera publikacje tylko w zasięgu sesji. W alternatywnym trybie atrybutu `cleanSession=false` subskrypcje są trwałe. Można nawiązywać połączenia z klientem i je rozłączać, a jego subskrypcje pozostają aktywne. Po ponownym nawiązaniu połączenia z klientem odbiera on wszystkie niedostarczone publikacje. Po nawiązaniu połączenia klient może modyfikować zestaw subskrypcji aktywnych na jego potrzeby.

Tryb atrybutu `cleanSession` należy ustawić przed nawiązaniem połączenia. Ten tryb będzie obowiązywać przez cały czas trwania sesji. Aby zmienić ustawienie trybu, należy rozłączyć połączenie z klientem, a następnie ponownie je nawiązać. W przypadku zmiany używanego trybu atrybutu z `cleanSession=false` na `cleanSession=true` wszystkie wcześniejsze subskrypcje klienta oraz wszystkie publikacje, które nie zostały odebrane, zostaną usunięte.

Pojęcia pokrewne

Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT
W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Identyfikator klienta

Znaczniki dostawy

Publikacja ostatniego testamentu i testamentu
Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Trwałość komunikatu w klientach MQTT

Publikacje
Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

Jakość usług świadczonych przez klienta MQTT
Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Zachowane publikacje i klienty MQTT
Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Subskrypcje
Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Łańcuchy tematów i filtry tematów w klientach MQTT
Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Identyfikator klienta

Identyfikator klienta to 23 bajtowy łańcuch, który identyfikuje klienta MQTT . Każdy identyfikator musi być unikalny w obrębie wszystkich połączonych w danym momencie klientów. Identyfikator musi zawierać tylko znaki poprawne w nazwie menedżera kolejek. W ramach tych ograniczeń można użyć dowolnego łańcucha identyfikacyjnego. Ważne jest, aby procedura przydzielania identyfikatorów klientów była procedurą, a także sposób konfigurowania klienta z wybranym identyfikatorem.

Identyfikator klienta jest używany w administrowaniu systemem MQTT . Z potencjalnie setkami tysięcy klientów do administrowania, konieczne jest szybkie zidentyfikowanie konkretnego klienta. Załóżmy na przykład, że urządzenie nie działa, a użytkownik jest powiadamiany, być może przez klienta dzwoniący do stanowiska pomocy. W jaki sposób klient identyfikuje urządzenie i w jaki sposób korelować tę identyfikację z serwerem, który jest zwykle połączony z klientem? Czy konieczne jest zapoznanie się z bazą danych, która odwzorowuje każde urządzenie na identyfikator klienta i na serwer? Czy nazwa urządzenia identyfikuje serwer, do którego jest przyłączony? Podczas przeglądania za pośrednictwem połączeń klienckich MQTT każde połączenie jest oznaczone identyfikatorem klienta. Czy chcesz wyszukać tabelę w celu odwzorowania identyfikatora klienta na urządzenie fizyczne?

Czy identyfikator klienta identyfikuje określone urządzenie, użytkownika lub aplikację działającą na kliencie? Czy jeśli klient zastąpi wadliwe urządzenie nowym urządzeniem, to czy nowe urządzenie ma ten sam identyfikator co stare urządzenie? Czy przydzielić nowy identyfikator? Jeśli zmienisz urządzenie fizyczne, ale zachowaj ten sam identyfikator, pozostałe publikacje i aktywne subskrypcje zostaną automatycznie przesłane do nowego urządzenia.

W jaki sposób identyfikatory klientów są unikalne? Podobnie jak w przypadku systemu generowania unikalnych identyfikatorów, należy mieć niezawodny proces ustawiania identyfikatora na kliencie. Być może urządzenie klienckie to "black-box", bez interfejsu użytkownika. Czy wytwarza się urządzenie o identyfikatorze klienta-takie jak przy użyciu jego adresu MAC? A może jest to proces instalacji i konfiguracji oprogramowania, który konfiguruje urządzenie przed jego aktywowaniem?

Identyfikator klienta można utworzyć na podstawie adresu MAC urządzenia 48-bitowego, aby identyfikator był krótki i unikalny. Jeśli wielkość transmisji nie jest newralgicznym problemem, można użyć pozostałych 17 bajtów, aby ułatwić administrowanie adresem.

Pojęcia pokrewne

Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT
W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Czyste sesje
Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Znaczniki dostawy

Publikacja ostatniego testamentu i testamentu
Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Trwałość komunikatu w klientach MQTT

Publikacje
Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

Jakość usług świadczonych przez klienta MQTT
Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Zachowane publikacje i klienty MQTT
Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Subskrypcje
Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Łańcuchy tematów i filtry tematów w klientach MQTT
Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Znaczniki dostawy

Gdy klient publikuje w temacie nowy znacznik dostarczania, zostanie utworzony. Za pomocą znacznika dostarczania można monitorować dostarczanie publikacji lub blokować aplikację kliencką do czasu zakończenia dostawy.

Token jest obiektem `MqttDeliveryToken` . Jest on tworzony przez wywołanie metody MqttTopic.publish () i jest zachowywany przez klienta MQTT do momentu, gdy sesja klienta zostanie rozłączona i dostarczenie zostanie zakończone.

Normalny sposób użycia tokenu polega na sprawdzeniu, czy dostawa jest kompletna. Zablokuj aplikację kliencką do czasu zakończenia dostarczania za pomocą zwróconego znacznika do wywołania `token.waitForCompletion`. Alternatywnie można podać procedurę obsługi produktu `MqttCallBack` . Po otrzymaniu przez klienta MQTT wszystkich potwierdzeń, których oczekuje w ramach dostarczania publikacji, wywołuje on `MqttCallBack.deliveryComplete` przekazując znacznik dostarczania jako parametr.

Dopóki dostawa nie zostanie zakończona, można sprawdzić publikację za pomocą zwróconego znacznika dostarczania, wywołując komendę `token.getMessage`.

Zrealizowane dostawy

Zakończenie dostaw jest asynchroniczne i zależy od jakości usługi powiązanej z publikacją.

Najwyżej raz

`QoS=0`
Dostarczanie jest kompletne natychmiast po powrocie z produktu `MqttTopic.publish`. `MqttCallback.deliveryComplete` jest wywoływana natychmiast.

Co najmniej raz

`QoS=1`
Dostarczanie jest kompletne, gdy potwierdzenie publikacji zostało odebrane z menedżera kolejek. `MqttCallback.deliveryComplete` jest wywoływana po odebraniu potwierdzenia. Komunikat może zostać dostarczony więcej niż jeden raz przed wywołaniem programu `MqttCallback.deliveryComplete` , jeśli komunikacja jest powolna lub niewiarygodna.

Dokładnie jeden raz

`QoS=2`

Dostarczanie jest kompletne, gdy klient otrzymuje komunikat o zakończeniu, że publikacja została opublikowana w subskrybentach. `MqttCallback.deliveryComplete` jest wywoływana natychmiast po odebraniu komunikatu publikacji. Nie czeka na komunikat o zakończeniu.

W rzadkich przypadkach aplikacja kliencka może nie wrócić do klienta MQTT z `MqttCallback.deliveryComplete` normalnie. Wiadomo, że dostarczenie zostało zakończone, ponieważ wywołano program `MqttCallback.deliveryComplete` . Jeśli klient zrestartuje tę samą sesję, program `MqttCallback.deliveryComplete` nie zostanie wywołany ponownie.

Dostawy niekompletne

Jeśli dostawa nie zostanie zakończona po rozłączeniu sesji klienta, można ponownie nawiązać połączenie z klientem i zakończyć dostarczanie. Dostarczanie komunikatu można zakończyć tylko wtedy, gdy komunikat został opublikowany w sesji z atrybutem `MqttConnectionOptions` ustawionym na wartość `false`.

Utwórz klienta przy użyciu tego samego identyfikatora klienta i adresu serwera, a następnie nawiąże połączenie, ustawiając ponownie atrybut `cleanSession` `MqttConnectionOptions` na wartość `false` . Jeśli parametr `cleanSession` zostanie ustawiony na wartość `true`, oczekujące tokeny dostarczania zostaną odrzucone.

Istnieje możliwość sprawdzenia, czy istnieją oczekujące dostawy, wywołując komendę `MqttClient.getPendingDeliveryTokens`. Program `MqttClient.getPendingDeliveryTokens` można wywołać przed nawiązaniem połączenia z klientem.

Pojęcia pokrewne

Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT
W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Czyste sesje
Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Identyfikator klienta

Publikacja ostatniego testamentu i testamentu
Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Trwałość komunikatu w klientach MQTT

Publikacje
Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

Jakość usług świadczonych przez klienta MQTT
Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Zachowane publikacje i klienty MQTT
Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Subskrypcje
Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Łańcuchy tematów i filtry tematów w klientach MQTT
Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Publikacja ostatniego testamentu i testamentu

Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Utwórz temat dla ostatniej testamentu i testamentu. Użytkownik może utworzyć temat, taki jak `MQTTManagement/Connections/`*`server URI`*`/`*`client identifer`*`/Lost`.

Skonfiguruj "ostatni testament i testament" przy użyciu metody `MqttConnectionOptions.setWill(MqttTopic lastWillTopic, byte [] lastWillPayload, int lastWillQos, boolean lastWillRetained)`.

Rozważ utworzenie znacznika czasu w komunikacie `lastWillPayload` . Dołączanie innych informacji o kliencie, które pomagają w identyfikacji klienta i okolicznościach połączenia. Przekaz obiekt `MqttConnectionOptions` do konstruktora `MqttClient` .

Set `lastWillQos` to 1 or 2, to make the message persistent in WebSphere MQ, and to guarantee delivery. Aby zachować ostatnie utracone informacje o połączeniu, ustaw wartość `lastWillRetained` na `true`.

Publikacja "last will and testament" jest wysyłana do subskrybentów, jeśli połączenie zostanie nieoczekiwanie zakończone. Jest on wysyłany, jeśli połączenie kończy się bez wywołania metody `MqttClient.disconnect` przez klienta.

Aby monitorować połączenia, należy uzupełnić publikację "ostatni testament i testament" z innymi publikacjami, aby rejestrować połączenia i programowane rozłączenia.

Pojęcia pokrewne
Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT
W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Czyste sesje
Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Identyfikator klienta
Znaczniki dostawy
Trwałość komunikatu w klientach MQTT
Publikacje

Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

Jakość usług świadczonych przez klienta MQTT
Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Zachowane publikacje i klienty MQTT
Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Subskrypcje
Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Łańcuchy tematów i filtry tematów w klientach MQTT
Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Trwałość komunikatu w klientach MQTT

Komunikaty publikacji są trwałe, jeśli są wysyłane z jakością usługi "co najmniej raz" lub "dokładnie jeden raz". Można zaimplementować własny mechanizm utrwalania na kliencie lub użyć domyślnego mechanizmu utrwalania udostępnianego wraz z klientem. Trwałość działa w obu kierunkach, w przypadku publikacji wysyłanych do klienta lub z niego.

W produkcie MQTTtrwałość komunikatu ma dwa aspekty: sposób przesyłania komunikatu oraz informację o tym, czy jest on umieszczany w kolejce w produkcie IBM MessageSight , jak i w produkcie IBM WebSphere MQ jako komunikat trwały.

1. Trwałość komunikatu dla par klienta MQTT z jakością usługi. W zależności od tego, jaką jakość usługi wybierzesz dla komunikatu, komunikat jest trwały. Trwałość komunikatu jest niezbędna do zaimplementowania wymaganej jakości usługi.

   Jeśli zostanie określona wartość "at most once" (najwyżej raz), `QoS=0`(klient), klient odrzuci komunikat natychmiast po jego opublikowaniu. Jeśli w procesie przetwarzania komunikatu wystąpi awaria, komunikat nie zostanie wysłany ponownie. Nawet jeśli klient nadal będzie aktywny, komunikat nie zostanie wysłany ponownie. Zachowanie komunikatów produktu `QoS=0` jest takie samo, jak szybkie komunikaty nietrwałe produktu IBM WebSphere MQ .

   Jeśli komunikat jest publikowany przez klienta z funkcją `QoS` 1 lub 2, jest on trwały. Komunikat jest przechowywany lokalnie i usuwany z klienta tylko wtedy, gdy nie jest już potrzebny do zagwarantowania "co najmniej raz", `QoS=1`lub "dokładnie raz", `QoS=2`, dostarczania.
2. Jeśli komunikat jest oznaczony jako `QoS` 1 lub 2, jest on umieszczany w kolejce w IBM MessageSight i IBM WebSphere MQ jako komunikat trwały. Jeśli jest ona oznaczona jako `QoS=0`, to jest umieszczana w kolejce IBM MessageSight i IBM WebSphere MQ jako komunikat nietrwały. W produkcie IBM WebSphere MQ komunikaty nietrwałe są przesyłane między menedżerami kolejek "dokładnie jeden raz", chyba że dla kanału komunikatów atrybut `NPMSPEED` jest ustawiony na wartość `FAST`.

Trwała publikacja jest przechowywana na kliencie, dopóki nie zostanie odebrana przez aplikację kliencką. W przypadku systemu `QoS=2`publikacja jest odrzucana od klienta, gdy wywołanie zwrotne aplikacji zwraca element sterujący. W przypadku `QoS=1` aplikacja może otrzymać publikację ponownie, jeśli wystąpi awaria. W przypadku systemu `QoS=0`wywołanie zwrotne odbiera publikację nie więcej niż raz. Może ona nie otrzymać publikacji, jeśli wystąpi awaria lub jeśli klient został odłączony w momencie publikacji.

Po zasubskrybowaniu tematu można zmniejszyć QoS , za pomocą którego subskrybent odbiera komunikaty, aby dopasować jego możliwości trwałości. Publikacje utworzone w wyższej jakości QoS są wysyłane z najwyższą jakością QoS , o którą zażądał subskrybent.

Zapisywanie komunikatów

Implementacja przechowywania danych na małych urządzeniach zmienia się bardzo wiele. Model tymczasowego zapisywania trwałych komunikatów w pamięci masowej, który jest zarządzany przez klienta MQTT , może być zbyt wolny lub wymagać zbyt dużej ilości pamięci masowej. W urządzeniach przenośnych mobilny system operacyjny może udostępniać usługę pamięci masowej, która jest idealna dla komunikatów MQTT .

Aby zapewnić elastyczność w spełnianiu ograniczeń dotyczących małych urządzeń, klient MQTT ma dwa interfejsy trwałości. Interfejsy definiują operacje, które są związane z zapisywaniem trwałych komunikatów. Interfejsy są opisane w dokumentacji interfejsu API dla Klient MQTT dla produktu Java. W celu uzyskania odsyłaczy do dokumentacji interfejsu API klienta dla bibliotek klienta MQTT należy zapoznać się z informacjami w sekcji Skorowidz programistyczny klienta MQTT. Interfejsy można zaimplementować w taki sposób, aby odpowiadał urządzeniu. Klient MQTT , który działa w środowisku Java SE, ma domyślną implementację interfejsów, które przechowują trwałe komunikaty w systemie plików. Używa pakietu `java.io` . Klient ma również domyślną implementację dla mechanizmu Java ME, `MqttDefaultMIDPPersistence`.

Klasy trwałości

`MqttClientPersistence`
Przekaz instancji implementacji produktu `MqttClientPersistence` do klienta MQTT jako parametr konstruktora `MqttClient` . Jeśli parametr `MqttClientPersistence` zostanie pominięty z konstruktora `MqttClient` , klient MQTT zapisuje komunikaty trwałe przy użyciu klasy `MqttDefaultFilePersistence` lub `MqttDefaultMIDPPersistence`.

`MqttPersistable`
Program `MqttClientPersistence` pobiera i umieszcza obiekty `MqttPersistable` przy użyciu klucza pamięci masowej. Należy udostępnić implementację produktu `MqttPersistable` oraz implementację produktu `MqttClientPersistence` , jeśli nie jest używany produkt `MqttDefaultFilePersistence` lub `MqttDefaultMIDPPersistence`.

`MqttDefaultFilePersistence`

Klient MQTT udostępnia klasę `MqttDefaultFilePersistence` . W przypadku tworzenia instancji produktu `MqttDefaultFilePersistence` w aplikacji klienckiej można udostępnić katalog, w którym będą przechowywane komunikaty trwałe jako parametr konstruktora `MqttDefaultFilePersistence` .

Alternatywnie klient MQTT może utworzyć instancję produktu `MqttDefaultFilePersistence` i umieścić pliki w katalogu domyślnym. Nazwa katalogu to *`client identifier`*`-tcp` *`hostname portnumber`*. Produkty `"\", "\\", "/", ":"` i `" "` są usuwane z łańcucha nazwy katalogu.

Ścieżka do katalogu jest wartością właściwości systemowej `rcp.data`. Jeśli parametr `rcp.data` nie jest ustawiony, to ścieżka jest wartością właściwości systemowej `usr.data`.

`rcp.data` to właściwość powiązana z instalacją platformy OSGi lub Eclipse Rich Client Platform (RCP).

`usr.data` to katalog, w którym uruchomiono komendę Java, która uruchomiła aplikację.

`MqttDefaultMIDPPersistence`
`MqttDefaultMIDPPersistence` ma konstruktor domyślny i nie ma żadnych parametrów. Używa on pakietu `javax.microedition.rms.RecordStore` do przechowywania komunikatów.

Pojęcia pokrewne

Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT
W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera.

Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Czyste sesje
Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Identyfikator klienta

Znaczniki dostawy

Publikacja ostatniego testamentu i testamentu
Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Publikacje
Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

Jakość usług świadczonych przez klienta MQTT
Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Zachowane publikacje i klienty MQTT
Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Subskrypcje
Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Łańcuchy tematów i filtry tematów w klientach MQTT
Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Publikacje

Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

`MqttMessage` ma tablicę bajtów jako jej ładunek. Dążą do tego, aby wiadomości były jak najmniejsze. Maksymalna długość komunikatu dozwolona przez protokół MQTT wynosi 250 MB.

Zwykle program kliencki MQTT używa produktu `java.lang.String` lub `java.lang.StringBuffer` do manipulowania treścią komunikatu. Dla wygody klasa `MqttMessage` ma metodę `toString` , która umożliwia przekształcenie jej ładunku w łańcuch. Aby utworzyć ładunek tablicy bajtów z poziomu `java.lang.String` lub `java.lang.StringBuffer`, należy użyć metody `getBytes` .

Metoda `getBytes` przekształca łańcuch w domyślny zestaw znaków dla platformy. Domyślnym zestawem znaków jest zazwyczaj UTF-8. Publikacje dotyczące produktu MQTT , które zawierają tylko tekst, są

zwykle kodowane w produkcie UTF-8. Użyj metody `getBytes("UTF8")` , aby przesłonić domyślny zestaw znaków.

W produkcie IBM WebSphere MQpublikacja MQTT jest odbierana w postaci komunikatu `jms-bytes` . Komunikat zawiera folder MQRFH2 zawierający <mqtt>oraz folder <mqps> . Folder <mqtt> zawiera elementy `clientId` i `qos`, ale ta treść może ulec zmianie w przyszłości.

`MqttMessage` ma trzy dodatkowe atrybuty: jakość usługi, bez względu na to, czy jest ona zachowywana, oraz czy jest duplikatem. Flaga duplikowania jest ustawiana tylko wtedy, gdy jakość usługi jest "co najmniej raz" lub "dokładnie jeden raz". Jeśli komunikat został wcześniej wysłany i nie został uznany za wystarczająco szybko przez klienta MQTT , komunikat zostanie wysłany ponownie z zduplikowanym atrybutem ustawionym na wartość `true`.

Publikowanie

Aby utworzyć publikację w aplikacji klienckiej MQTT , należy utworzyć `MqttMessage`. Ustaw jego ładunek, jakość usługi i to, czy jest on zachowywany, a następnie wywołaj metodę `MqttTopic.publish(MqttMessage message)` ; `MqttDeliveryToken` jest zwracany, a zakończenie publikacji jest asynchroniczne.

Alternatywnie klient MQTT może utworzyć tymczasowy obiekt komunikatu dla użytkownika z parametrów metody `MqttTopic.publish(byte [] payload, int qos, boolean retained)` podczas tworzenia publikacji.

Jeśli w publikacji znajduje się "co najmniej raz" lub "dokładnie jeden raz" jakość usług, `QoS=1` lub `QoS=2`, klient MQTT wywołuje interfejs `MqttClientPersistence` . Wywołuje on produkt `MqttClientPersistence` w celu zapisania komunikatu przed zwróceniem znacznika dostarczenia do aplikacji.

Aplikacja może zablokować do momentu dostarczenia wiadomości do serwera za pomocą metody `MqttDeliveryToken.waitForCompletion` . Alternatywnie, aplikacja może być kontynuowana bez blokowania. Jeśli chcesz sprawdzić, czy publikacje są dostarczane bez blokowania, zarejestruj instancję klasy zwrotnej, która implementuje `MqttCallback` z klientem MQTT . Klient MQTT wywołuje metodę `MqttCallback.deliveryComplete` od razu po dostarczyniu publikacji. W zależności od jakości usługi dostawa może być niemal natychmiastowa w przypadku produktu `QoS=0`lub może potrwać pewien czas w przypadku produktu `QoS=2`.

Użyj metody `MqttDeliveryToken.isComplete` , aby odpytać, czy dostawa jest kompletna. Jeśli wartością parametru `MqttDeliveryToken.isComplete` jest `false`, można wywołać `MqttDeliveryToken.getMessage` , aby pobrać treść wiadomości. Jeśli wynikiem wywołania programu `MqttDeliveryToken.isComplete` jest `true`, komunikat został usunięty, a wywołanie metody `MqttDeliveryToken.getMessage` spowodowałoby zgłoszenie wyjątku pustego wskaźnika. Nie ma wbudowanej synchronizacji między `MqttDeliveryToken.getMessage` i `MqttDeliveryToken.isComplete`.

Jeśli klient rozłącza się przed odebraniem wszystkich oczekujących tokenów dostarczania, nowa instancja klienta może wysłać zapytanie do tokenów dostarczania oczekujących przed nawiązaniem połączenia. Dopóki klient nie połączy się, nie zostaną wykonane żadne nowe dostawy i można bezpiecznie zadzwonić do produktu `MqttDeliveryToken.getMessage`. Użyj metody `MqttDeliveryToken.getMessage` , aby dowiedzieć się, które publikacje nie zostały dostarczone. Oczekujące tokeny dostarczania są usuwane w przypadku nawiązania połączenia z `MqttConnectOptions.cleanSession` ustawionym na wartość domyślną `true`.

subskrypcja

Menedżer kolejek lub IBM MessageSight jest odpowiedzialny za tworzenie publikacji, które mają być wysyłane do subskrybenta MQTT . Menedżer kolejek sprawdza, czy filtr tematu w subskrypcji utworzonej przez klienta MQTT jest zgodny z łańcuchem tematu w publikacji. Zgodność może być dokładnym dopasowaniu lub może zawierać znaki wieloznaczne. Zanim opublikowanie zostanie przekazane do subskrybenta przez menedżera kolejek, menedżer kolejek sprawdza atrybuty tematu powiązane z publikacją. Jest ona zgodna z procedurą wyszukiwania opisaną w sekcji Subskrybowanie łańcucha

tematu zawierającego znaki wieloznaczne w celu określenia, czy obiekt tematu administracyjnego nadaje uprawnienia użytkownika do subskrybowania.

Gdy klient MQTT otrzymuje publikację z "co najmniej raz" jakością usługi, wywołuje metodę `MqttCallback.messageArrived` w celu przetworzenia publikacji. Jeśli jakość usługi publikacji to "dokładnie jeden raz", `QoS=2`, klient MQTT wywołuje interfejs `MqttClientPersistence` w celu zapisania komunikatu po jego odebraniu. Następnie wywołuje on `MqttCallback.messageArrived`.

Pojęcia pokrewne

Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT
W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Czyste sesje
Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Identyfikator klienta

Znaczniki dostawy

Publikacja ostatniego testamentu i testamentu
Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Trwałość komunikatu w klientach MQTT

Jakość usług świadczonych przez klienta MQTT
Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Zachowane publikacje i klienty MQTT
Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Subskrypcje
Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Łańcuchy tematów i filtry tematów w klientach MQTT
Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Jakość usług świadczonych przez klienta MQTT

Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Jakość usługi publikacji jest atrybutem produktu `MqttMessage`. Jest ona ustawiana przy użyciu metody `MqttMessage.setQos`.

Metoda `MqttClient.subscribe` może zmniejszyć jakość usługi stosowaną do publikacji wysyłanych do klienta w temacie. Jakość usługi publikacji przekazanej abonentowi może być inna niż jakość usługi w publikacji. Niższa z tych dwóch wartości jest używana do przekazywania publikacji.

Najwyżej raz
`QoS=0`

Komunikat jest dostarczany nie więcej niż jeden raz lub nie jest dostarczany w ogóle. Dostarczenie komunikatu za pośrednictwem sieci nie jest potwierdzane.

Komunikat nie jest zapisany. Komunikat może zostać utracony, jeśli klient zostanie rozłączony lub nastąpi awaria serwera.

`QoS=0` jest najszybszym trybem przesyłania. Czasami nazywa się "ogień i zapomnij".

Protokół MQTT nie wymaga, aby serwery przekazywały publikacje pod adresem `QoS=0` do klienta. Jeśli klient zostanie odłączony od chwili otrzymania przez serwer publikacji, może ona zostać odrzucona w zależności od serwera. Usługa telemetryczna (MQXR) nie usuwa komunikatów wysłanych z produktem `QoS=0`. Są one zapisywane jako komunikaty nietrwałe i są usuwane tylko w przypadku zatrzymania menedżera kolejek.

Co najmniej raz
`QoS=1`

`QoS=1` jest domyślnym trybem przesyłania.

Komunikat jest zawsze dostarczany co najmniej raz. Jeśli nadawca nie otrzyma potwierdzenia, komunikat zostanie wysłany ponownie z ustawioną opcją `DUP` , dopóki nie zostanie odebrany potwierdzenie. Odbiornik wyników może wielokrotnie wysyłać ten sam komunikat i może przetwarzać go wiele razy.

Komunikat musi być przechowywany lokalnie u nadawcy i odbiorcy, dopóki nie zostanie przetworzony.

Komunikat zostanie usunięty z odbiornika po przetworzeniu komunikatu. Jeśli odbiornik jest brokerem, komunikat jest publikowany do jego subskrybentów. Jeśli odbiorcą jest klient, komunikat jest dostarczany do aplikacji subskrybenta. Po usunięciu wiadomości odbiorca wysyła potwierdzenie do nadawcy.

Wiadomość zostanie usunięta od nadawcy po odebraniu potwierdzenia od odbiorcy.

Dokładnie jeden raz
`QoS=2`

Komunikat jest zawsze dostarczany dokładnie jeden raz.

Komunikat musi być przechowywany lokalnie u nadawcy i odbiorcy, dopóki nie zostanie przetworzony.

`QoS=2` jest najbezpieczniejszym, ale najwolniejszym trybem przesyłania. Przed usunięciem komunikatu z nadajnika pobiera ona co najmniej dwie pary transmisji między nadawcą i odbiorcą. Komunikat może być przetwarzany u odbiorcy po pierwszej transmisji.

W pierwszej parze transmisji nadawca przesyła wiadomość i otrzymuje potwierdzenie od odbiorcy, że zapisał komunikat. Jeśli nadawca nie otrzyma potwierdzenia, komunikat zostanie wysłany ponownie z ustawioną opcją `DUP` , dopóki nie zostanie odebrany potwierdzenie.

W drugiej parze transmisji nadawca informuje odbiorcę o tym, że może zakończyć przetwarzanie komunikatu `"PUBREL"`. Jeśli nadawca nie otrzyma potwierdzenia komunikatu `"PUBREL"` , komunikat `"PUBREL"` zostanie wysłany ponownie, dopóki nie zostanie odebrany potwierdzenie. Nadawca usuwa komunikat, który został zapisany po odebraniu potwierdzenia do komunikatu produktu `"PUBREL"` .

Odbiornik może przetworzyć komunikat w pierwszej lub drugiej fazie, pod warunkiem, że nie przeprzetwarzali komunikatu. Jeśli odbiornik jest brokerem, publikuje on komunikat do subskrybentów. Jeśli odbiorcą jest klient, dostarcza on komunikat do aplikacji subskrybenta. Odbiornik wysyła komunikat o zakończeniu z powrotem do nadawcy, który zakończył przetwarzanie komunikatu.

Pojęcia pokrewne

Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT
W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Czyste sesje
Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Identyfikator klienta

Znaczniki dostawy

Publikacja ostatniego testamentu i testamentu
Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Trwałość komunikatu w klientach MQTT

Publikacje
Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

Zachowane publikacje i klienty MQTT
Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Subskrypcje
Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Łańcuchy tematów i filtry tematów w klientach MQTT
Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Zachowane publikacje i klienty MQTT

Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Metoda `MqttMessage.setRetained` służy do określania, czy publikacja na danym temacie jest zachowywana, czy nie.

Aby usunąć zachowaną publikację w programie IBM WebSphere MQ, należy uruchomić komendę MQSC **`CLEAR TOPICSTR`** .

Jeśli zostanie utworzona publikacja z ładunkiem o wartości NULL, pusta publikacja jest przekazywana do subskrybentów. Inne brokery produktu MQTT mogą nie przesyłać pustej publikacji do subskrybentów.

Jeśli opublikujesz niezachowaną publikację do tematu, który ma zachowaną publikację, zachowana publikacja nie będzie miała wpływu na tę publikację. Aktualni subskrybenci otrzymują nową publikację. Nowi subskrybenci otrzymują po raz pierwszy zachowaną publikację, a następnie otrzymują nowe publikacje.

Po utworzeniu lub zaktualizowaniu zachowanej publikacji należy wysłać publikację za pomocą opcji QoS lub 1 lub 2. Jeśli zostanie ona wysłana z QoS z 0, program IBM WebSphere MQ tworzy nietrwałą zachowaną publikację. Jeśli menedżer kolejek zostanie zatrzymany, publikacja nie zostanie zachowana.

Użyj zachowanych publikacji, aby zarejestrować najnowszą wartość pomiaru. Nowi subskrybenci zatrzymanego tematu natychmiast otrzymują najnowszą wartość pomiaru. Jeśli nie są pobierane żadne nowe pomiary od czasu ostatniego subskrybowania subskrybenta do tematu publikacji, a subskrybent ponownie subskrybuje subskrybent, subskrybent otrzymuje najnowszą zachowaną publikację w temacie ponownie.

Pojęcia pokrewne

Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT
W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Czyste sesje
Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Identyfikator klienta

Znaczniki dostawy

Publikacja ostatniego testamentu i testamentu
Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Trwałość komunikatu w klientach MQTT

Publikacje
Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

Jakość usług świadczonych przez klienta MQTT
Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Subskrypcje
Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Łańcuchy tematów i filtry tematów w klientach MQTT
Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Subskrypcje

Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące

tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Utwórz subskrypcje przy użyciu metod `MqttClient.subscribe` , przekazując jeden lub więcej filtrów tematów oraz parametry jakości usług. Parametr jakości usługi ustawia maksymalną jakość usługi, którą subskrybent jest przygotowany do użycia w celu odebrania komunikatu. Wiadomości wysłane do tego klienta nie mogą być dostarczane z wyższą jakością usług. Jakość usługi jest ustawiana na niższą od pierwotnej wartości, gdy komunikat został opublikowany, a poziom określony dla subskrypcji. Domyślną jakością usługi dla odbierania komunikatów jest `QoS=1`, co najmniej raz.

Samo żądanie subskrypcji jest wysyłane z produktem `QoS=1`.

Publikacje są odbierane przez subskrybenta, gdy klient MQTT wywołuje metodę `MqttCallback.messageArrived` . Metoda `messageArrived` przekazuje także łańcuch tematu, z którym komunikat został opublikowany w subskrybencie.

Za pomocą metod `MqttClient.unsubscribe` można usunąć subskrypcję lub zestaw lub subskrypcje.

Komenda WebSphere MQ może usunąć subskrypcję. Lista subskrypcji za pomocą programu WebSphere MQ Explorer lub za pomocą komend **runmqsc** lub PCF. Zostaną nazwane wszystkie subskrypcje klienta MQTT. Nadano im nazwę w postaci: *ClientIdentifier*:*Topic name*

W przypadku korzystania z domyślnej klasy `MqttConnectOptions` lub w przypadku ustawienia wartości `true` na potrzeby atrybutu `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia z klientem wszystkie stare subskrypcje klienta zostaną usunięte po nawiązaniu połączenia z klientem. Wszystkie nowe subskrypcje dokonane przez klienta podczas sesji zostaną usunięte po rozłączeniu.

W przypadku ustawienia wartości `false` na potrzeby atrybutu `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia wszystkie subskrypcje tworzone przez klienta będą dodawane do wszystkich subskrypcji klienta, które istniały przed nawiązaniem połączenia. Wszystkie subskrypcje pozostaną aktywne po rozłączeniu połączenia z klientem.

Innym podejściem umożliwiającym zrozumienie wpływu atrybutu `cleanSession` na subskrypcje jest uznanie go za atrybut modalny. W domyślnym trybie atrybutu `cleanSession=true` klient tworzy subskrypcje i odbiera publikacje tylko w zasięgu sesji. W alternatywnym trybie atrybutu `cleanSession=false` subskrypcje są trwałe. Można nawiązywać połączenia z klientem i je rozłączać, a jego subskrypcje pozostają aktywne. Po ponownym nawiązaniu połączenia z klientem odbiera on wszystkie niedostarczone publikacje. Po nawiązaniu połączenia klient może modyfikować zestaw subskrypcji aktywnych na jego potrzeby.

Tryb atrybutu `cleanSession` należy ustawić przed nawiązaniem połączenia. Ten tryb będzie obowiązywać przez cały czas trwania sesji. Aby zmienić ustawienie trybu, należy rozłączyć połączenie z klientem, a następnie ponownie je nawiązać. W przypadku zmiany używanego trybu atrybutu z `cleanSession=false` na `cleanSession=true` wszystkie wcześniejsze subskrypcje klienta oraz wszystkie publikacje, które nie zostały odebrane, zostaną usunięte.

Publikacje, które są zgodne z aktywnymi subskrypcjami, są wysyłane do klienta natychmiast po ich opublikowaniu. Jeśli klient zostanie rozłączony, zostaną one wysłane do klienta, jeśli połączy się ponownie z tym samym serwerem z tym samym identyfikatorem klienta i z `MqttConnectOptions.cleanSession` ustawionym na `false`.

Subskrypcje dla konkretnego klienta są identyfikowane przez identyfikator klienta. Można ponownie podłączyć klient z innego urządzenia klienckiego do tego samego serwera i kontynuować te same subskrypcje i otrzymywać niedostarczone publikacje.

Pojęcia pokrewne

Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT
W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Czyste sesje

Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Identyfikator klienta

Znaczniki dostawy

Publikacja ostatniego testamentu i testamentu
Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Trwałość komunikatu w klientach MQTT

Publikacje
Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

Jakość usług świadczonych przez klienta MQTT
Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Zachowane publikacje i klienty MQTT
Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Łańcuchy tematów i filtry tematów w klientach MQTT
Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Łańcuchy tematów i filtry tematów w klientach MQTT

Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

Łańcuchy tematów są używane do wysyłania publikacji do subskrybentów. Utwórz łańcuch tematu przy użyciu metody `MqttClient.getTopic(java.lang.String topicString)`.

Filtry tematów są używane do subskrybowania tematów i otrzymywania publikacji. Filtry tematów mogą zawierać znaki wieloznaczne. W przypadku znaków wieloznacznych można zasubskrybować wiele tematów. Aby utworzyć filtr tematów, należy użyć metody subskrypcji, na przykład `MqttClient.subscribe(java.lang.String topicFilter)`.

Łańcuchy tematów

Składnia łańcucha tematu IBM WebSphere MQ jest opisana w sekcji Strings tematów. Składnia łańcuchów tematów produktu MQTT jest opisana w klasie `MqttClient` w dokumentacji interfejsu API dla Klient MQTT dla produktu Java. W celu uzyskania odsyłaczy do dokumentacji interfejsu API klienta dla bibliotek klienta MQTT należy zapoznać się z informacjami w sekcji Skorowidz programistyczny klienta MQTT.

Składnia każdego typu łańcucha tematu jest prawie identyczna. Istnieją cztery niewielkie różnice:

1. Łańcuchy tematów wysyłane do programu IBM WebSphere MQ przez klienty MQTT muszą być zgodne z konwencją dla nazw menedżerów kolejek. W szczególności łańcuchy tematów nie mogą zawierać łączników.

2. Maksymalna długość różni się od siebie. Łańcuchy tematów IBM WebSphere MQ są ograniczone do 10,240 znaków. Klient MQTT może tworzyć łańcuchy tematów o długości do 65535 bajtów.
3. Łańcuch tematu utworzony przez klient MQTT nie może zawierać znaku o kodzie zero.
4. W produkcie WebSphere Message Broker, poziom tematu o wartości NULL, `'...//...'` był niepoprawny. Poziomy tematów o wartości NULL są obsługiwane przez produkt IBM WebSphere MQ.

W przeciwieństwie do publikowania/subskrybowania produktu IBM WebSphere MQ protokół `mqttv3` nie zawiera pojęcia administracyjnego obiektu tematu. Nie można skonstruować łańcucha tematu z obiektu tematu i łańcucha tematu. Jednak łańcuch tematu jest odwzorowywany na temat administracyjny w produkcie IBM WebSphere MQ. Kontrola dostępu powiązana z tematem administracyjnym określa, czy publikacja jest publikowana w temacie, czy odrzucona. Atrybuty, które są stosowane do publikacji, gdy są przekazywane do subskrybentów, mają wpływ na atrybuty tematu administracyjnego.

Filtry tematów

Składnia filtru tematu produktu IBM WebSphere MQ jest opisana w sekcji Schemat znaków wieloznacznych oparty na temacie. Składnia filtrów tematów, które można skonstruować za pomocą klienta MQTT , jest opisana w klasie `MqttClient` w dokumentacji interfejsu API dla Klient MQTT dla produktu Java. W celu uzyskania odsyłaczy do dokumentacji interfejsu API klienta dla bibliotek klienta MQTT należy zapoznać się z informacjami w sekcji Skorowidz programistyczny klienta MQTT.

Składnia każdego typu filtru tematów jest prawie identyczna. Jedyna różnica polega na tym, że różne brokery MQTT interpretują filtr tematów. W produkcie WebSphere Message Broker V6wielopoziomowy znak wieloznaczny może być używany tylko na końcu filtru tematów. W produkcie IBM WebSphere MQna dowolnym poziomie drzewa tematów może być używany wielopoziomowy znak wieloznaczny, na przykład `USA/#/Dutchess County`.

Pojęcia pokrewne

Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT
W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

Czyste sesje
Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

Identyfikator klienta

Znaczniki dostawy

Publikacja ostatniego testamentu i testamentu
Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

Trwałość komunikatu w klientach MQTT

Publikacje
Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT może tworzyć publikacje, które mają być wysyłane do produktu IBM WebSphere MQ, a następnie zasubskrybować tematy w produkcie IBM WebSphere MQ MQ w celu otrzymywania publikacji.

Jakość usług świadczonych przez klienta MQTT
Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT

wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

Zachowane publikacje i klienty MQTT
Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

Subskrypcje
Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

Pojęcia dotyczące programowania w języku C

Różnice między klientem C i klientem Java dla wersji 3.1 produktu MQ Telemetry Transport zostały opisane w tym temacie. Temat uzupełnia pojęcia dotyczące klienta oraz informacje uzupełniające dotyczące języka C.

Temat jest zorganizowany w taki sam sposób, jak produkt “Pojęcia związane z programowaniem klienta MQTT” na stronie 537. Każdy nagłówek odpowiada tematowi w programie *Pojęcia związane z programowaniem klienta produktu WebSphere(r) MQ Telemetry Transport*. W sekcjach opisano różnice między klientem C a klientem Java. Niewielkie różnice w podpisach między metodami Java a funkcjami C nie są opisane.

Klient C jest najczęściej używany do implementowania uproszczonego adaptera między urządzeniem pomiarowym a demonem WebSphere MQ Telemetry dla urządzeń. Demon jest powszechnie używany jako koncentrator sieciowy między bardzo lekkimi urządzeniami telemetrycznych i usługą telemetryczną (MQXR).

Demon WebSphere MQ Telemetry dla urządzeń jest również klientem C, a różnice w jego zachowaniu z usługi telemetrycznej (MQXR) są opisane. Demon nie udostępnia implementacji usługi JAAS ani protokołu SSL dla klientów łączących się z nim.

`mqttclient.dll` i `mqttclient.lib` to 32-bitowe biblioteki Windows, które zawierają funkcje klienta dla implementacji C protokołu MQ Telemetry Transport w wersji 3.1 . 32-bitowe biblioteki produktu Linux to `libmqttclient.so` i `libmqttclient.a`. Dwa pliki nagłówkowe zawierają funkcję i inne deklaracje wymagane przez aplikacje klienckie: `MQTTClient.h` i `MQTTClientPersistence.h`. Te pliki są dostarczane wraz z instalacją produktu WebSphere MQ Telemetry.

Aby utworzyć i uruchomić klienta transportu produktu MQ Telemetry, należy skopiować te pliki na urządzenie klienckie. W przeciwieństwie do klientów WebSphere MQ , nie ma potrzeby instalowania odrębnego środowiska wykonawczego klienta.

Zapoznaj się z warunkami licencji powiązanymi z funkcją WebSphere MQ Telemetry , która zarządza łączącym się klientami MQ Telemetry Transport z produktem WebSphere MQ i demonem WebSphere MQ Telemetry dla urządzeń.

Klient C jest implementacją referencyjną w wersji 3.1 produktu MQ Telemetry Transport. Istnieje możliwość zaimplementowania własnych klientów w różnych językach odpowiednich dla różnych platform urządzeń. Szczegółowe informacje na ten temat można znaleźć w sekcji MQ Telemetry Transport format and protocol .

Identyfikator klienta MQTT

"Identyfikator klienta" na stronie 543	Identyfikator klienta to 23 bajtowy łańcuch, który identyfikuje klienta MQTT . Każdy identyfikator musi być unikalny w obrębie wszystkich połączonych w danym momencie klientów. Identyfikator musi zawierać tylko znaki poprawne w nazwie menedżera kolejek. W ramach tych ograniczeń można użyć dowolnego łańcucha identyfikacyjnego. Ważne jest, aby procedura przydzielania identyfikatorów klientów była procedurą, a także sposób konfigurowania klienta z wybranym identyfikatorem.

- Brak różnic.

Publikacje

"Publikacje" na stronie 549	Publikacje to instancje produktu `MqttMessage` , które są powiązane z łańcuchem tematu. Klient MQTT

- Funkcja zwrotna nie jest wywoływana w przypadku publikacji z "fire and forget", `QoS=0`, jakości usługi.

Znaczniki dostawy

"Znaczniki dostawy" na stronie 544	Gdy klient publikuje w temacie nowy znacznik dostarczania, zostanie utworzony. Za pomocą znacznika dostarczania można monitorować dostarczanie publikacji lub blokować aplikację kliencką do czasu zakończenia dostawy.

- Token dostarczania to `int`. Ma ona typedef produktu `MQTTClient_deliveryToken` .
- Funkcja zwrotna nie jest wywoływana w przypadku publikacji z "fire and forget", `QoS=0`, jakości usługi.

Zachowane publikacje

"Zachowane publikacje i klienty MQTT" na stronie 553	Jeśli subskrypcja została utworzona w temacie, który ma zachowaną publikację, najświeższą zachowaną publikację w tym temacie jest natychmiast przekazywana do użytkownika.

- Zachowane komunikaty są zapisywane tylko w demonie, jeśli skonfigurowano trwałość. Patrz sekcja Zapisywanie zachowanych komunikatów i subskrypcji.

  W przypadku produktu WebSphere MQjakość usługi wpływa na to, czy zachowany komunikat jest zapisywany na stałe. Jeśli klient jest przyłączony do usługi telemetrycznej, zachowywany jest komunikat "fire and forget" (ogień i zapomnij), jakość usługi `QoS=0` jest odrzucana, jeśli menedżer kolejek zostanie wyłączony.

Subskrypcje

"Subskrypcje" na stronie 554	Utwórz subskrypcje, aby zarejestrować zainteresowanie tematami publikacji przy użyciu filtru tematów. Klient może utworzyć wiele subskrypcji lub subskrypcję zawierającą filtr tematów, który korzysta ze znaków wieloznacznych, w celu zarejestrowania zainteresowania w wielu tematach. Publikacje dotyczące tematów zgodnych z filtrami są wysyłane do klienta. Subskrypcje mogą pozostać aktywne, gdy klient jest odłączony. Publikacje są wysyłane do klienta po ponownym nawiązaniu połączenia.

- Trwałe subskrypcje są zapisywane tylko w demonie, jeśli skonfigurowano trwałość. Patrz sekcja Zapisywanie zachowanych komunikatów i subskrypcji.

- Publikacje mogą być odbierane synchronicznie. Wywołaj funkcję `MQTTClient_receive` .

Wywołania zwrotne i synchronizacja

“Procedury zwrotne i synchronizacja w aplikacjach klienckich MQTT” na stronie 538	W modelu programistycznym klienta produktu MQTT są używane obszernie wątki. Wątki dekodują aplikację kliencką MQTT , o ile tylko mogą, z opóźnień w przesyłaniu komunikatów do serwera i z serwera. Publikacje, tokeny dostarczania i zdarzenia utraconych połączeń są dostarczane do metod w klasie wywołania zwrotnego, która implementuje interfejs `MqttCallback`.

- Operacja synchronizacji w kliencie C jest modalna. Wywołanie programu `MQTTClient_setCallback` powoduje umieszczenie klienta w trybie asynchronicznym.
- W trybie synchronicznym klient aplikacji musi dobrowolnie kontrolować wydajność, tak aby klient MQTT mógł przetwarzać potwierdzenia i wydać komendę ping MQTT, aby utrzymać sieć przy życiu. Kontrola wydajności poprzez wywołanie funkcji `MQTTClient_receive` lub `MQTTClient_yield`.

Łańcuchy tematów i filtry

“Łańcuchy tematów i filtry tematów w klientach MQTT” na stronie 556	Łańcuch tematu i filtry tematów są używane do publikowania i subskrybowania. Składnia łańcuchów tematów i filtrów w klientach MQTT jest w dużej mierze taka sama, jak łańcuchy tematów w produkcie IBM WebSphere MQ.

- Demon WebSphere MQ Telemetry dla urządzeń obsługuje wielopoziomowy znak wieloznaczny # inaczej niż w produkcie WebSphere MQ v7. /# musi być ostatnim dwoma znakami w łańcuchu filtru, aby produkt # zachował się jak znak wieloznaczny. W produkcie WebSphere MQ v7produkt `../#/..` jest poprawnym użyciem wielopoziomowego znaku wieloznacznego. Demon WebSphere MQ Telemetry dla urządzeń traktuje wielopoziomowy znak wieloznaczny w taki sam sposób, jak produkt WebSphere MQ Broker v6.

Jakość usług

“Jakość usług świadczonych przez klienta MQTT” na stronie 551	Klient MQTT udostępnia trzy cechy usługi umożliwiające dostarczanie publikacji do produktu WebSphere MQ oraz klienta MQTT : "co najwyżej raz", "co najmniej raz" i "dokładnie jeden raz". Gdy klient MQTT wysyła żądanie do produktu WebSphere MQ w celu utworzenia subskrypcji, żądanie jest wysyłane razem z jakością usługi "co najmniej raz".

- Brak różnic.

Trwałość komunikatu

“Trwałość komunikatu w klientach MQTT” na stronie 547	Komunikaty publikacji są trwałe, jeśli są wysyłane z jakością usługi "co najmniej raz" lub "dokładnie jeden raz". Można zaimplementować własny mechanizm utrwalania na kliencie lub użyć domyślnego mechanizmu utrwalania udostępnianego wraz z klientem. Trwałość działa w obu kierunkach, w przypadku publikacji wysyłanych do klienta lub z niego.

- Ze względu na różnice w powiązaniach językowych, należy ustawić mechanizm utrwalania komunikatów w kliencie C w następujący sposób. Wywołaj klienta MQTT C z jedną z trzech opcji ustawionych jako czwarty parametr w produkcie `MQTTClient_create`:

  `MQTTCLIENT_PERSISTENCE_DEFAULT`
  Trwałość pliku, którego szczegóły są specyficzne dla platformy klienta.

MQTTCLIENT_PERSISTENCE_NONE
Dane przechowywane są tylko w pamięci i są tracone po zatrzymaniu klienta. Demon WebSphere MQ Telemetry dla urządzeń obsługuje tylko tę opcję.

MQTTCLIENT_PERSISTENCE_USER
Istnieje możliwość tworzenia funkcji w celu zaimplementowania własnego mechanizmu trwałości. Przekaz strukturę, `MQTTClient_persistence` , która zawiera wskaźniki do funkcji w wywołaniu `MQTTClient_create` . Szczegółowe informacje można znaleźć w informacjach referencyjnych klienta produktu MQTT C.

Czyste sesje

"Czyste sesje" na stronie 541	Klient MQTT i usługa telemetryczna (MQXR) obsługują informacje o stanie sesji. Informacje o stanie są wykorzystywane w celu zapewnienia "co najmniej raz" i "dokładnie raz" dostarczenia, oraz "dokładnie raz" otrzymania publikacji. Stan sesji obejmuje również subskrypcje utworzone przez klienta MQTT. Użytkownik może wybrać opcję uruchomienia klienta MQTT z zachowaniem informacji o stanie między sesjami lub bez niego. Zmień tryb czyszczenia sesji, ustawiając wartość `MqttConnectOptions.cleanSession` przed nawiązaniem połączenia.

- Brak różnic.

Ostatnia wola i testament

"Publikacja ostatniego testamentu i testamentu" na stronie 546	Jeśli połączenie klienta MQTT niespodziewanie kończy się, można skonfigurować produkt WebSphere MQ Telemetry w taki sposób, aby wysyłał publikację "ostatni testament i testament". Należy wstępnie zdefiniować treść publikacji oraz temat, do którego ma zostać wysłana. Właściwość "last will and testament" jest właściwością połączenia. Utwórz go przed nawiązaniem połączenia z klientem.

- Brak różnic.

Obsługa błędów programu

Te informacje wyjaśniają błędy związane z wywołaniami MQI aplikacji, gdy wywołuje połączenie, lub gdy jego komunikat jest dostarczany do miejsca docelowego.

Jeśli jest to możliwe, menedżer kolejek zwraca wszystkie błędy, gdy tylko zostanie wykonane wywołanie MQI. Są to *błędy określone lokalnie*.

Podczas wysyłania komunikatów do kolejki zdalnej błędy mogą nie być widoczne podczas wywołania MQI. W tym przypadku menedżer kolejek, który identyfikuje błędy, zgłasza je, wysyłając kolejny komunikat do programu inicjującego. Są to *zdalnie określone błędy*.

Błędy określane lokalnie

Informacje o lokalnie określonych błędach, które obejmują: niepowodzenie wywołania MQI, przerwania systemowe i komunikaty zawierające niepoprawne dane.

Trzy najczęstszych przyczyn błędów, które menedżer kolejek może zgłosić natychmiast, są następujące:

- Niepowodzenie wywołania MQI; na przykład, ponieważ kolejka jest pełna
- Przerwa w uruchamianiu pewnej części systemu, od której zależy aplikacja, na przykład menedżer kolejek
- Komunikaty zawierające dane, których nie można pomyślnie przetworzyć

Jeśli używany jest asynchroniczny obiekt put, błędy nie są zgłaszane natychmiast. Użyj wywołania MQSTAT, aby pobrać informacje o statusie poprzednich asynchronicznych operacji put.

Niepowodzenie wywołania MQI

Menedżer kolejek może natychmiast zgłosić jakiekolwiek błędy w kodzie wywołania MQI. Wykonuje to za pomocą zestawu predefiniowanych kodów powrotu. Są one podzielone na kody zakończenia i kody przyczyny.

Aby określić, czy wywołanie powiodło się, menedżer kolejek zwraca *kod zakończenia* po zakończeniu wywołania. Istnieją trzy kody zakończenia, wskazujące powodzenie, częściowe zakończenie i niepowodzenie wywołania. Menedżer kolejek zwraca także *kod przyczyny* , który wskazuje przyczynę częściowego zakończenia lub niepowodzenia wywołania.

Kody zakończenia i przyczyny dla każdego wywołania są wyświetlane wraz z opisem tego wywołania w sekcji Kody powrotu. Więcej szczegółowych informacji, w tym pomysłów na działania naprawcze, można znaleźć w:

- Kody przyczyn dla wszystkich pozostałych platform WebSphere MQ

Zaprojektuj programy tak, aby obsługiwały wszystkie kody powrotu, które mogą pojawić się w każdym wywołaniu.

Przerwy w systemie

Jeśli menedżer kolejek, z którym jest on połączony, musi odzyskać od awarii systemu, aplikacja użytkownika może nie mieć żadnych informacji o przerwaniu. Należy jednak zaprojektować aplikację, aby upewnić się, że dane nie zostaną utracone, jeśli wystąpi taka przerwa.

Metody, których można użyć w celu upewnia się, że dane pozostają spójne, zależy od platformy, na której działa menedżer kolejek:

Systemy UNIX, Linux i Windows
W tych środowiskach można wykonywać wywołania MQPUT i MQGET w zwykły sposób, ale punkty synchronizacji muszą być deklarowane za pomocą wywołań MQCMIT i MQBACK (patrz sekcja “Zatwierdzanie i wycofywanie jednostek pracy” na stronie 331). W środowisku CICS komendy MQCMIT i MQBACK są wyłączone, ponieważ można tworzyć wywołania MQPUT i MQGET w ramach jednostek pracy zarządzanych przez program CICS.

Użyj trwałych komunikatów do przenoszenia wszystkich danych, których nie można sobie pozwolić na utratę danych. Komunikaty trwałe są przywracane w kolejkach, jeśli menedżer kolejek ma do odtworzenia po awarii. Jeśli produkt WebSphere MQ jest używany w systemach UNIX, Linuxi Windows , wywołanie MQGET lub MQPUT w aplikacji nie powiedzie się w momencie zapełnienia wszystkich plików dziennika, za pomocą komunikatu MQRC_RESOURCE_PROBLEM. Więcej informacji na temat plików dziennika w systemach AIX, HP-UX, Linux, Solaris i Windows zawiera sekcja Administrowanie.

Jeśli menedżer kolejek jest zatrzymany przez operatora podczas działania aplikacji, zwykle używana jest opcja wygaszania. Menedżer kolejek przechodzi w stan wygaszania, w którym aplikacje mogą kontynuować pracę, ale muszą zakończyć się tak szybko, jak jest to wygodne. Małe, szybkie aplikacje mogą prawdopodobnie ignorować stan wygaszania i kontynuować aż do zakończenia pracy w normalnym trybie. Dłuższe działające aplikacje lub te, które oczekują na nadejście komunikatów, powinny używać opcji *fail if quiescing* (niepowodzenie w przypadku wygaszania), gdy są używane wywołania MQOPEN, MQPUT, MQPUT1i MQGET. Te opcje oznaczają, że wywołania nie powiodą się, gdy menedżer kolejek wygaśnie, ale aplikacja może nadal mieć czas na zakończenie czyszczenia, wydając wywołania, które ignorują stan wygaszania. Takie aplikacje mogą również zatwierdzić lub wycofać zmiany, które zostały wprowadzone, a następnie zakończyć.

Jeśli menedżer kolejek jest zmuszony do zatrzymania (czyli zatrzymania bez wygaszania), aplikacje odbierają kod przyczyny MQRC_CONNECTION_BROKEN podczas tworzenia wywołań MQI. Wyjdź z aplikacji lub, alternatywnie, w systemach UNIX, Linuxi Windows , wywołaj wywołanie MQDISC.

Komunikaty zawierające niepoprawne dane

Jeśli w aplikacji używane są jednostki pracy, jeśli program nie może pomyślnie przetworzyć komunikatu, który pobiera z kolejki, zostanie utworzona kopia zapasowa wywołania MQGET.

Menedżer kolejek przechowuje licznik (w polu *BackoutCount* deskryptora komunikatu) liczby przypadków, które się zdarzy. Ta liczba jest zachowana w deskryptorze każdego komunikatu, który ma wpływ na działanie. Licznik ten może dostarczyć cennych informacji na temat efektywności aplikacji. Komunikaty z licznymi wycofami, które zwiększają się w czasie, są wielokrotnie odrzucane; projektuj aplikację tak, aby analizował przyczyny tego działania i odpowiednio obsługuje takie komunikaty.

W produkcie WebSphere MQ dla systemów Windows, UNIX i Linux liczba wycofań zawsze jest zachowana po restarcie menedżera kolejek.

Korzystanie z komunikatów raportu w celu określenia problemu

Menedżer kolejek zdalnych nie może zgłaszać błędów, takich jak niepowodzenie umieszczenia komunikatu w kolejce podczas wywoływania interfejsu MQI, ale może wysłać komunikat raportu, aby stwierdzić, w jaki sposób został przetworzony komunikat.

W ramach aplikacji można tworzyć (MQPUT) komunikaty raportów, a także wybierać opcję ich odbierania (w takim przypadku są one wysyłane przez inną aplikację lub przez menedżera kolejek).

Tworzenie komunikatów raportu

Komunikaty raportów umożliwiają aplikacji poinformowanie innej aplikacji o tym, że nie może zajmować się wysłanym komunikatem.

Jednak początkowo pole *Report* musi być analizowane w celu określenia, czy aplikacja, która wysłała komunikat, jest zainteresowana informowaniu o problemach. Po ustaleniu, że wymagany jest komunikat raportu, należy podjąć decyzję:

- Określa, czy ma zostać dołączona cała oryginalna wiadomość, tylko pierwsze 100 bajtów danych, czy też żaden z pierwotnych komunikatów.
- Co zrobić z oryginalnym komunikatem. Można je odrzucić lub pozwolić na przejście do kolejki niedostarczonych komunikatów.
- Określa, czy zawartość pól *MsgId* i *CorrelId* jest również potrzebna.

Użyj pola *Feedback* , aby wskazać przyczynę wygenerowania komunikatu raportu. Umieszczanie komunikatów raportu w kolejce odpowiedzi aplikacji. Więcej informacji na ten temat zawiera sekcja Opinia .

Żądanie i odbieranie (MQGET) komunikatów raportów

Po wysłaniu wiadomości do innej aplikacji użytkownik nie jest informowany o żadnych problemach, chyba że zostanie zakończone pole *Report* w celu wskazania, że wymagana jest opinia. Informacje na temat dostępnych opcji zawiera sekcja Struktura pola raportu .

Menedżery kolejek zawsze umieszczają komunikaty raportów w kolejce odpowiedzi aplikacji, dlatego zalecane jest, aby własne aplikacje były takie same. Jeśli używany jest mechanizm komunikatów raportu, należy określić nazwę kolejki odpowiedzi w deskryptorze komunikatu komunikatu. W przeciwnym razie wywołanie MQPUT nie powiedzie się.

Aplikacja musi zawierać procedury, które monitorują kolejkę odpowiedzi i przetwarzali wszystkie komunikaty, które są na nim przesyłane. Należy pamiętać, że komunikat raportu może zawierać całą oryginalną wiadomość, pierwsze 100 bajtów oryginalnego komunikatu lub żaden z oryginalnych komunikatów.

Menedżer kolejek ustawia pole *Feedback* komunikatu raportu w celu wskazania przyczyny błędu. Na przykład kolejka docelowa nie istnieje. Programy powinny być takie same.

Aby uzyskać więcej informacji na temat komunikatów raportów, zobacz: “Komunikaty raportów” na stronie 11.

Zdalnie określone błędy

Podczas wysyłania komunikatów do kolejki zdalnej, nawet jeśli lokalny menedżer kolejek przetworzy wywołanie MQI bez znalezienia błędu, inne czynniki mogą mieć wpływ na sposób obsługi komunikatu przez zdalny menedżer kolejek.

Na przykład kolejka kierowana przez użytkownika może być pełna lub nawet nie istnieje. Jeśli komunikat musi być obsługiwany przez inne pośrednie menedżery kolejek na trasie do kolejki docelowej, to każdy z tych menedżerów może znaleźć błąd.

Problemy z dostarczeniem komunikatu

Gdy wywołanie MQPUT nie powiedzie się, można spróbować ponownie umieścić komunikat w kolejce, zwrócić go do nadawcy lub umieścić w kolejce niedostarczonych komunikatów.

Każda opcja ma swoje zalety, ale może nie być konieczne ponowne umieszczanie komunikatu, jeśli przyczyną niepowodzenia operacji MQPUT było to, że kolejka docelowa była pełna. W tym przypadku umieszczenie go w kolejce niedostarczonych komunikatów pozwala na dostarczenie go do właściwej kolejki docelowej w późniejszym czasie.

Ponowna próba dostarczenia komunikatu

Zanim komunikat zostanie umieszczony w kolejce niedostarczonych komunikatów, zdalny menedżer kolejek próbuje ponownie umieścić komunikat w kolejce, jeśli atrybuty *MsgRetryCount* i *MsgRetryInterval* zostały ustawione dla kanału, lub jeśli istnieje program obsługi wyjścia ponawiania, który ma być używany (którego nazwa jest wstrzymana w polu *MsgRetryExitId* atrybutu kanału).

Jeśli pole *MsgRetryExitId* jest puste, używane są wartości z atrybutów *MsgRetryCount* i *MsgRetryInterval* .

Jeśli pole *MsgRetryExitId* nie jest puste, zostanie uruchomiony program obsługi wyjścia o tej nazwie. Więcej informacji na temat korzystania z własnych programów obsługi wyjścia zawiera sekcja “Programy obsługi wyjścia kanału dla kanałów przesyłania komunikatów” na stronie 406.

Zwróć wiadomość do nadawcy

Użytkownik zwraca komunikat do nadawcy, żądając wygenerowania komunikatu raportu, który ma zawierać wszystkie oryginalne wiadomości.

Szczegółowe informacje na temat opcji komunikatów raportu zawiera sekcja “Komunikaty raportów” na stronie 11 .

Korzystanie z kolejki niedostarczonych komunikatów (niedostarczonych komunikatów)

Gdy menedżer kolejek nie może dostarczyć komunikatu, próbuje on umieścić komunikat w jego kolejce niedostarczonych komunikatów. Ta kolejka powinna być zdefiniowana podczas instalowania menedżera kolejek.

Programy mogą korzystać z kolejki niedostarczonych komunikatów w taki sam sposób, w jaki korzysta z niego menedżer kolejek. Nazwę kolejki niedostarczonych komunikatów można znaleźć, otwierając obiekt menedżera kolejek (przy użyciu wywołania MQOPEN) i zapytaj o atrybut *DeadLetterQName* (przy użyciu wywołania MQINQ).

Gdy menedżer kolejek umieszcza komunikat w tej kolejce, dodaje nagłówek do komunikatu, którego format jest opisany przez strukturę nagłówka niedostarczonych komunikatów (MQDLH). Patrz sekcja MQDLH-Nagłówek niedostarczonych komunikatów . Nagłówek ten zawiera nazwę kolejki docelowej oraz przyczynę umieszczenia komunikatu w kolejce niedostarczonych komunikatów. Musi on zostać usunięty, a problem musi zostać rozwiązany, zanim komunikat zostanie umieszczony w przeznaczonej dla niej

kolejce. Ponadto menedżer kolejek zmienia pole *Format* deskryptora komunikatu (MQMD) w celu wskazania, że komunikat zawiera strukturę MQDLH.

Struktura MQDLH

Zaleca się dodanie struktury MQDLH do wszystkich komunikatów umieszczonych w kolejce niedostarczonych komunikatów. Jeśli jednak planowane jest użycie procedury obsługi niedostarczonych komunikatów udostępnianych przez niektóre produkty WebSphere MQ , użytkownik ***musi*** dodać strukturę MQDLH do komunikatów.

Dodanie nagłówka do komunikatu może spowodować zbyt długi komunikat dla kolejki niedostarczonych komunikatów, dlatego zawsze należy się upewnić, że komunikaty są krótsze od maksymalnej wielkości dozwolonej dla kolejki niedostarczonych komunikatów, przez co najmniej wartość stałej MQ_MSG_HEADER_LENGTH. Maksymalna wielkość komunikatów dozwolonych w kolejce jest określana na podstawie wartości atrybutu *MaxMsgLength* kolejki. W przypadku kolejki niedostarczonych komunikatów należy upewnić się, że ten atrybut jest ustawiony na wartość maksymalną dozwoloną przez menedżer kolejek. Jeśli aplikacja nie może dostarczyć komunikatu, a komunikat jest zbyt długi, aby można go było umieścić w kolejce niedostarczonych komunikatów, należy postępować zgodnie z zaleceniami podanymi w opisie struktury MQDLH.

Upewnij się, że kolejka niedostarczonych komunikatów jest monitorowana i że wszystkie komunikaty docierające do niego są przetwarzane. Procedura obsługi kolejki niedostarczonych komunikatów jest uruchamiana jako program narzędziowy wsadowy i może być używana do wykonywania różnych działań na wybranych komunikatach w kolejce niedostarczonych komunikatów. Więcej informacji na ten temat zawiera sekcja “Przetwarzanie kolejki niedostarczanych komunikatów” na stronie 565.

Jeśli konieczna jest konwersja danych, menedżer kolejek przekształca informacje w nagłówku, gdy używana jest opcja MQGMO_CONVERT w wywołaniu MQGET. Jeśli proces umieszczający komunikat jest agentem MCA, po nagłówku znajduje się cały tekst oryginalnego komunikatu.

Komunikaty umieszczone w kolejce niedostarczonych komunikatów mogą zostać obcięte, jeśli są zbyt długie dla tej kolejki. Możliwe wskazanie tej sytuacji oznacza, że komunikaty w kolejce niedostarczonych komunikatów mają taką samą długość, jak wartość atrybutu *MaxMsgLength* kolejki.

Przetwarzanie kolejki niedostarczanych komunikatów

Informacje te zawierają ogólne informacje na temat interfejsu programistycznego podczas korzystania z przetwarzania w kolejce niedostarczonych komunikatów.

Przetwarzanie kolejki niedostarczanych komunikatów zależy od lokalnych wymagań systemowych, ale podczas rysowania specyfikacji należy rozważyć następujące kwestie:

- Komunikat może zostać zidentyfikowany jako posiadający nagłówek kolejki niedostarczonych komunikatów, ponieważ wartość pola formatu w deskryptyce MQMD to MQFMT_DEAD_LETTER_HEADER.
- W produkcie WebSphere MQ for z/OS przy użyciu programu CICS, jeśli agent MCA umieszcza ten komunikat w kolejce niedostarczonych komunikatów, pole *PutApplType* to MQAT_CICS, a pole *PutApplName* jest *ApplId* systemu CICS , po którym następuje nazwa transakcji agenta MCA.
- Przyczyna, dla której komunikat ma być kierowany do kolejki niedostarczonych komunikatów, znajduje się w polu *Reason* nagłówka kolejki niedostarczonych komunikatów.
- Nagłówek kolejki niedostarczonych komunikatów zawiera szczegółowe informacje na temat nazwy kolejki docelowej i nazwy menedżera kolejek.
- Nagłówek kolejki niedostarczonych komunikatów zawiera pola, które muszą zostać przywrócone do deskryptora komunikatu, zanim komunikat zostanie umieszczony w kolejce docelowej. Są to:
  1. *Encoding*
  2. *CodedCharSetId*
  3. *Format*
- Deskryptor komunikatu jest taki sam jak deskryptor PUT przez oryginalną aplikację, z wyjątkiem trzech wyświetlonych pól (Encoding, CodedCharSetIdi Format).

Aplikacja kolejki niedostarczonych komunikatów musi wykonać jedną lub więcej z następujących czynności:

- Sprawdź pole *Reason* . Agent MCA może umieścić komunikat z następujących powodów:
  - Komunikat był dłuższy niż maksymalna wielkość komunikatu dla kanału

    Przyczyna: MQRC_MSG_TOO_BIG_FOR_CHANNEL
  - Nie można umieścić komunikatu w jego kolejce docelowej

    Przyczyna to dowolny kod przyczyny MQRC_ *, który może zostać zwrócony przez operację MQPUT .
  - Wyjście użytkownika zażądało tego działania

    Kodem przyczyny jest kod dostarczony przez program użytkownika lub domyślna wartość MQRC_SUPPRESSED_BY_EXIT
- Spróbuj przekazać komunikat do zamierzonego miejsca docelowego, gdzie jest to możliwe.
- Należy zachować wiadomość przez pewien czas przed usunięciem, gdy przyczyna dywersji jest określona, ale nie jest ona natychmiast korygowana.
- Podaj instrukcje dla administratorów, którzy rozwiążają problemy, jeśli zostały one określone.
- Odrzuć komunikaty, które są uszkodzone lub w inny sposób nieprzetwarzalne.

Istnieją dwa sposoby radzenia sobie z komunikatami, które zostały odzyskane z kolejki niedostarczonych komunikatów:

1. Jeśli komunikat dotyczy kolejki lokalnej:
   - Przeprowadzanie tłumaczeń kodu wymaganych do wyodrębnienia danych aplikacji
   - Przeprowadzanie konwersji kodu na tych danych, jeśli jest to funkcja lokalna
   - Umieść komunikat wynikowy w kolejce lokalnej ze wszystkimi szczegółami odtwarzanego deskryptora komunikatu
2. Jeśli komunikat jest przeznaczony dla kolejki zdalnej, należy umieścić komunikat w kolejce.

Informacje na temat sposobu obsługi niedostarczanych komunikatów w rozproszonym środowisku kolejkowania można znaleźć w sekcji Co się dzieje, gdy nie można dostarczyć komunikatu?.

Programowanie grupowe

Informacje zawarte w tej sekcji umożliwiają zapoznanie się z zadaniami programowania WebSphere MQ Multicast, takimi jak łączenie się z menedżerem kolejek i raportowanie wyjątków.

Program WebSphere MQ Multicast został zaprojektowany tak, aby był jak najbardziej przejrzysty dla użytkownika, a mimo to jest kompatybilny z istniejącymi aplikacjami. Zdefiniowanie obiektu COMMINFO i ustawienie parametrów **MCAST** i **COMMINFO** obiektu TOPIC oznacza, że istniejące aplikacje produktu WebSphere MQ nie wymagają istotnego przepisania w celu użycia rozsyłania grupowego. Jednak mogą wystąpić pewne ograniczenia (więcej informacji na ten temat zawiera sekcja “Rozsyłanie grupowe i interfejs kolejki komunikatów” na stronie 566), a niektóre zagadnienia związane z bezpieczeństwem (więcej informacji na ten temat można znaleźć w temacie Multicast security).

Rozsyłanie grupowe i interfejs kolejki komunikatów

Te informacje umożliwiają zrozumienie głównych pojęć związanych z interfejsem MQI oraz sposobu ich powiązania z programem WebSphere MQ Multicast.

Subskrypcje rozsyłania grupowego są nietrwałe, ponieważ nie ma żadnych aktywnych kolejek fizycznych, nie ma miejsca do przechowywania wiadomości w trybie bez połączenia, które są tworzone przez subskrypcje trwałe.

Po zasubskrybowaniu przez aplikację tematu rozsyłania grupowego nadawany jest uchwyt obiektu, z którego może korzystać lub MQGET, tak jakby był to uchwyt do kolejki. Oznacza to, że obsługiwane są tylko zarządzane subskrypcje rozsyłania grupowego (subskrypcje utworzone za pomocą komendy MQSO_MANAGED), tj. nie jest możliwe utworzenie subskrypcji i 'wskaż' komunikaty w kolejce. Oznacza

to, że komunikaty muszą być pobierane z uchwytu obiektu zwróconego w wywołaniu subskrypcji. Na kliencie komunikaty są zapisywane w buforze komunikatów do czasu ich konsumowania przez klienta. Więcej informacji na ten temat zawiera sekcja MessageBuffer sekcji pliku konfiguracyjnego klienta . Jeśli klient nie nadąże za pomocą szybkości publikowania, komunikaty są usuwane zgodnie z wymaganiami, a najstarsze komunikaty są usuwane jako pierwsze.

Zwykle jest to decyzja administracyjna, niezależnie od tego, czy aplikacja używa rozsyłania grupowego, czy nie, określonych przez ustawienie atrybutu MCAST dla obiektu TOPIC. Jeśli aplikacja publikowania musi upewnić się, że rozsyłanie grupowe nie jest używane, może użyć opcji `MQOO_NO_MULTICAST` . Podobnie aplikacja subskrybująca może zapewnić, że rozsyłanie grupowe nie jest używane przez zasubskrybowanie opcji `MQSO_NO_MULTICAST` .

WebSphere MQ Multicast supports the use of message selectors. Selektor jest używany przez aplikację w celu zarejestrowania jej zainteresowania tylko tymi komunikatami o właściwościach, które spełniają zapytanie SQL92 , które reprezentuje łańcuch wyboru. Więcej informacji na temat selektorów komunikatów zawiera sekcja “Selektory” na stronie 22.

W poniższej tabeli przedstawiono wszystkie główne pojęcia MQI oraz sposób ich powiązania z Multicast:

Tabela 71. Pojęcia MQI i sposób ich powiązania z rozsyłaniem grupowym

Pojęcie MQI	**Działanie podczas próby przy użyciu rozsyłania grupowego**	**Kod przyczyny**
Umieszczanie komunikatu o zerowej długości	Odrzucone	2005 (07D5) (RC2005): MQRC_BUFFER_LENGTH_ERROR
Grupowanie	Odrzucone	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
Segmentacja	Odrzucone	2443 (098B) (RC2443): MQRC_SEGMENTATION_NOT_ALLOWED
Lista dystrybucyjna	Odrzucone	2154 (086A) (RC2154): MQRC_RECS_PRESENT_ERROR
MQINQ	Odrzucone dla uchwytów tematów: MQINQ i MQSET tematów nie są obsługiwane.	2038 (07F6) (RC2038): MQRC_NOT_OPEN_FOR_INQUIRE
	Akceptowany dla zarządzanego uchwytu. Tylko `Bieżące zapełnienie` może zostać zapytane.	• Jeśli wartością jest Bieżące zapełnienie, nie ma żadnego odpowiedniego kodu przyczyny. • Jeśli wartość jest inna niż Bieżące zapełnienie, kodem przyczyny jest 2067 (0813) (RC2067): MQRC_SELECTOR_ERROR.
MQSET	Odrzucono dla wszystkich uchwytów.	2040 (07F8) (RC2040): MQRC_NOT_OPEN_FOR_SET
Transakcje (XA lub nie)	Odrzucone	2072 (0818) (RC2072): MQRC_SYNCPOINT_NOT_AVAILABLE
Przeglądanie komunikatów	Odrzucone	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Blokowanie komunikatów	Odrzucone	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR

Tabela 71. Pojęcia MQI i sposób ich powiązania z rozsyłaniem grupowym (kontynuacja)

Pojęcie MQI	**Działanie podczas próby przy użyciu rozsyłania grupowego**	**Kod przyczyny**
Przeglądaj z zaznaczonym znakiem	Odrzucone	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Przekaż kontekst	Odrzucone	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPUT1	Odrzucono. Próba wykonania komendy MQPUT1 jest niepoprawna tylko w przypadku tematu rozsyłania grupowego (Multicast).	2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY
subskrypcja trwała	Odrzucono, jeśli temat jest oznaczony jako "Tylko do rozsyłania grupowego", w przeciwnym razie jest dokonywana subskrypcja inna niż Multicast.	2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED
TopicString > 255	Odrzucono. Jeśli łańcuch tematu jest większy niż 255 znaków, zostanie on odrzucony w kliencie.	2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR
Subskrypcja niezarządzana została wykonana	Odrzucono, jeśli temat jest oznaczony jako "Tylko do rozsyłania grupowego", w przeciwnym razie jest dokonywana subskrypcja inna niż Multicast.	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPMO_NOT_OWN_SUBS	Odrzucone	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR

Następujące elementy rozwijają się w niektórych pojęciach MQI z poprzedniej tabeli i zawierają informacje na temat niektórych pojęć MQI, które nie znajdują się w tabeli:

Trwałość komunikatu
W przypadku nietrwałych subskrybentów rozsyłania grupowego komunikaty trwałe pochodzące od publikatora są dostarczane w sposób nienaprawialny.

Obcinanie komunikatów
Obcinanie komunikatów jest obsługiwane, co oznacza, że aplikacja może wykonać następujące czynności:

1. Wprowadź komendę MQGET.
2. Pobieranie komendy MQRC_TRUNCATED_MSG_FAILED.
3. Przydziel większy bufor.
4. Ponownie wydaj komendę MQGET, aby pobrać komunikat.

Utrata ważności subskrypcji
Utrata ważności subskrypcji nie jest obsługiwana. Każda próba ustawienia utraty ważności jest ignorowana.

Wysoka dostępność dla rozsyłania grupowego

Te informacje umożliwiają zrozumienie operacji WebSphere MQ Multicast continuous peer-to-peer; mimo że produkt WebSphere MQ łączy się z menedżerem kolejek produktu WebSphere MQ , komunikaty nie są przepływać przez ten menedżer kolejek.

Mimo że połączenie z menedżerem kolejek musi zostać nawiązane w celu wywołania MQOPEN lub MQSUB obiektu tematu rozsyłania grupowego, same komunikaty nie przepłyną przez menedżer kolejek. Oznacza to, że po zakończeniu operacji MQOPEN lub MQSUB w obiekcie tematu rozsyłania grupowego można kontynuować przesyłanie komunikatów rozsyłania grupowego, nawet jeśli połączenie z menedżerem kolejek zostało utracone. Istnieją dwa tryby pracy:

Do menedżera kolejek wykonywane jest normalne połączenie.
Komunikacja między rozsyłaniem grupowym jest możliwa, gdy istnieje połączenie z menedżerem kolejek. Jeśli nawiązanie połączenia nie powiedzie się, zostaną zastosowane normalne reguły MQI, na przykład: operacja MQPUT dla uchwytu obiektu rozsyłania grupowego zwraca wartość 2009 (07D9) (RC2009): MQRC_CONNECTION_BROKEN.

Nawiąże połączenie z menedżerem kolejek podczas ponownego nawiązywania połączenia klienta.
Komunikacja rozsyłania grupowego jest możliwa nawet podczas cyklu ponownego połączenia. Oznacza to, że nawet jeśli połączenie z menedżerem kolejek zostało zerwane, nie ma to wpływu na umieszczanie i korzystanie z komunikatów rozsyłania grupowego. Klient podejmuje próbę ponownego nawiązania połączenia z menedżerem kolejek. Jeśli ponowne nawiązanie połączenia nie powiedzie się, uchwyt połączenia zostanie zerwany, a wszystkie wywołania MQI, w tym rozgłaszanie, nie powiodą się. Więcej informacji na ten temat zawiera sekcja Automated client reconnection (Automatyczne ponowne połączenie klienta)

Jeśli dowolna aplikacja jawnie zgłasza komendę MQDISC, wszystkie subskrypcje rozsyłania grupowego i uchwyty obiektów są zamykane.

Ciągłe grupowe operacje typu peer-to-peer

Jedną z zalet komunikacji równorzędnej między klientami jest to, że komunikaty nie muszą przepływać przez menedżer kolejek. W związku z tym, jeśli połączenie z menedżerem kolejek zostanie zerowane, przesyłanie komunikatów będzie kontynuowane. W przypadku wymagań dotyczących komunikatów ciągłych w tym trybie obowiązują następujące ograniczenia:

- Połączenie musi być nawiązane za pomocą jednej z opcji MQCNO_RECONNECT_ * w celu ciągłej pracy. Ten proces oznacza, że chociaż sesja komunikacyjna może być uszkodzona, rzeczywisty uchwyt połączenia nie jest zerwany i znajduje się w stanie ponownego połączenia. Jeśli ponowne nawiązanie połączenia nie powiedzie się, uchwyt połączenia zostanie zerwany, co uniemożliwia wykonanie wszystkich kolejnych wywołań MQI.

- W tym trybie obsługiwane są tylko zmaterializowane tabele MQPUT, MQGET, MQINQ i Async. Wszystkie czasowniki MQOPEN, MQCLOSE lub MQDISC wymagają ponownego nawiązania połączenia z menedżerem kolejek w celu zakończenia.
- Status przepływa do menedżera kolejek; w związku z tym każdy stan w menedżerze kolejek może być nieaktualny lub nie. Oznacza to, że klienty mogą wysyłać i odbierać komunikaty, a w menedżerze kolejek nie ma statusu znanego. Więcej informacji na ten temat zawiera sekcja Monitorowanie aplikacji Multicast

Konwersja danych w przesyłaniu komunikatów MQI na potrzeby rozsyłania grupowego

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.

Dane są przekształcane na kliencie dla ruchu multicast produktu WebSphere MQ . Jeśli zostanie podana opcja **MQGMO_CONVERT** , konwersja danych jest wykonywana zgodnie z żądaniem. Formaty zdefiniowane przez użytkownika wymagają wyjścia konwersji danych zainstalowanego na kliencie. Informacje o bibliotekach znajdują się teraz w pakietach klienta i serwera. Informacje o bibliotekach znajdują się teraz w “Pisanie wyjść konwersji danych” na stronie 426 .

Więcej informacji na temat administrowania konwersją danych zawiera sekcja Włączanie konwersji danych na potrzeby przesyłania komunikatów w trybie 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.

Raportowanie wyjątków rozsyłania grupowego

Use this information to learn about WebSphere MQ Multicast event handlers and reporting WebSphere MQ Multicast exceptions.

Produkt WebSphere MQ Multicast jest pomocny przy określaniu problemu przez wywołanie procedury obsługi zdarzeń w celu zgłaszania zdarzeń rozsyłania grupowego, które są zgłaszane za pomocą standardowego mechanizmu procedury obsługi zdarzeń WebSphere MQ .

Pojedyncze zdarzenie rozsyłania grupowego może spowodować wywołanie więcej niż jednego zdarzenia WebSphere MQ , ponieważ może istnieć wiele uchwytów połączeń produktu `MQHCONN` korzystających z tego samego nadajnika rozsyłania grupowego lub odbiornika. Jednak każdy wyjątek rozsyłania grupowego powoduje wywołanie tylko jednej procedury obsługi zdarzeń dla połączenia produktu WebSphere MQ .

Stała WebSphere MQ `MQCBDO_EVENT_CALL` umożliwia aplikacjom zarejestrowanie wywołania zwrotnego w celu odbierania tylko zdarzeń produktu WebSphere MQ , a program `MQCBDO_MC_EVENT_CALL` umożliwia aplikacjom rejestrowanie wywołania zwrotnego w celu odbierania tylko zdarzeń rozsyłania grupowego. Jeśli używane są obie stałe, odbierane są oba typy zdarzeń.

Żądanie zdarzeń rozsyłania grupowego

WebSphere MQ Multicast events use the `MQCBDO_MC_EVENT_CALL` constant in the `cbd.Options` field. Poniższy przykład demonstruje sposób żądania zdarzeń rozsyłania grupowego:

```
cbd.CallbackType     = MQCBT_EVENT_HANDLER;
cbd.Options          = MQCBDO_MC_EVENT_CALL;
cbd.CallbackFunction = EventHandler;
MQCB(Hcon,MQOP_REGISTER,&cbd,MQHO_UNUSABLE_HOBJ,NULL,NULL,&CompCode,&Reason);
```

When the MQCBDO_MC_EVENT_CALL option is specified for the cbd.Options field, the event handler is sent only WebSphere MQ Multicast events instead of connection level events. Aby zażądać, aby oba typy zdarzeń zostały wysłane do procedury obsługi zdarzeń, aplikacja musi określić stałą MQCBDO_EVENT_CALL w polu cbd.Options oraz stałą MQCBDO_MC_EVENT_CALL , jak pokazano w poniższym przykładzie:

```
cbd.CallbackType     = MQCBT_EVENT_HANDLER;
cbd.Options          = MQCBDO_EVENT_CALL | MQCBDO_MC_EVENT_CALL
cbd.CallbackFunction = EventHandler;
MQCB(Hcon,MQOP_REGISTER,&cbd,MQHO_UNUSABLE_HOBJ,NULL,NULL,&CompCode,&Reason);
```

Jeśli żadna z tych stałych nie jest używana, do procedury obsługi zdarzeń wysyłane są tylko zdarzenia na poziomie połączenia.

Więcej informacji na temat wartości w polu Options zawiera sekcja Opcje (MQLONG).

Format zdarzenia rozsyłania grupowego

WebSphere MQ Wyjątki rozsyłania grupowego obejmują niektóre informacje dodatkowe zwracane w parametrze **Buffer** funkcji zwrotnej. Wskaźnik **Buffer** wskazuje na tablicę wskaźników, a pole MQCBC.DataLength określa wielkość tablicy w bajtach. Pierwszy element tablicy zawsze wskazuje na krótki tekst opisu zdarzenia. W zależności od typu zdarzenia może być podana większa liczba parametrów. W poniższej tabeli przedstawiono wyjątki:

Tabela 72. Opisy kodów zdarzeń rozsyłania grupowego

Kod zdarzenia	**Opis**	**Dane dodatkowe**
MQMCEV_PACKET_LOSS	Nienaprawialna utrata pakietów	Liczba utraconych pakietów
MQMCEV_HEARTBEAT_TIMEOUT	Długa nieobecność pakietu kontrolnego pulsu	Nie dotyczy
MQMCEV_VERSION_CONFLICT	Odbiór nowszych pakietów wersji protokołu	Nie dotyczy
MQMCEV_RELIABILITY	Różne tryby niezawodności nadajnika i odbiornika	Nie dotyczy
MQMCEV_CLOSED_TRANS	Transmisja tematu jest zamknięta przez 1 źródło	Nie dotyczy
Błąd MQMCEV_STREAM_ERROR	Wykryto błąd w strumieniu	Nie dotyczy
MQMCEV_NEW_SOURCE	Nowe źródło rozpoczyna transmisję w temacie	Struktura źródłowa
MQMCEV_RECEIVE_QUEUE_TRIMMED	Pakiety usunięte z pakietu PacketQ z powodu utraty ważności czasu lub miejsca	Liczba pakietów trimmed
MQMCEV_PACKET_LOSS_NACK_EXPIRE	Nienaprawialna utrata pakietów z powodu utraty ważności NACK	Liczba utraconych pakietów
MQMCEV_ACK_RETRIES_EXCEEDED	Liczba pakietów usuniętych z historii po **max_ack_retries** została przekroczona	Liczba usuniętych pakietów
MQMCEV_STREAM_SUSPEND_NACK	NACKs zostało zawieszone w strumieniu zaakceptowany przez ten temat	Identyfikator strumienia zawieszenia
		Czas (w milisekundach), przez który strumień jest zawieszony

Tabela 72. Opisy kodów zdarzeń rozsyłania grupowego (kontynuacja)

Kod zdarzenia	Opis	Dane dodatkowe
MQMCEV_STREAM_RESUME_NACK	NACKs zostały wznowione po zawieszeniu ich w strumieniu	Identyfikator strumienia
MQMCEV_STREAM_EXPELLED	Strumień zaakceptowany przez ten wątek został odrzucony ze względu na żądanie expel	Identyfikator strumienia
Komunikat MQMCEV_FIRST_MESSAGE	Pierwsza wiadomość od źródła	Numer komunikatu
Błąd MQMCEV_LATE_JOIN_FAILURE	Nie powiodło się uruchomienie opóźnionej sesji łączenia	Nie dotyczy
MQMCEV_MESSAGE_LOSS	Nienaprawialna utrata komunikatów	Liczba utraconych komunikatów
Błąd MQMCEV_SEND_PACKET_FAILURE	Nie powiodło się wysłanie przez nadajnik rozsyłania grupowego pakietu rozsyłania grupowego	Nie dotyczy
MQMCEV_REPAIR_DELAY	Odbiornik rozsyłania grupowego nie otrzymał pakietu naprawczego dla zaległego NAK	Nie dotyczy
MQMCEV_MEMORY_ALERT_ON	Bufory odbiorcze odbiorcze zapełnia	Procent wykorzystania puli buforów
MQMCEV_MEMORY_ALERT_OFF	Bufory odbiorcze odbiorcze są wyłączone	Procent wykorzystania puli buforów
MQMCEV_NACK_ALERT_ON	Współczynnik żądań pakietu naprawczego odbiornika osiągnął wysoki wskaźnik poziomu	Bieżąca szybkość żądań napraw w pakietach na sekundę
MQMCEV_NACK_ALERT_OFF	Częstotliwość żądań pakietów napraw odbiornika jest wyłączona	Bieżąca szybkość żądań napraw w pakietach na sekundę
MQMCEV_REPAIR_ALERT_ON	Szybkość wysyłania pakietów przez nadajnik osiągnęła wysoki wskaźnik poziomu wody	Nie dotyczy
MQMCEV_REPAIR_ALERT_OFF	Szybkość wysyłania pakietów do naprawy nadajnika jest wyłączona do normy	Nie dotyczy
MQMCEV_SHM_DEST_UNUSABLE	Wykryto, że obszar pamięci współużytkowanej używany przez miejsce docelowe tematu nadajnika nie może być używany.	Nie dotyczy
MQMCEV_SHM_PORT_UNUSABLE	Wykryto, że port pamięci współużytkowanej używany przez instancję odbiornika nie może być używany	Nie dotyczy
Funkcja MQMCEV_CCT_GETTIME_FAILED	Nie powiodła się próba pobrania czasu z koordynowanego klastra	Nie dotyczy

Tabela 72. Opisy kodów zdarzeń rozsyłania grupowego (kontynuacja)		
Kod zdarzenia	**Opis**	**Dane dodatkowe**
MQMCEV_DEST_INTERFACE_FAILURE	Interfejs sieciowy używany przez miejsce docelowe tematu nadajnika nie powiódł się, a zapasowy interfejs sieciowy jest niedostępny	
MQMCEV_DEST_INTERFACE_FAILOVER	Interfejs sieciowy używany przez miejsce docelowe tematu nadajnika nie powiódł się, a przełączenie awaryjne na inny interfejs zostało zakończone	
MQMCEV_PORT_INTERFACE-NIEPOWODZENIE	Interfejs sieciowy używany przez odbiornik rmmPort nie powiódł się, a zapasowy interfejs sieciowy jest niedostępny (lub też nie powiódł się)	Konfiguracja RMM
MQMCEV_PORT_INTERFACE_FAILOVER	Interfejs sieciowy używany przez odbiornik rmmPort uległ awarii, a przełączenie awaryjne na inny interfejs zostało zakończone.	Konfiguracja RMM

Używanie środowiska .NET

Klasy produktu WebSphere MQ dla środowiska .NET umożliwiają programowi napisanego w środowisku programowania .NET nawiązanie połączenia z produktem WebSphere MQ jako klienta MQI produktu WebSphere MQ lub nawiązanie bezpośredniego połączenia z serwerem WebSphere MQ .

Jeśli istnieją aplikacje, które używają środowiska .NET firmy Microsoft i chcą skorzystać z infrastruktury produktu WebSphere MQ, należy użyć klas produktu WebSphere MQ dla środowiska .NET.

Zorientowany obiektowo interfejs WebSphere MQ .NET różni się od interfejsu MQI w tym, że używa metod obiektów, a nie za pomocą komend MQI.

Proceduralny interfejs programistyczny aplikacji WebSphere MQ jest zbudowany wokół komend, takich jak na poniższej liście: .

```
MQCONN, MQDISC, MQOPEN, MQCLOSE,
MQINQ, MQSET, MQGET, MQPUT, MQSUB
```

Te czasowniki przyjmują, jako parametr, uchwyt obiektu WebSphere MQ , na którym mają działać.
Ze względu na to, że środowisko .NET jest obiektowe, interfejs programistyczny .NET włącza tę runę.
Program składa się z zestawu obiektów WebSphere MQ , które są używane przez wywołania metod na tych obiektach. Programy można pisać w dowolnym języku obsługiwanym przez .NET.

Jeśli używany jest interfejs proceduralny, należy odłączyć się od menedżera kolejek za pomocą wywołania MQDISC (*Hconn*, CompCode, Reason), gdzie *Hconn* jest uchwytem menedżera kolejek.

W interfejsie .NET menedżer kolejek jest reprezentowany przez obiekt klasy MQQueueManager. Odłącz się od menedżera kolejek, wywołując metodę Disconnect () dla tej klasy.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.Disconnect();
```

Produkt WebSphere MQ classes for .NET to zestaw klas, które umożliwiają aplikacjom .NET wchodzić w interakcje z produktem WebSphere MQ. Reprezentują one różne komponenty produktu WebSphere MQ , z których korzysta aplikacja, takie jak menedżery kolejek, kolejki, kanały i komunikaty. Szczegółowe informacje na temat tych klas zawiera sekcja Klasy i interfejsy środowiska .NET produktu WebSphere MQ.

Zanim możliwe będzie skompilowanie wszystkich napisanych aplikacji, konieczne jest zainstalowanie środowiska .NET Framework. Instrukcje dotyczące instalowania klas produktu WebSphere MQ dla środowiska .NET i środowiska .NET Framework zawiera sekcja “Instalowanie klas produktu WebSphere MQ dla środowiska .NET” na stronie 575.

Pojęcia pokrewne

Przegląd techniczny

“Opcje nawiązywania połączeń” na stronie 574
Istnieją trzy tryby łączenia klas produktu WebSphere MQ dla środowiska .NET z menedżerem kolejek. Należy wziąć pod uwagę, który typ połączenia najlepiej odpowiada wymaganiom użytkownika.

“Korzystanie z klas produktu WebSphere MQ dla środowiska .NET” na stronie 586
W tej kolekcji tematów opisano sposób konfigurowania systemu w celu uruchamiania przykładowych programów w celu sprawdzenia klas WebSphere MQ dla instalacji .NET oraz sposobu uruchamiania własnych programów.

“Rozwiązywanie problemów z produktem WebSphere MQ .NET” na stronie 588
Jeśli program nie zakończy się pomyślnie, uruchom jedną z przykładowych aplikacji i postępuj zgodnie z zaleceniami podanymi w komunikatach diagnostycznych.

“Pisanie i wdrażanie programów WebSphere MQ .NET” na stronie 589
Aby używać klas produktu WebSphere MQ dla środowiska .NET w celu uzyskiwania dostępu do kolejek produktu WebSphere MQ , należy pisać programy w dowolnym języku obsługiwanym przez środowisko .NET, zawierającym wywołania wstawiające komunikaty do kolejek produktu WebSphere MQ i pobierające je z nich.

“Kanał niestandardowy produktu IBM WebSphere MQ dla produktu Microsoft Windows Communication Foundation (WCF)” na stronie 609
Kanał niestandardowy produktu Microsoft Windows Communication Foundation (WCF) dla produktu IBM WebSphere MQ wysyła i odbiera komunikaty między klientami i usługami WCF.

“Wybór języka programowania do użycia” na stronie 80
W tej sekcji znajdują się informacje na temat języków programowania i środowisk obsługiwanych przez produkt IBM WebSphere MQ, a także niektóre zagadnienia dotyczące ich używania.

“Projektowanie aplikacji” na stronie 7
Produkt IBM WebSphere MQ udostępnia kilka sposobów tworzenia aplikacji w celu wysyłania i odbierania komunikatów, które są potrzebne do obsługi procesów biznesowych. Istnieje również możliwość tworzenia aplikacji do zarządzania menedżerami kolejek i powiązanymi zasobami.

Pierwsze kroki z klasami produktu WebSphere MQ dla środowiska .NET

Klasy produktu WebSphere MQ dla środowiska .NET umożliwiają programowi napisanego w środowisku programowania .NET nawiązanie połączenia z produktem WebSphere MQ jako klienta MQI produktu WebSphere MQ lub nawiązanie bezpośredniego połączenia z serwerem WebSphere MQ .

Opcje nawiązywania połączeń

Istnieją trzy tryby łączenia klas produktu WebSphere MQ dla środowiska .NET z menedżerem kolejek. Należy wziąć pod uwagę, który typ połączenia najlepiej odpowiada wymaganiom użytkownika.

Połączenie powiązań klienta

Aby można było używać klas WebSphere MQ dla środowiska .NET jako klienta MQI produktu WebSphere MQ , można go zainstalować przy użyciu klienta MQI produktu WebSphere MQ na komputerze serwera WebSphere MQ lub na osobnym komputerze. Połączenie z powiązaniami klienta może używać transakcji XA lub innych niż XA.

Połączenie powiązań serwera

W przypadku użycia w trybie powiązań serwera klasy produktu WebSphere MQ dla środowiska .NET korzystają z interfejsu API menedżera kolejek, a nie komunikują się za pośrednictwem sieci. Zapewnia to lepszą wydajność aplikacji WebSphere MQ niż korzystanie z połączeń sieciowych.

Aby korzystać z połączenia powiązań, należy zainstalować klasy produktu WebSphere MQ dla środowiska .NET na serwerze WebSphere MQ .

Połączenie z klientem zarządzanym

Połączenie wykonane w tym trybie łączy się jako klient WebSphere MQ z serwerem WebSphere MQ uruchomionym na komputerze lokalnym lub zdalnym.

Klasy produktu WebSphere MQ dla środowiska .NET łączące się w tym trybie pozostają w kodzie zarządzanym .NET i nie wywołują żadnych wywołań rodzimych usług. Więcej informacji na temat kodu zarządzanego można znaleźć w dokumentacji firmy Microsoft .

Istnieje wiele ograniczeń dotyczących korzystania z zarządzanego klienta. Aby uzyskać więcej informacji na ten temat, zobacz: “Połączenia zarządzane przez klienta” na stronie 589.

Instalowanie klas produktu WebSphere MQ dla środowiska .NET

Klasy produktu WebSphere MQ dla środowiska .NET, w tym przykłady, są instalowane razem z produktem WebSphere MQ. Istnieje wymaganie wstępne dla środowiska Microsoft .NET Framework.

Najnowsza wersja klas WebSphere MQ dla środowiska .NET jest instalowana domyślnie jako część standardowej instalacji produktu WebSphere MQ w funkcji *Java and .NET Messaging and Web Services* (Przesyłanie komunikatów języka Java i .NET). Instrukcje dotyczące instalowania znajdują się w sekcji Instalowanie serwera IBM WebSphere MQ w systemie Windows lub Instalowanie klienta IBM WebSphere MQ w systemach Windows .

W środowisku z wieloma instalacyjnymi, jeśli wcześniej zainstalowano klasy produktu WebSphere MQ dla środowiska .NET jako pakiet obsługi, nie można zainstalować produktu WebSphere MQ , chyba że pakiet poprawek zostanie zdeinstalowany. Składnik WebSphere MQ classes for .NET, który jest instalowany razem z produktem WebSphere MQ , zawiera tę samą funkcjonalność, co pakiet wsparcia.

Dostarczane są również przykładowe aplikacje, w tym pliki źródłowe; patrz “Aplikacje przykładowe” na stronie 586.

Aby uruchomić klasy produktu WebSphere MQ dla platformy .NET na platformach 32-bitowych lub 64-bitowych, należy zainstalować środowisko Microsoft .NET Framework w wersji V2.0 lub nowszej.

Uwaga: Jeśli środowisko Microsoft .NET Framework w wersji v2.0 lub nowszej nie jest zainstalowane przed zainstalowaniem produktu WebSphere MQ V7.0.1, instalacja produktu WebSphere MQ będzie kontynuowana bez błędów, ale klasy produktu WebSphere MQ dla środowiska .NET nie będą dostępne. Jeśli środowisko .NET Framework jest zainstalowane po zainstalowaniu produktu WebSphere MQ 7.0.1, zespoły WebSphere .NET muszą być zarejestrowane, uruchamiając skrypt *WMQInstallDir*`\bin\amqiRegisterdotNet.cmd` , gdzie *WMQInstallDir* to katalog, w którym zainstalowano produkt WebSphere MQ 7.0.1 . Ten skrypt zainstaluje wymagane zespoły w globalnej pamięci podręcznej zespołu (Global Assembly Cache-GAC). W katalogu %TEMP% tworzony jest zestaw plików produktu `amqi*.log` , które rejestrują działania.

Informacje na temat korzystania z niestandardowego kanału WebSphere MQ dla systemu Microsoft WCF z platformą .NET 3 można znaleźć pod adresem: “Kanał niestandardowy produktu IBM WebSphere MQ dla produktu Microsoft Windows Communication Foundation (WCF)” na stronie 609

Rozproszone transakcje w środowisku .NET

Transakcje rozproszone lub transakcje globalne umożliwiają aplikacjom klienckim kilka różnych źródeł danych na dwóch lub więcej systemach sieciowych w jednej transakcji.

W transakcjach rozproszonych menedżer transakcji koordynuje transakcję i zarządza nią wśród dwóch lub większej liczby menedżerów zasobów.

Transakcje mogą być procesem zatwierdzania jednofazowego lub dwufazowego. Zatwierdzanie jednofazowe to proces, w którym tylko jeden menedżer zasobów uczestniczy w procesie zatwierdzania transakcji, a proces zatwierdzania dwufazowego jest w przypadku, gdy uczestniczy w transakcji więcej niż jeden menedżer zasobów. W procesie zatwierdzania dwufazowego menedżer transakcji wysyła wywołanie przygotowania w celu sprawdzenia, czy wszystkie menedżery zasobów są przygotowane do zatwierdzenia. Po odebraniu potwierdzenia ze wszystkich menedżerów zasobów wywołanie zatwierdzenia jest wykonywane. W przeciwnym razie dochodzi do wycofania zmian w całej transakcji. Więcej informacji na ten temat zawiera sekcja Zarządzanie transakcjami i obsługa . Kierownicy zasobów powinni informować kierowników transakcji o swoim udziale w transakcji. Gdy menedżer zasobów poinformuje menedżera transakcji o swoim udziale, menedżer zasobów pobiera oddzwonienia od menedżera transakcji, gdy transakcja ma zostać zatwierdzona lub wycofana.

WebSphere MQ .NET classes already supports distributed transactions in unmanaged and server bindings mode connections. W tych trybach klasy WebSphere MQ .NET delegują wszystkie jego wywołania do rozszerzonego klienta transakcji w języku C, który zarządza przetwarzaniem transakcji w imieniu środowiska .NET.

WebSphere MQ.NET obsługują teraz rozproszone transakcje w trybie zarządzanym, gdzie WebSphere MQ .NET Klasy korzysta z przestrzeni nazw System.Transactions dla obsługi transakcji rozproszonych. Infrastruktura System.Transactions sprawia, że programowanie transakcyjne jest proste i wydajne, dzięki obsłudze transakcji zainicjowanych we wszystkich menedżerach zasobów, w tym WebSphere MQ. Aplikacja WebSphere MQ .NET umożliwia wstawianie i pobieranie komunikatów przy użyciu niejawnego programowania transakcji .NET lub jawnego modelu programowania transakcji. W transakcjach niejawnych granice transakcji są tworzone przez program użytkowy, który podejmuje decyzję o zatwierdzeniu, wycofaniu (w przypadku transakcji jawnych) lub zakończeniu transakcji. W transakcjach jawnych konieczne jest jawne określenie, czy transakcja ma zostać zatwierdzona, wycofana, czy zakończona.

WebSphere MQ.NET korzysta z koordynatora transakcji rozproszonych Microsoft (MS DTC) jako menedżera transakcji, który koordynuje i zarządza transakcją między wieloma menedżerami zasobów. Produkt WebSphere MQ jest używany jako menedżer zasobów.

WebSphere MQ.NET jest zgodny z modelem X/Open Distributed Transaction Processing (DTP). Model X/Open Distributed Transaction Processing jest rozproszonym modelem przetwarzania transakcji zaproponowanym przez Open Group, konsorcjum dostawców. Model ten jest standardem wśród większości komercyjnych dostawców w zakresie przetwarzania transakcyjnego i domen baz danych. Większość produktów do zarządzania transakcjami komercyjnymi obsługuje model X/DTP.

Tryby transakcji

- “Transakcje rozproszone w trybie zarządzanym” na stronie 577
- Rozproszone transakcje dla trybu niezarządzanego

Koordynowanie transakcji w różnych scenariuszach

- Połączenie może uczestniczyć w kilku transakcjach, ale tylko jedna transakcja jest aktywna w dowolnym momencie.
- Podczas transakcji MQQueueManager.Wywołanie odłączenia zostało uhonorowane. W takim przypadku transakcja jest proszona o wycofanie zmian.
- Podczas transakcji honorowane jest wywołanie MQQueue.Close lub MQTopic.Close . W tym przypadku transakcja jest proszona o wycofanie zmian.
- Granice transakcji są tworzone przez program użytkowy, który decyduje, kiedy zatwierdzić, wycofać (w przypadku transakcji jawnych) lub zakończyć (dla transakcji niejawnych) transakcję.

- Jeśli aplikacja kliencka przerwie działanie podczas transakcji z nieoczekiwanym błędem przed wywołaniem funkcji Put lub Get w wywołaniu kolejki lub tematu, transakcja jest wycofana i zgłaszany jest wyjątek MQException.
- Jeśli kod przyczyny MQCC_FAILED jest zwracany podczas wywoływania operacji umieszczania lub pobierania w kolejce lub w wywołaniu tematu, zgłaszany jest wyjątek MQException z kodem przyczyny, a transakcja jest wycofana. Jeśli menedżer transakcji wygenerował już wywołanie przygotowania, produkt WebSphere MQ .NET zwraca żądanie przygotowania, wymuszając wycofanie transakcji. Następnie menedżer transakcji DTC powoduje wycofanie zmian w bieżącej pracy ze wszystkimi menedżerami zasobów w bieżących transakcjach otoczenia.
- W przypadku transakcji obejmującej wiele menedżerów zasobów, jeśli jakiś powód środowiskowy powoduje, że wywołanie funkcji Put lub Get jest nieokreślone, menedżer transakcji oczekuje na określony czas. Po zakończeniu tego czasu wycofywanie wszystkich bieżących prac ze wszystkimi menedżerami zasobów w bieżących transakcjach otoczenia jest przyczyną wycofania wszystkich bieżących prac. Jeśli ten nieokreślony czas oczekiwania zostanie przekroczony podczas fazy przygotowania, menedżer transakcji może przekroczenie limitu czasu lub wywołać wątpliwe wywołanie zasobu, w którym to przypadku transakcja jest wycofana.
- Aplikacje korzystające z transakcji muszą umieścić lub pobrać komunikaty w katalogu SYNC_POINT. Jeśli komunikat Put lub Get jest generowany w kontekście transakcyjnym, który nie znajduje się w obszarze SYNC_POINT, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_UNIT_OF_WORK_NOT_STARTED.

Różnice behawioralne między obsługą transakcji klienta zarządzanego i niezarządzanego za pomocą przestrzeni nazw Microsoft .NET System.Transactions

Transakcje zagnieżdżone mają TransactionScope w innym elemencie TransactionScope

- W pełni zarządzany klient WebSphere MQ .NET obsługuje zagnieżdżony TransactionScope
- Niezarządzany klient WebSphere MQ .NET nie obsługuje zagnieżdżonego elementu TransactionScope

Transakcje zależne od System.Transactions

- W pełni zarządzany klient WebSphere MQ .NET obsługuje funkcję transakcji zależnych udostępnianą przez System.Transactions.
- Niezarządzany klient WebSphere MQ .NET nie obsługuje mechanizmu transakcji zależnych udostępnianego przez System.Transactions.

Przykłady produktów

Nowe przykłady produktów `SimpleXAPut`i `SimpleXAGet` są dostępne pod `WebSphere MQ\tools\dotnet\samples\cs\base` . Przykłady to aplikacje typu C#, które demonstrują za pomocą operacji MQPUT i MQGET w ramach transakcji rozproszonych przy użyciu przestrzeni nazw SystemTransactions . Więcej informacji na temat tych przykładów zawiera sekcja “Tworzenie prostych komunikatów put i get w ramach TransactionScope” na stronie 580 .

Transakcje rozproszone w trybie zarządzanym

WebSphere MQ .NET classes use System.Transactions namespace for the distributed transactions support in managed mode. W trybie zarządzanym MS DTC koordynuje i zarządza rozproszonymi transakcjami na wszystkich serwerach wymienionych w transakcji.

Klasy .NET produktu WebSphere MQ udostępniają jawny model programistyczny oparty na klasie System.Transactions.Transaction i niejawny model programistyczny przy użyciu klasy System.Transactions.TransactionScope, w której transakcje są automatycznie zarządzane przez infrastrukturę.

Transakcja niejawna

Poniższy fragment kodu opisuje sposób, w jaki aplikacja WebSphere MQ .NET umieszcza komunikat przy użyciu niejawnego programowania transakcji .NET.

```
Using (TransactionScope scope = new TransactionScope ())
{
    Q.Put (putMsg,pmo);
    scope.Complete ();
}

Q.close();
qMgr.Disconect();}
```

Wyjaśnienie przepływu kodu dla transakcji niejawnej

Kod tworzy obiekt *TransactionScope* i umieszcza komunikat w zasięgu. Następnie wywołuje *Zakończone* , aby poinformować koordynatora transakcji o zakończeniu transakcji. Koordynator transakcji wydaje teraz *przygotowanie* i *zatwierdzanie* w celu zakończenia transakcji. Jeśli problem zostanie wykryty, wywoływana jest *wycofywanie zmian* .

Transakcja jawna

Poniższy kod opisuje sposób, w jaki aplikacja WebSphere MQ .NET umieszcza komunikaty przy użyciu jawnego modelu programowania transakcji .NET.

```
MQQueueManager qMgr = new MQQueuemanager ("MQQM);
MQQueue Q = QMGR.AccessQueue("Q", MQC.MQOO_OUTPUT+MQC.MQOO_INPUT_SHARED);
MQPutMessageOptions pmo = new MQPutMessageOptions();
pmo.Options = MQC.MQPMO_SYNCPOINT;
MQMessage putMsg1 = new MQMessage();
Using(CommittableTransaction tx = new CommittableTransaction()){
Transaction.Current = tx;
    try
    {
    Q.Put(MSG,pmo);
    tx.commit();
    }
    catch(Exception)
    {tx.rollback();}
    }

Q.close();
qMgr.Disconnect();
}
```

Wyjaśnienie przepływu kodu dla transakcji jawnej

Fragment kodu tworzy transakcję przy użyciu klasy *CommitableTransaction* . Umieszcza komunikat w tym zasięgu, a następnie jawnie wywołuje komendę *commit* , aby zakończyć transakcję. Jeśli wystąpią jakieś problemy, *wycofywanie zmian* jest wywoływane.

Transakcje rozproszone w trybie niezarządzanym

WebSphere MQ.NET obsługują połączenia niezarządzane (klient) przy użyciu rozszerzonego klienta transakcji, a COM + /MTS jako koordynator transakcji, przy użyciu niejawnego lub jawnego modelu programowania transakcji. W trybie niezarządzanym klasy .NET produktu WebSphere MQ delegują wszystkie wywołania do rozszerzonego klienta transakcji w języku C, który zarządza przetwarzaniem transakcji w imieniu środowiska .NET.

Przetwarzanie transakcji jest kontrolowane przez zewnętrznego menedżera transakcji, koordynując globalną jednostkę pracy pod kontrolą interfejsu API menedżera transakcji. Czasowniki MQBEGIN, MQCMIT i MQBACK są niedostępne. WebSphere MQ .NET classes expose this support by way of its unmanaged transport mode (C client). Patrz sekcja Konfigurowanie menedżerów transakcji zgodnych z interfejsem XA .

System MTS jest oparty na systemie przetwarzania transakcyjnego (TP) w celu udostępnienia tych samych funkcji w systemie Windows NT dostępnym w systemach CICS, Tuxedo i na innych platformach. Po zainstalowaniu MTS do systemu Windows NT dodawana jest osobna usługa, która jest nazywana koordynatorem Microsoft Distributed Transaction Coordinator (MSDTC). MSDTC koordynuje transakcje, które obejmują oddzielne składnice danych lub zasoby. Do pracy wymagane jest, aby każda składnica danych zaimplementował własny, własny menedżer zasobów.

Produkt WebSphere MQ staje się kompatybilny z MSDTC przez zaimplementowanie interfejsu (interfejsu menedżera zasobów zastrzeżonych), w którym zarządza on odwzorowywaniem wywołań DTC XA na wywołania WebSphere MQ(X/Open). Produkt WebSphere MQ pełni rolę menedżera zasobów.

Gdy komponent, taki jak COM + żąda dostępu do produktu WebSphere MQ, COM zwykle sprawdza odpowiedni obiekt kontekstu MTS, jeśli jest wymagana transakcja. Jeśli transakcja jest wymagana, COM informuje o tym DTC i automatycznie uruchamia transakcję WebSphere MQ dla tej operacji. Następnie COM współpracuje z danymi za pomocą oprogramowania MQMTS, umieszczając i pobierając komunikaty zgodnie z wymaganiami. Instancja obiektu uzyskana z modelu COM wywołuje metodę SetComplete lub SetAbort po zakończeniu wszystkich działań na danych. Gdy aplikacja wydaje komendę SetComplete, wywołanie sygnalizuje DTC, że aplikacja zakończyła transakcję, a DTC może wyprzeć proces zatwierdzania dwufazowego. Następnie DTC wywołuje wywołania MQMTS, które z kolei wywołują WebSphere MQ w celu zatwierdzenia lub wycofania transakcji.

Pisanie aplikacji WebSphere MQ .NET przy użyciu niezarządzanego klienta

Aby można było uruchomić w kontekście COM +, klasa .NET musi dziedziczyć z systemu System.EnterpriseServices.ServicedComponent. Reguły i zalecenia dotyczące tworzenia zespołów, które korzystają z obsługiwanych komponentów, są następujące:

Uwaga: Poniższe kroki są istotne tylko wtedy, gdy używany jest tryb System.EnterpriseServices .

- Klasa i metoda uruchamiana w COM + muszą być publiczne (nie ma klas wewnętrznych i nie są chronione ani statyczne).
- Atrybuty klasy i metody: atrybut TransactionOption określa poziom transakcji klasy, to znaczy, czy transakcje są wyłączone, obsługiwane lub wymagane. Atrybut AutoComplete w metodzie ExecuteUOW() instruuje COM +, aby zatwierdzić transakcję, jeśli nie został zgłoszony żaden nieobsługiwany wyjątek.
- Silne-nazywanie zespołu: Zespół musi być mocny-nazwany i zarejestrowany w globalnej pamięci podręcznej zespołu (GAC). Montaż jest rejestrowany w COM + wyraźnie lub przez opóźnione rejestrację po zarejestrowaniu się w GAC.
- Rejestrowanie zespołu w COM +: Przygotowanie zespołu do wystawiania na potrzeby klientów COM. Następnie należy utworzyć bibliotekę typów przy użyciu narzędzia Assembly Registration, regasm.exe.

  ```
          regasm UnmanagedToManagedXa.dll
  ```

- Zarejestruj zespół w GAC `gacutil /i UnmanagedToManagedXa.dll`.
- Zarejestruj zespół w COM +, korzystając z narzędzia instalatora usług .NET, regsvcs.exe. Zapoznaj się z biblioteką typów utworzoną przez program regasm.exe:

  ```
  Regsvcs /appname:UnmanagedToManagedXa  /tlb:UnmanagedToManagedXa.tlb
  UnmanagedToManagedXa.dll
  ```

- Montaż jest wdrażany w GAC, a później jest zarejestrowany w COM + przez leniwą rejestrację. Środowisko .NET zajmuje się rejestracją po uruchomieniu kodu po raz pierwszy.

Przykładowy przepływ kodu przy użyciu modelu System.EnterpriseServices i pliku System.Transactions z COM + są opisane w następujących sekcjach:

Przykładowy przepływ kodu przy użyciu modelu System.EnterpriseServices

```
using System;
using IBM.WMQ;
using IBM.WMQ.Nmqi;
using System.Transactions;
using System.EnterpriseServices;

namespace UnmanagedToManagedXa
{

[ComVisible(true)]
[System.EnterpriseServices.Transaction(System.EnterpriseServices.TransactionOption.Required)
]
    public class MyXa : System.EnterpriseServices.ServicedComponent
    {
```

```
        public MQQueueManager QMGR = null;
        public MQQueueManager QMGR1 = null;
        public MQQueue QUEUE = null;
        public MQQueue QUEUE1 = null;
        public MQPutMessageOptions pmo = null;
        public MQMessage MSG = null;

        public MyXa()
        {
        }

        [System.EnterpriseServices.AutoComplete()]
        public void ExecuteUOW()
        {
            QMGR = new MQQueueManager("usemq");

            QUEUE = QMGR.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE",
                                     MQC.MQOO_INPUT_SHARED +
                                     MQC.MQOO_OUTPUT +
                                     MQC.MQOO_BROWSE);

            pmo = new MQPutMessageOptions();
            pmo.Options = MQC.MQPMO_SYNCPOINT;
            MSG = new MQMessage();
            QUEUE.Put(MSG, pmo);
            QMGR.Disconnect();
        }
    }
}


public void RunNow()
{
   MyXa xa = new MyXa();
   xa.ExecuteUOW();
}
```

Przykładowy przepływ kodu przy użyciu metody System.Transactions dla interakcji z COM +

```
[STAThread]
public void ExecuteUOW()
{
Hashtable t1 = new Hashtable();
t1.Add(MQC.CHANNEL_PROPERTY, "SYSTEM.DEF.SVRCONN");
t1.Add(MQC.HOST_NAME_PROPERTY, "localhost");
t1.Add(MQC.PORT_PROPERTY, 1414);
t1.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_CLIENT);
TransactionOptions opts = new TransactionOptions();

using(TransactionScope scope = new  TransactionScope(TransactionScopeOption.RequiresNew,
                               opts,     EnterpriseServicesInteropOption.Full)
  {

        QMGR = new MQQueueManager("usemq", t1);
        QUEUE = QMGR.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE",
                                       MQC.MQOO_INPUT_SHARED +
                                       MQC.MQOO_OUTPUT +
                                       MQC.MQOO_BROWSE);

         pmo = new MQPutMessageOptions();
         pmo.Options = MQC.MQPMO_SYNCPOINT;
         MSG = new MQMessage();
         QUEUE.Put(MSG, pmo);
         scope.Complete();
  }
 QMGR.Disconnect();
}
```

Tworzenie prostych komunikatów put i get w ramach TransactionScope

Przykładowe aplikacje C# produktu są dostępne w produkcie WebSphere MQ. Te proste aplikacje demonstrują wprowadzanie i pobieranie komunikatów w elemencie TransactionScope. Po zakończeniu zadania użytkownik będzie mógł umieścić i pobrać komunikaty z kolejki lub tematu.

Zanim rozpoczniesz

Usługa MSDTC musi być uruchomiona i włączona dla transakcji XA.

O tym zadaniu

Przykładem jest prosta aplikacja, `SimpleXAPut` i `SimpleXAGet`. Programy `SimpleXAPut` i `SimpleXAGet` to aplikacje C# dostępne w produkcie WebSphere MQ. Produkt `SimpleXAPut` demonstruje za pomocą wywołania MQPUT w ramach transakcji rozproszonych przy użyciu przestrzeni nazw SystemTransactions . `SimpleXAGet` demonstruje za pomocą wywołania MQGET w ramach transakcji rozproszonych przy użyciu przestrzeni nazw SystemTransactions .

`SimpleXAPut` znajduje się w `WebSphere MQ\tools\dotnet\samples\cs\base`

Procedura

Aplikacje mogą być uruchamiane z parametrami wiersza komend z programu `tools\dotnet\samples\cs\base\bin`

```
SimpleXAPut.exe -d destinationURI [-h host -p port -l channel -tx transaction -tm mode -n
numberOfMsgs]
```

```
SimpleXAGet.exe -d destinationURI [-h host -p port -l channel -tx transaction -tm mode -n
numberOfMsgs]
```

gdzie parametry są następujące:

-destinationURI

Może to być kolejka lub temat. W przypadku kolejki należy określić jako `queue://queueName` , a dla tematu określić jako `topic://topicName`.

-host

Może to być nazwa hosta, taka jak localhost lub adres IP.

-port

Port, na którym działa menedżer kolejek.

-channel

Używany kanał połączenia. Wartością domyślną jest `SYSTEM.DEF.SVRCONN`

-transaction

Wynik transakcji, na przykład zatwierdzenie lub wycofanie zmian.

-mode

Tryb transportu, na przykład zarządzany lub niezarządzany.

-numberOfMsgs

Liczba komunikatów. Wartość domyślna to 1.

Przykład

```
SimpleXAPut -d topic://T01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

```
SimpleXAGet -d queue://Q01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

Odzyskiwanie transakcji

W tej sekcji opisano proces odtwarzania transakcji w produkcie WebSphere MQ .NET XA przy użyciu trybu zarządzanego.

Przegląd

W przypadku przetwarzania transakcji rozproszonych transakcje mogą być pomyślnie zakończone. Ale mogą wystąpić sytuacje, w których transakcja może nie powieść się z wielu powodów. Przyczyny te mogą obejmować awarię systemu, awarię sprzętu, błąd sieci, niepoprawne lub niepoprawne dane, błędy aplikacji lub klęski żywiołowe lub katastrofy spowodowane przez człowieka. Zapobieganie awariom transakcji nie jest możliwe. Rozproszony system transakcyjny musi być w stanie obsługiwać te niepowodzenia. Musi być w stanie wykryć i poprawić błędy w momencie ich wystąpienia. Ten proces jest znany jako Odtwarzanie transakcji.

Istotnym aspektem przetwarzania rozproszonego transakcji jest odzyskanie niekompletnych lub wątpliwych transakcji. Konieczne jest uruchomienie odtwarzania, ponieważ część konkretnej transakcji jest wstrzymana do czasu jej odtworzenia. Microsoft .NET ze swojej biblioteki klas System.Transactions udostępnia opcję odtwarzania niekompletnych/wątpliwych transakcji. Ta obsługa odtwarzania oczekuje, że Resource Manager będzie obsługiwał dzienniki transakcji i w razie potrzeby będzie uruchamiać odtwarzanie.

Model odzyskiwania

W modelu odzyskiwania transakcji firmy Microsoft .NET menedżer transakcji (System.Transactionslub Microsoft Distributed Transaction Coordinator (MS DTC) lub oba te elementy) inicjuje, koordynuje i kontroluje odtwarzanie transakcji. Menedżery zasobów oparte na protokole OLE Tx (Microsoft's XA protocol) udostępniają opcje służące do konfigurowania DTC do kierowania, koordynowania i sterowania odtwarzając je. Aby to zrobić, menedżerowie zasobów muszą zarejestrować XA_Switch z MS DTC za pomocą rodzimego interfejsu.

Funkcja XA_Switch udostępnia punkty wejścia funkcji XA, takie jak xa_start, xa_end i xa_recover, w Resource Manager , do koordynatora transakcji rozproszonych.

Odtwarzanie przy użyciu programu Microsoft Distributed Transaction Coordinator (DTC):

Koordynator programu Microsoft Distributed Transaction udostępnia dwa rodzaje procesów odtwarzania.

Zimne odtwarzanie

Odtwarzanie na zimno jest wykonywane, jeśli proces menedżera transakcji zakończy się niepowodzeniem, gdy połączenie z menedżerem zasobów XA jest otwarte. Po zrestartowaniu menedżera transakcji odczytuje on dzienniki menedżera transakcji i ponownie nawiązuje połączenie z menedżerem zasobów XA, a następnie inicjuje odtwarzanie.

Odtwarzanie podczas pracy

Odtwarzanie podczas pracy jest wykonywane wtedy, gdy menedżer transakcji pozostaje włączony, gdy połączenie między menedżerem transakcji a menedżerem zasobów XA nie powiedzie się, ponieważ menedżer zasobów XA lub sieć nie powiodły się. Po awarii menedżer transakcji okresowo podejmuje próbę ponownego nawiązania połączenia z menedżerem zasobów XA. Po ponownym nawiązaniu połączenia menedżer transakcji inicjuje odtwarzanie XA.

Przestrzeń nazw System.Transactions udostępnia zarządzane implementację transakcji rozproszonych, które są oparte na programie MS DTC jako menedżer transakcji. Udostępnia on podobne funkcje, jak interfejs rodzimy w programie MS DTC, ale w pełni zarządzane środowisko. Jedyna różnica polega na odzyskiwaniu transakcji. System.Transactions oczekuje, że kierownicy zasobów będą prowadzić odtwarzanie samodzielnie, a następnie koordynować je z menedżerami transakcji (MS DTC). Resource Manager musi poprosić o odzyskanie konkretnej niekompletnej transakcji, a następnie menedżer transakcji akceptuje je i koordynuje w oparciu o rzeczywisty wynik tej konkretnej transakcji.

Proces odtwarzania transakcji dla środowiska .NET produktu WebSphere MQ

W tej sekcji opisano sposób, w jaki transakcje rozproszone mogą być odtwarzane z klasami .NET produktu WebSphere MQ .

Przegląd

Aby odzyskać niekompletną transakcję, wymagane są informacje o odtwarzaniu. Informacje o odtwarzaniu transakcji muszą być rejestrowane w pamięci masowej przez menedżery zasobów. Klasy środowiska .NET produktu WebSphere MQ są zgodne z podobną ścieżką. Informacje o odtwarzaniu transakcji są rejestrowane w kolejce systemowej o nazwie SYSTEM.DOTNET.XARECOVERY.QUEUE.

Odtwarzanie transakcji w produkcie WebSphere MQ .NET jest procesem dwuetapowym.

1. Rejestrowanie informacji o odtwarzaniu transakcji.
   - Dla każdej transakcji podczas fazy przygotowywania komunikat trwały zawierający informacje o odtwarzaniu jest dodawany do systemu SYSTEM.DOTNET.XARECOVERY.QUEUE.
   - Komunikat zostanie usunięty, jeśli wywołanie zatwierdzenia powiedzie się.
2. Odtwarzanie transakcji przy użyciu aplikacji monitorującego WmqDotnetXAMonitor.
   - WmqDotnetXAMonitor to zarządzana aplikacja .NET, która przetwarza komunikaty w systemie SYSTEM.DOTNET.XARECOVERY.QUEUE i odzyskuje niekompletne transakcje

   Jeśli agent MCA nie może umieścić komunikatu w kolejce docelowej, generuje raport o wyjątkach zawierający oryginalny komunikat i umieszcza go w kolejce transmisji, która ma zostać wysłana do kolejki odpowiedzi określonej w pierwotnym komunikacie. (Jeśli kolejka zwrotna znajduje się w tym samym menedżerze kolejek co agent MCA, komunikat jest umieszczany bezpośrednio w tej kolejce, a nie do kolejki transmisji).

SYSTEM.DOTNET.XARECOVERY.QUEUE

Jest to kolejka systemowa, w której przechowywane są informacje o odtwarzaniu transakcji niekompletnych transakcji. Ta kolejka jest tworzona podczas tworzenia menedżera kolejek.

Uwaga: Nie należy usuwać systemu SYSTEM.DOTNET.XARECOVERY.QUEUE .

Aplikacja WMQDotnetXAMonitor

Aplikacja WebSphere MQ .NET Monitor XA monitoruje dany menedżer kolejek i odzyskuje niekompletne transakcje, jeśli takie istnieją. Następujące transakcje są uważane za transakcje niekompletne i są odzyskane:

Niekompletne transakcje

- Jeśli transakcja jest przygotowana, ale COMMIT nie została zakończona w okresie limitu czasu.
- Jeśli transakcja jest przygotowana, ale menedżer kolejek produktu WebSphere MQ został wyłączony.
- Jeśli transakcja jest przygotowana, ale menedżer transakcji został wyłączony.

  Aplikacja monitorującego musi być uruchamiana z tego samego systemu, w którym działa aplikacja kliencka WebSphere MQ .NET. Jeśli istnieją aplikacje działające na wielu systemach łączących się z tym samym menedżerem kolejek, aplikacja monitorującego musi być uruchomiona ze wszystkich systemów. Mimo że każdy komputer kliencki ma aplikację monitorującą działającą w celu odzyskania aplikacji, każdy monitor powinien być w stanie zidentyfikować komunikat, który odpowiada transakcji, którą koordynował lokalny MS DTC w bieżącym monitorze, tak aby mógł on ponownie zarejestrować i zakończyć działanie tego monitora.

Przypadki użycia odtwarzania transakcji dla środowiska .NET produktu WebSphere MQ

Poniżej przedstawiono różne scenariusze użycia:

- **AplikacjaWebSphere MQ Application używała pojedynczej instancji DTC i pojedynczej instancji menedżera kolejek:** W tym scenariuszu po nawiązaniu połączenia z menedżerem kolejek i uruchomieniu w ramach transakcji jednostki pracy (UoW), a jeśli transakcja nie powiedzie się i stanie się niekompletna, aplikacja monitorujący odzyskuje transakcję i zakończy ją.

W tym scenariuszu zostanie uruchomiona pojedyncza instancja aplikacji monitorującego, ponieważ pojedynczy menedżer kolejek jest powiązany z transakcjami.

- **Wiele aplikacji produktu WebSphere MQ korzystających z pojedynczego DTC i pojedynczej instancji menedżera kolejek:** w tym scenariuszu istnieje więcej niż jedno aplikacje WMQ w pojedynczym DTC i wszystkie połączenia z tym samym menedżerem kolejek i z uruchomionym UoW w ramach transakcji są połączone.

  Jeśli transakcje nie powiodą się i staną się niekompletne, aplikacja monitorujący odzyskuje je i uzupełnia transakcje dotyczące wszystkich aplikacji.

  W tym scenariuszu działa jedna aplikacja monitora, ponieważ jeden menedżer kolejek jest używany w transakcjach.
- **Wiele aplikacji produktu WebSphere MQ , wiele obiektów DTC i różnych instancji menedżera kolejek:** w tym scenariuszu istnieje więcej niż jedna aplikacja WMQ w różnych DTC (czyli każda aplikacja jest uruchomiona na innym komputerze) i łączy się z różnymi menedżerami kolejek.

  Jeśli wystąpi awaria, a transakcja stanie się niekompletna, aplikacja monitorującego sprawdza, czy TransactionManagerw tym komunikacie ma być używany do określenia adresu DTC. Jeśli wartość TransactionManagerpowoduje, że wartość jest zgodna z adresem DTC, pod którym monitor jest uruchomiony, następuje zakończenie odtwarzania, a w przeciwnym razie trwa wyszukiwanie aż do znalezienia komunikatu odpowiadającego jego DTC.

  W tym scenariuszu dla każdego klienta (użytkownika lub komputera) będzie działać tylko jedna instancja aplikacji monitorującego, ponieważ każdy klient ma własny menedżer kolejek używany w transakcjach.
- **Wiele instancji produktu WebSphere MQ , wiele rekordów DTC i wiele instancji menedżerów kolejek:** w tym scenariuszu istnieje więcej niż jedna aplikacja WMQ w różnych DTC (każda aplikacja działa na innym komputerze), a wszystkie połączenia nawiązują połączenie z tym samym menedżerem kolejek.

  Jeśli wystąpi awaria, a transakcja stanie się niekompletna, aplikacja monitorującego weryfikuje obiekt TransactionManagerGdziekolwiek w komunikacie w celu sprawdzenia, czy adres DTC i wartość są zgodne z DTC, pod którym monitor jest uruchomiony. Jeśli obie wartości są zgodne, kontynuuje wyszukiwanie, dopóki nie znajdzie komunikatu odpowiadającego jego DTC.

  W tym scenariuszu zostanie uruchomiona tylko jedna instancja aplikacji monitorujących na klienta (użytkownik lub komputer), ponieważ każdy klient ma własne powiązanie menedżera kolejek używane w transakcjach.
- **Wiele aplikacji WebSphere MQ , pojedyncze DTC, różne instancje menedżera kolejek:** W tym scenariuszu istnieje więcej niż jedna aplikacja WMQ w ramach pojedynczego DTC (czyli na komputerze, na którym działa więcej niż jedna aplikacja WMQ) i łącząca się z różnymi menedżerami kolejek.

  Jeśli transakcja nie powiedzie się i stanie się niekompletna, aplikacja monitoruj odzyskuje transakcję.

  W tym scenariuszu istnieje wiele instancji aplikacji monitorujących działających jako menedżery kolejek połączonych z tym menedżerem kolejek, ponieważ każda aplikacja ma własny menedżer kolejek używany w transakcjach, a każda z nich musi być odzyskana.

  Uwaga: Jeśli aplikacja monitora nie jest uruchomiona w tle, można ją uruchomić.

Korzystanie z aplikacji WMQDotnetXAMonitor

Aplikacja monitora XA musi być uruchamiana ręcznie. Można ją uruchomić w dowolnym momencie. Można go uruchomić, gdy wyświetlane są komunikaty w systemie SYSTEM.DOTNET.XARECOVERY.QUEUE lub można ją zachować w tle, zanim wykonasz jakiekolwiek prace transakcyjne z aplikacjami napisanych przy użyciu klas WebSphere MQ .NET.

Komenda uruchamiający aplikację monitorującego

```
WmqDotnetXAMonitor.exe -m <QueueManagerName> -n <ConnectionName> -c <Channel> -i
```

Gdzie:

- **n** -Nazwa połączenia w formacie hosta (port). Nazwa połączenia może zawierać więcej niż jedną nazwę połączenia. Wiele nazw połączeń musi być podane w postaci listy rozdzielanej przecinkami, na przykład "localhost (1414), localhost (1415), localhost (1416)". Aplikacja Monitor uruchamia odtwarzanie dla każdej z nazw połączeń podanych w rozdzielanej przecinkami liście.
- **c** -Nazwa kanału.
- **m** -Nazwa menedżera kolejek. Opcjonalne
- **i** -zakończenie gałęzi heurystycznej. Opcjonalne

Aplikacja monitorującego wykonuje następujące działania:

1. Sprawdza głębokość kolejki SYSTEM.DOTNET.XARECOVERY.QUEUE w odstępie 100 sekund.
2. Jeśli głębokość kolejki jest większa niż zero, monitor XA przegląda kolejkę dla komunikatów i sprawdza, czy komunikat spełnia niekompletne kryteria transakcji.
3. Jeśli którekolwiek z komunikatów spełnia niekompletne kryteria transakcji, monitor go pobiera i pobiera informacje o odtwarzaniu transakcji.
4. Następnie określa, czy informacje o odtwarzaniu odnoszą się do lokalnego DTC. Jeśli tak, to przechodzi w celu odzyskania transakcji. W przeciwnym razie zostanie ponownie wyświetlony następny komunikat.
5. Następnie wywołuje on wywołania do menedżera kolejek w celu odzyskania niekompletnej transakcji.

Ustawienia pliku konfiguracyjnego aplikacji WmqDotNETXAMonitor
Aby można było monitorować aplikację, dane wejściowe mogą być również udostępniane przy użyciu pliku konfiguracyjnego aplikacji. Przykładowy plik konfiguracyjny aplikacji jest dostarczany razem z produktem WebSphere MQ .NET. Ten plik można modyfikować zgodnie z wymaganiami użytkownika.

Plik konfiguracyjny aplikacji ma najwyższy priorytet podczas uwzględniania wartości wejściowych. Jeśli wartości wejściowe są podane zarówno w wierszu komend, jak i w pliku konfiguracyjnym aplikacji, wówczas brane są pod uwagę wartości z konfiguracji aplikacji.

Przykładowy plik konfiguracyjny aplikacji.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<configSections>
<sectionGroup name="IBM.WMQ">
<section name="dnetxa" type="System.Configuration.NameValueFileSectionHandler" />
</sectionGroup>
</configSections>
<IBM.WMQ>
<dnetxa>
<add key="ConnectionName" value=""/>
<add key="ChannelName" value="" />
<add key="QueueManagerName" value="" />
<add key="UserId" value="" />
<add key="SecurityExit" value="" />
<add key="SecurityExitUserData" value = "">
</dnetxa>
</dnetxa>
</configuration>
```

Dziennik aplikacji WmqDotNetXAMonitor
Aplikacja Monitor tworzy plik dziennika w katalogu aplikacji, który umożliwia rejestrowanie postępu monitorowania i statusu odtwarzania transakcji. Rejestrowanie rozpoczyna się od nazwy połączenia i szczegółów kanału w celu wyświetlenia bieżącego menedżera kolejek, dla którego uruchomiono odtwarzanie.

Po uruchomieniu odtwarzania zostanie zarejestrowany identyfikator MessageId komunikatu o odtwarzaniu transakcji TransactionId niekompletnej transakcji i rzeczywisty wynik transakcji, jak na koordynację menedżera transakcji.

Przykładowy plik dziennika:

```
Time|ProcessId|ThreadId|WMQ .NET XA Recovery Monitor, Running now for
ConnectionName:xxxx, Time|ProcessId|ThreadId|Channel=xxxx
```

```
Time|ProcessId|ThreadId|Current QueueDepth = n
Time|ProcessId|ThreadId|Current MessageId = xxxx
Time|ProcessId|ThreadId|Current Incomplete Transaction being recovered = xxxxx
Time|ProcessId|ThreadId|Actual Outcome of the transaction(as per DTC)= Commit/Rollback
Time|ProcessId|ThreadId|Recovery Completed for TransactionId= xxxxx
Time|ProcessId|ThreadId|Current QueueDepth = n
Time|ProcessId|ThreadId|Current MessageId = xxxx
Time|ProcessId|ThreadId|Current Incomplete Transaction being recovered = xxxxx
Time|ProcessId|ThreadId|Actual Outcome of the transaction(as per DTC)= Commit/Rollback
Time|ProcessId|ThreadId| Recovery Completed for TransactionId= xxxxx
```

Korzystanie z klas produktu WebSphere MQ dla środowiska .NET

W tej kolekcji tematów opisano sposób konfigurowania systemu w celu uruchamiania przykładowych programów w celu sprawdzenia klas WebSphere MQ dla instalacji .NET oraz sposobu uruchamiania własnych programów.

Konfigurowanie menedżera kolejek w celu akceptowania połączeń klientów TCP/IP

Aby skonfigurować menedżera kolejek w taki sposób, aby akceptować przychodzące żądania połączeń od klientów:

1. Zdefiniuj kanał połączenia z serwerem:
   a. Uruchom menedżer kolejek.
   b. Zdefiniuj przykładowy kanał o nazwie NET.CHANNEL[3]:

```
DEF CHL('NET.CHANNEL') CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ') +
DESCR('Sample channel for WebSphere MQ classes for .NET')
```

2. Uruchom program nasłuchujący:

```
runmqlsr -t tcp [-m qmnqme] [-p portnum]
```

   Uwaga: Nawiasy kwadratowe wskazują parametry opcjonalne; *nazwa_qm* nie jest wymagany dla domyślnego menedżera kolejek, a numer portu *numer_portu* nie jest wymagany, jeśli używana jest wartość domyślna (1414).

Aplikacje przykładowe

Aby uruchomić własne aplikacje .NET, należy użyć instrukcji dla programów weryfikujących, podstawiając nazwę aplikacji w miejsce aplikacji przykładowych.

Dostarczane są pięć przykładowych aplikacji:

- Aplikacja umieszczanie komunikatów
- Aplikacja pobierania komunikatów
- Aplikacja "hello world"
- Aplikacja publikowania/subskrypcji
- Aplikacja używała właściwości komunikatu

Wszystkie te przykładowe aplikacje są dostarczane w języku C#, a niektóre z nich są również dostarczane w języku C++ i Visual Basic. Aplikacje można pisać w dowolnym języku obsługiwanym przez platformę .NET.

Program SPUT programu "Put message" (nmqsput.cs, mmqsput.cpp, vmqsput.vb)
Ten program pokazuje, jak umieścić komunikat w nazwanej kolejce. Program ma trzy parametry:

- Nazwa kolejki (wymagane), na przykład SYSTEM.DEFAULT.LOCAL.QUEUE

[3] W tym przykładzie nie rozważamy implikacji związanych z bezpieczeństwem. W przypadku systemu produkcyjnego należy rozważyć użycie protokołu SSL lub wyjścia zabezpieczeń. Więcej informacji na ten temat zawiera sekcja Zabezpieczenia .

- Nazwa menedżera kolejek (opcjonalnie)
- Definicja kanału (opcjonalnie), na przykład SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Jeśli nie zostanie podana nazwa menedżera kolejek, menedżer kolejek zostanie domyślnie ustawiony na domyślny lokalny menedżer kolejek. Jeśli kanał jest zdefiniowany, ma taki sam format, jak w przypadku zmiennej środowiskowej MQSERVER.

Program SGET programu "Get message" (nmqsget.cs, mmqsget.cpp, vmqsget.vb)
Ten program pokazuje, jak uzyskać komunikat z nazwanej kolejki. Program ma trzy parametry:

- Nazwa kolejki (wymagane), na przykład SYSTEM.DEFAULT.LOCAL.QUEUE
- Nazwa menedżera kolejek (opcjonalnie)
- Definicja kanału (opcjonalnie), na przykład SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Jeśli nie zostanie podana nazwa menedżera kolejek, menedżer kolejek zostanie domyślnie ustawiony na domyślny lokalny menedżer kolejek. Jeśli kanał jest zdefiniowany, ma taki sam format, jak w przypadku zmiennej środowiskowej MQSERVER.

Program "Hello World" (nmqwrld.cs, mmqwrld.cpp, vmqwrld.vb)
Ten program pokazuje, jak umieścić i otrzymać wiadomość. Program ma trzy parametry:

- Nazwa kolejki (opcjonalnie), na przykład SYSTEM.DEFAULT.LOCAL.QUEUE lub SYSTEM.DEFAULT.MODEL.QUEUE
- Nazwa menedżera kolejek (opcjonalnie)
- Definicja kanału (opcjonalna), na przykład SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Jeśli nazwa kolejki nie jest podana, wartością domyślną jest SYSTEM.DEFAULT.LOCAL.QUEUE. Jeśli nie zostanie podana nazwa menedżera kolejek, menedżer kolejek zostanie domyślnie ustawiony na domyślny lokalny menedżer kolejek.

Program "Publish/subscribe" (MQPubSubSample.cs)
W tym programie pokazano, jak należy używać publikowania/subskrypcji produktu WebSphere MQ . Jest on dostarczany tylko w C#. Program ma dwa parametry:

- Nazwa menedżera kolejek (opcjonalnie)
- Definicja kanału (opcjonalnie)

Program "Właściwości komunikatu" (MQMessagePropertiesSample.cs)
Ten program pokazuje, jak używać właściwości komunikatu. Jest on dostarczany tylko w C#. Program ma dwa parametry:

- Nazwa menedżera kolejek (opcjonalnie)
- Definicja kanału (opcjonalnie)

Instalację można zweryfikować, kompilując i uruchamiając te aplikacje.

Aplikacje przykładowe są instalowane w następujących lokalizacjach, zgodnie z językiem, w którym są zapisywane. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqswrld.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqsput.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqsget.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\MQPubSubSample.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\MQMessagePropertiesSample.cs

Zarządzany C++

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqswrld.cpp

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqsput.cpp

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqsget.cpp

Visual Basic

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqswrld.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqsput.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqsget.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqswrld.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqsput.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqsget.vb

Aby zbudować przykładowe aplikacje, dla każdego języka dostarczono plik wsadowy.

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\bldcssamp.bat

Plik bldcssamp.bat zawiera wiersz dla każdej próbki, która jest niezbędna do zbudowania tego przykładowego programu:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib:MQ_INSTALLATION_PATH\bin
/out:nmqwrld.exe nmqwrld.cs
```

Zarządzany C++

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\bldmcpsamp.bat

Plik bldmcpsamp.bat zawiera wiersz dla każdej próbki, która jest niezbędna do zbudowania tego przykładowego programu:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Jeśli chcesz skompilować te aplikacje w programie Microsoft Visual Studio 2003/.NET SDKv1.1, zastąp komendę kompilowania:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

z

```
cl /clr MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Visual Basic

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\bldvbsamp.bat

Plik bldvbsamp.bat zawiera wiersz dla każdej próbki, która jest niezbędna do zbudowania tego przykładowego programu:

```
vbc /r:System.dll /r:MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:vmqwrld.exe vmqwrld.vb
```

Rozwiązywanie problemów z produktem WebSphere MQ .NET

Jeśli program nie zakończy się pomyślnie, uruchom jedną z przykładowych aplikacji i postępuj zgodnie z zaleceniami podanymi w komunikatach diagnostycznych.

Te przykładowe aplikacje zostały opisane w sekcji “Korzystanie z klas produktu WebSphere MQ dla środowiska .NET” na stronie 586.

Jeśli problemy są kontynuowane, a użytkownik musi skontaktować się z zespołem serwisowym IBM , może zostać poproszony o włączenie funkcji śledzenia.

Śledzenie przykładowej aplikacji

Instrukcje dotyczące korzystania z narzędzia śledzenia można znaleźć w sekcji "Śledzenie programów WebSphere MQ .NET" na stronie 609.

Komunikaty o błędzie

Może zostać wyświetlony następujący wspólny komunikat o błędzie:

Nieobsługiwany wyjątek typu 'System.IO.FileNotFoundExceptionwystąpił w nieznanym module
Jeśli ten błąd wystąpi dla pliku amqmdnet.dll lub amqmdxcs.dll, upewnij się, że oba są zarejestrowane w 'Global Assembly Cache', lub utwórz plik konfiguracyjny wskazujący na zespoły amqmdnet.dll i amqmdxcs.dll . Zawartość pamięci podręcznej zespołu można sprawdzić i zmienić za pomocą komendy mscorcfg.msc, która jest dostarczana jako część środowiska .NET.

Jeśli środowisko .NET było niedostępne podczas instalowania produktu WebSphere MQ , klasy te mogą nie być zarejestrowane w globalnej pamięci podręcznej zespołu. Proces rejestracji można ponownie uruchomić ręcznie za pomocą komendy

```
amqidnet -c MQ_INSTALLATION_PATH\bin\amqidotn.txt -l logfile.txt
```

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

Informacje na temat tej instalacji są zapisywane w podanym pliku dziennika (w tym przykładzie`logfile.txt`).

Pisanie i wdrażanie programów WebSphere MQ .NET

Aby używać klas produktu WebSphere MQ dla środowiska .NET w celu uzyskiwania dostępu do kolejek produktu WebSphere MQ , należy pisać programy w dowolnym języku obsługiwanym przez środowisko .NET, zawierającym wywołania wstawiające komunikaty do kolejek produktu WebSphere MQ i pobierające je z nich.

Dokumentacja produktu WebSphere MQ zawiera informacje tylko w językach C#, C++ i Visual Basic.

Ta kolekcja tematów zawiera informacje pomocne w pisaniu aplikacji w celu interakcji z systemami WebSphere MQ . Szczegółowe informacje na temat poszczególnych klas zawiera sekcja Klasy i interfejsy środowiska .NET produktu WebSphere MQ.

Różnice między połączeniami

Sposób, w jaki program WebSphere MQ .NET ma zależności od trybów połączenia, które mają być używane.

Połączenia zarządzane przez klienta

Jeśli klasy produktu WebSphere MQ dla środowiska .NET są używane jako klient zarządzany, istnieje wiele różnic ze standardowego klienta MQI produktu WebSphere MQ .

Następujące opcje nie są dostępne dla klienta zarządzanego:

- Kompresja kanału
- Obsługa protokołu SSL
- Łączenie wyjścia kanału

W przypadku próby użycia tych opcji z klientem zarządzanym zostanie zwrócony wyjątek MQException. Jeśli błąd zostanie wykryty na końcu połączenia klienta, zostanie użyty kod przyczyny MQRC_ENVIRONMENT_ERROR. Jeśli zostanie on wykryty na końcu serwera, zostanie użyty kod przyczyny zwrócony przez serwer.

Wyjścia kanału zapisane dla niezarządzanego klienta nie działają. Należy napisać nowe wyjścia specjalnie dla zarządzanego klienta. Sprawdź, czy w tabeli definicji kanału klienta (CCDT) nie określono żadnych niepoprawnych wyjść kanału.

Nazwa wyjścia kanału zarządzanego może mieć długość do 999 znaków. Jeśli jednak do określenia nazwy wyjścia kanału zostanie użyta wartość CCDT, to będzie ona ograniczona do 128 znaków.

Komunikacja jest obsługiwana tylko przez protokół TCP/IP.

Jeśli menedżer kolejek zostanie zatrzymany za pomocą komendy **endmqm** , kanał połączenia z serwerem zarządzanym przez klienta .NET może zająć więcej czasu niż kanały połączenia z serwerem innym klientom.

Jeśli parametr *NMQ_MQ_LIB* jest ustawiony na wartość `managed` , aby możliwe było korzystanie z zarządzanej diagnostyki problemów produktu WebSphere MQ , nie jest obsługiwana żadna z parametrów -i, -p, -s, -b lub -c komendy **strmqtrc** .

Zarządzana aplikacja .NET używała transakcji XA nie będzie działać z menedżerem kolejek produktu z/OS . Próba nawiązania połączenia z menedżerem kolejek produktu z/OS przez klienta sieciowego kończy się niepowodzeniem z błędem, MQRC_UOW_ENLISTMENT_ERROR (mqrc=2354) w wywołaniu MQOPEN. Jednak zarządzana aplikacja .NET używała transakcji XA będzie działać z rozproszonym menedżerem kolejek.

Definiowanie typu połączenia, który ma być używany

Typ połączenia jest określany na podstawie ustawienia nazwy połączenia, nazwy kanału, wartości dostosowania NMQ_MQ_LIB i właściwości MQC.TRANSPORT_PROPERTY.

Nazwę połączenia można określić w następujący sposób:

- Jawnie w konstruktorze MQQueueManager :

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- W tym celu należy ustawić właściwości MQC.HOST_NAME_PROPERTY i opcjonalnie MQC.PORT_PROPERTY w pozycji tabeli mieszającej w konstruktorze MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Jako jawne wartości MQEnvironment

```
MQEnvironment.Hostname
```

  `MQEnvironment.Port`(opcjonalnie).
- W tym celu należy ustawić właściwości MQC.HOST_NAME_PROPERTY i opcjonalnie MQC.PORT_PROPERTY w tabeli mieszającej MQEnvironment.properties .

Nazwę kanału można określić w następujący sposób:

- Jawnie w konstruktorze MQQueueManager :

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- W tym celu należy ustawić właściwość MQC.CHANNEL_PROPERTY w postaci tabeli mieszającej w konstruktorze MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Jako jawna wartość MQEnvironment

```
MQEnvironment.Channel
```

- W tym celu należy ustawić właściwość MQC.CHANNEL_PROPERTY w tabeli mieszającej MQEnvironment.properties .

Właściwość transportu można określić w następujący sposób:

- W tym celu należy ustawić właściwość MQC.TRANSPORT_PROPERTY we wpisie tabeli mieszającej w konstruktorze MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- W tym celu należy ustawić właściwość MQC.TRANSPORT_PROPERTY w tabeli hashtable MQEnvironment.properties .

Wybierz wymagany typ połączenia, używając jednej z następujących wartości:

MQC.TRANSPORT_MQSERIES_BINDINGS -połączenie jako serwer
MQC.TRANSPORT_MQSERIES_CLIENT -nawiąż połączenie jako klient inny niż XA
MQC.TRANSPORT_MQSERIES_XACLIENT -połączenie jako klient XA
MQC.TRANSPORT_MQSERIES_MANAGED -połączenie jako klient zarządzany bez interfejsu XA

Wartość dostosowania NMQ_MQ_LIB można ustawić w taki sposób, aby typ połączenia był jawnie wybierany w sposób przedstawiony w poniższej tabeli.

Wartość NMQ_MQ_LIB	**Typ połączenia**
mqic.dll	Połącz jako klient inny niż XA
mqicxa.dll	Połącz jako klient XA
mqm.dll	Połącz jako serwer lub jako klient inny niż XA
zarządzany	Połącz jako klient zarządzany inny niż XA
Uwaga: Wartości parametrów mqic32.dll i mqic32xa.dll są akceptowane jako synonimy plików mqic.dll i mqicxa.dll w celu zapewnienia zgodności z wcześniejszymi wersjami. Jednak pliki mqm.dll i mqm.pdb są tylko częścią pakietu klienta, począwszy od wersji 7.1 .	

Jeśli zostanie wybrany typ połączenia, który nie jest dostępny w danym środowisku, na przykład określony zostanie plik mqic32xa.dll i nie ma obsługi interfejsu XA, produkt WebSphere MQ .NET zgłasza wyjątek.

Ustawienie wartości NMQ_MQ_LIB na wartość "managed" powoduje, że klient będzie używać zarządzanych testów diagnostycznych problemów z produktem WebSphere MQ , konwersji danych .NET i innych zarządzanych funkcji WebSphere MQ na niskim poziomie.

Wszystkie pozostałe wartości właściwości NMQ_MQ_LIB powodują, że proces .NET jest używany do korzystania z niezarządzanych testów diagnostycznych i testów problemów z produktem WebSphere MQ , a także innych niezarządzanych funkcji WebSphere MQ niskiego poziomu (przy założeniu, że w systemie jest zainstalowany klient lub serwer WebSphere MQ MQI).

Produkt WebSphere MQ .NET wybiera typ połączenia w następujący sposób:

1. Jeśli MQC.TRANSPORT_PROPERTY , która łączy się z wartością MQC.TRANSPORT_PROPERTY.

   Należy jednak pamiętać, że ustawienie MQC.TRANSPORT_PROPERTY do MQC.TRANSPORT_MQSERIES_MANAGED nie gwarantuje, że proces klienta jest uruchamiany. Nawet w przypadku tego ustawienia klient nie jest zarządzany w następujących przypadkach:

- Jeśli inny wątek w procesie nawiąże połączenie z MQC.TRANSPORT_PROPERTY jest ustawiona na wartość inną niż MQC.TRANSPORT_MQSERIES_MANAGED.
- Jeśli wartość NMQ_MQ_LIB nie jest ustawiona na "managed", testy diagnostyczne problemów, konwersja danych i inne funkcje niskiego poziomu nie są w pełni zarządzane (zakładając, że w systemie jest zainstalowany klient lub klient MQI produktu WebSphere MQ).

2. Jeśli nazwa połączenia została określona bez nazwy kanału lub nazwa kanału została określona bez nazwy połączenia, to zgłasza błąd.
3. Jeśli określono zarówno nazwę połączenia, jak i nazwę kanału:
   - Jeśli wartość NMQ_MQ_LIB jest ustawiona na wartość mqic32xa.dll, łączy się ona jako klient XA.
   - Jeśli wartość NMQ_MQ_LIB jest ustawiona na zarządzany, łączy się ona jako klient zarządzany.
   - W przeciwnym razie łączy się ono jako klient inny niż XA.
4. Jeśli wartość NMQ_MQ_LIB jest określona, łączy się ona zgodnie z wartością NMQ_MQ_LIB.
5. Jeśli serwer WebSphere MQ jest zainstalowany, łączy się on jako serwer.
6. Jeśli zainstalowany jest klient MQI produktu WebSphere MQ , łączy się on jako klient inny niż XA.
7. W przeciwnym razie łączy się on jako klient zarządzany.

Pliki konfiguracyjne dla klas produktu WebSphere MQ dla środowiska .NET

Aplikacja kliencka .NET może używać pliku konfiguracyjnego klienta MQI produktu WebSphere MQ oraz, jeśli jest używany typ połączenia zarządzanego, plik konfiguracyjny aplikacji .NET. Ustawienia w pliku konfiguracyjnym aplikacji mają priorytet.

plik konfiguracyjny klienta

Aplikacja kliencka klasy WebSphere MQ dla klienta .NET może używać pliku konfiguracyjnego klienta w taki sam sposób, jak w przypadku dowolnego innego klienta MQI produktu WebSphere MQ . Ten plik zwykle nosi nazwę mqclient.ini, ale można podać inną nazwę pliku. Więcej informacji na temat pliku konfiguracyjnego klienta zawiera sekcja Konfigurowanie klienta przy użyciu pliku konfiguracyjnego WebSphere MQ -plik konfiguracyjny klienta MQI.

Tylko następujące atrybuty w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ są istotne dla klas produktu WebSphere MQ dla środowiska .NET. Jeśli zostaną określone inne atrybuty, nie będzie to miało żadnego wpływu.

sekcja	**Atrybut**
Kanały	CCSID
Kanały	Katalog ChannelDefinition
Kanały	Plik ChannelDefinition
Kanały	Parametry ServerConnection
ŚcieżkaClientExit	ExitsDefaultPath
ŚcieżkaClientExit	ExitsDefaultPath64
MessageBuffer	MaximumSize
MessageBuffer	PurgeTime
MessageBuffer	UpdatePercentage
TCP	ClntRcvBufSize
TCP	ClntSndBufSize
TCP	IPAddressVersion

sekcja	Atrybut
TCP	KeepAlive

Dowolne z tych atrybutów można przesłonić, używając odpowiedniej zmiennej środowiskowej.

Plik konfiguracyjny aplikacji

Jeśli używany jest typ połączenia zarządzanego, można także przesłonić plik konfiguracyjny klienta WebSphere MQ oraz równoważne zmienne środowiskowe za pomocą pliku konfiguracyjnego aplikacji .NET.

Ustawienia pliku konfiguracyjnego aplikacji .NET działają tylko wtedy, gdy są uruchomione z typem połączenia zarządzanego i są ignorowane dla innych typów połączeń.

Plik konfiguracyjny aplikacji .NET i jego format są definiowane przez firmę Microsoft dla ogólnego zastosowania w środowisku .NET, ale poszczególne nazwy sekcji, klucze i wartości wymienione w tej dokumentacji są specyficzne dla produktu Websphere MQ.

Format pliku konfiguracyjnego aplikacji .NET jest następujący: *sekcje*. Każda sekcja zawiera jeden lub więcej *kluczy*, a każdy klucz ma powiązaną *wartość*. W poniższym przykładzie przedstawiono sekcje, klucze i wartości używane w pliku konfiguracyjnym aplikacji .NET do sterowania właściwością TCP/IP KeepAlive :

```
<configuration>
  <configSections>
    <section name="TCP" type="System.Configuration.NameValueSectionHandler"/>
  </configSections>
  <TCP>
    <add key="KeepAlive" value="true"></add>
  </TCP>
<configuration>
```

Słowa kluczowe używane w nazwach sekcji i kluczach w sekcji pliku konfiguracyjnego aplikacji .NET są dokładnie zgodne ze słowami kluczowymi dla sekcji i atrybutów zdefiniowanych w pliku konfiguracyjnym klienta.

Więcej informacji na ten temat zawiera dokumentacja firmy Microsoft .

Przykładowy fragment kodu

Poniższy fragment kodu C# przedstawia aplikację, która wykonuje trzy działania:

1. Nawiązywanie połączenia z menedżerem kolejek
2. Umieść komunikat w systemie SYSTEM.DEFAULT.LOCAL.QUEUE
3. Pobierz komunikat z powrotem

Przedstawiono również sposób zmiany typu połączenia.

```
// ============================================================================
// Licensed Materials - Property of IBM
// 5724-H72
// (c) Copyright IBM Corp. 2003, 2024
// ============================================================================
using System;
using System.Collections;

using IBM.WMQ;

class MQSample
{
  // The type of connection to use, this can be:-
  // MQC.TRANSPORT_MQSERIES_BINDINGS for a server connection.
  // MQC.TRANSPORT_MQSERIES_CLIENT for a non-XA client connection
  // MQC.TRANSPORT_MQSERIES_XACLIENT for an XA client connection
  // MQC.TRANSPORT_MQSERIES_MANAGED for a managed client connection
  const String connectionType = MQC.TRANSPORT_MQSERIES_CLIENT;
```

```
// Define the name of the queue manager to use (applies to all connections)
const String qManager = "your_Q_manager";

// Define the name of your host connection (applies to client connections only)
const String hostName = "your_hostname";

// Define the name of the channel to use (applies to client connections only)
const String channel = "your_channelname";



/// <summary>
/// Initialise the connection properties for the connection type requested
/// </summary>
/// <param name="connectionType">One of the MQC.TRANSPORT_MQSERIES_ values</param>
static Hashtable init(String connectionType)
{
  Hashtable connectionProperties = new Hashtable();

  // Add the connection type
  connectionProperties.Add(MQC.TRANSPORT_PROPERTY, connectionType);

  // Set up the rest of the connection properties, based on the
  // connection type requested
  switch(connectionType)
  {
    case MQC.TRANSPORT_MQSERIES_BINDINGS:
      break;
    case MQC.TRANSPORT_MQSERIES_CLIENT:
    case MQC.TRANSPORT_MQSERIES_XACLIENT:
    case MQC.TRANSPORT_MQSERIES_MANAGED:
      connectionProperties.Add(MQC.HOST_NAME_PROPERTY, hostName);
      connectionProperties.Add(MQC.CHANNEL_PROPERTY, channel);
      break;
  }

  return connectionProperties;
}
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static int Main(string[] args)
{
  try
  {
    Hashtable connectionProperties = init(connectionType);

    // Create a connection to the queue manager using the connection
    // properties just defined
    MQQueueManager qMgr = new MQQueueManager(qManager, connectionProperties);

    // Set up the options on the queue we want to open
    int openOptions = MQC.MQOO_INPUT_AS_Q_DEF | MQC.MQOO_OUTPUT;

    // Now specify the queue that we want to open,and the open options
    MQQueue system_default_local_queue =
      qMgr.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE", openOptions);

    // Define a WebSphere MQ message, writing some text in UTF format
    MQMessage hello_world = new MQMessage();
    hello_world.WriteUTF("Hello World!");

    // Specify the message options
    MQPutMessageOptions pmo = new MQPutMessageOptions(); // accept the defaults,
                                                         // same as MQPMO_DEFAULT

    // Put the message on the queue
    system_default_local_queue.Put(hello_world, pmo);



    // Get the message back again

    // First define a WebSphere MQ message buffer to receive the message
    MQMessage retrievedMessage =new MQMessage();
    retrievedMessage.MessageId =hello_world.MessageId;

    // Set the get message options
    MQGetMessageOptions gmo =new MQGetMessageOptions(); //accept the defaults
                                                        //same as MQGMO_DEFAULT
```

```
      // Get the message off the queue
      system_default_local_queue.Get(retrievedMessage,gmo);

      // Prove we have the message by displaying the UTF message text
      String msgText = retrievedMessage.ReadUTF();
      Console.WriteLine("The message is: {0}", msgText);

      // Close the queue
      system_default_local_queue.Close();

      // Disconnect from the queue manager
      qMgr.Disconnect();
    }

    //If an error has occurred,try to identify what went wrong.

    //Was it a WebSphere MQ error?
    catch (MQException ex)
    {
      Console.WriteLine("A WebSphere MQ error occurred: {0}", ex.ToString());
    }

    catch (System.Exception ex)
    {
      Console.WriteLine("A System error occurred: {0}", ex.ToString());
    }

    return 0;
  }//end of start
}//end of sample
```

Operacje na menedżerach kolejek

W tej sekcji opisano sposób łączenia i rozłączenia menedżera kolejek przy użyciu klas produktu WebSphere MQ dla platformy .NET.

Konfigurowanie środowiska produktu WebSphere MQ

Przed użyciem połączenia klienta w celu nawiązania połączenia z menedżerem kolejek należy skonfigurować środowisko WebSphere MQ .

Uwaga: Ten krok nie jest konieczny w przypadku używania klas WebSphere MQ classes for .NET w trybie powiązań serwera.

Interfejs programistyczny .NET umożliwia użycie wartości dostosowania NMQ_MQ_LIB, ale obejmuje również klasę MQEnvironment. Ta klasa umożliwia określenie szczegółów, które mają być używane podczas próby nawiązania połączenia, na przykład tych z poniższej listy:

- Nazwa kanału
- Nazwa hosta
- Numer portu
- Wyjścia kanału
- Parametry SSL
- Nazwa i hasło użytkownika

Pełne informacje na temat klasy MQEnvironment można znaleźć w sekcji KlasaMQEnvironment .NET .

Aby określić nazwę kanału i nazwę hosta, należy użyć następującego kodu:

```
MQEnvironment.Hostname = "host.domain.com";
MQEnvironment.Channel  = "client.channel";
```

Domyślnie klienty próbują połączyć się z programem nasłuchującym produktu WebSphere MQ na porcie 1414. Aby określić inny port, należy użyć kodu:

```
MQEnvironment.Port = nnnn;
```

Nawiąże połączenie z menedżerem kolejek

Teraz można nawiązać połączenie z menedżerem kolejek, tworząc nową instancję klasy MQQueueManager :

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Aby rozłączyć się z menedżerem kolejek, wywołaj metodę `Disconnect` w menedżerze kolejek:

```
queueManager.Disconnect();
```

Podczas próby nawiązania połączenia z menedżerem kolejek konieczne jest sprawdzenie uprawnień (`inq`) w menedżerze kolejek. Próba nawiązania połączenia nie powiedzie się bez uzyskiwania informacji o uprawnieniach.

Jeśli zostanie wywołana metoda `Disconnect` , wszystkie otwarte kolejki i procesy, do których użytkownik uzyskuje dostęp za pośrednictwem tego menedżera kolejek, są zamykane. Jednak dobrym rozwiązaniem programowym jest zamknięcie tych zasobów w sposób jawny po zakończeniu korzystania z nich. Aby zamknąć zasoby, należy użyć metody `Close` w obiekcie powiązanym z każdym zasobem.

Metody `Commit` i `Backout` w menedżerze kolejek zastępują wywołania MQCMIT i MQBACK, które są używane z interfejsem proceduralnym.

Dostęp do kolejek i tematów

Dostęp do kolejek i tematów można uzyskać za pomocą metod MQQueueManager lub odpowiednich konstruktorów.

Aby uzyskać dostęp do kolejek, należy użyć metod klasy MQQueueManager . Tabela MQOD (struktura deskryptora obiektu) jest zwinięta do parametrów tych metod. Na przykład, aby otworzyć kolejkę w menedżerze kolejek reprezentowanym przez obiekt MQQueueManager o nazwie queueManager, należy użyć następującego kodu:

```
MQQueue queue = queueManager.AccessQueue("qName",
                                         MQC.MQOO_OUTPUT,
                                         "qMgrName",
                                         "dynamicQName",
                                         "altUserId");
```

Parametr *options* jest taki sam, jak parametr Options w wywołaniu MQOPEN.

Metoda AccessQueue zwraca nowy obiekt klasy MQQueue.

Po zakończeniu korzystania z kolejki należy użyć metody Close (), aby ją zamknąć, tak jak w następującym przykładzie:

```
queue.Close();
```

Za pomocą produktu WebSphere MQ .NET można także utworzyć kolejkę przy użyciu konstruktora MQQueue. Parametry są dokładnie takie same, jak w przypadku metody accessQueue , wraz z dodaniem parametru menedżera kolejek określającego instancję obiektu MQQueueManager , która ma być używana. Na przykład:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            MQC.MQOO_OUTPUT,
```

```
                                "qMgrName",
                                "dynamicQName",
                                "altUserId");
```

Konstruowanie obiektu kolejki w ten sposób umożliwia zapisywanie własnych podklas kolejki MQQueue.

Podobnie można również uzyskać dostęp do tematów przy użyciu metod klasy MQQueueManager . Aby otworzyć temat, należy użyć metody AccessTopic(). Zwraca nowy obiekt klasy MQTopic. Po zakończeniu korzystania z tematu należy użyć metody Close () w MQTopic, aby ją zamknąć.

Istnieje również możliwość utworzenia tematu przy użyciu konstruktora MQTopic. Istnieje pewna liczba konstruktorów dla tematów; więcej informacji na ten temat zawiera sekcja KlasaMQTopic .NET.

Obsługa komunikatów

Komunikaty są obsługiwane przy użyciu metod z klas kolejek lub tematów. Aby zbudować nowy komunikat, utwórz nowy obiekt MQMessageobject.

Umieszczanie komunikatów w kolejkach lub tematach przy użyciu metody Put () klasy MQQueue lub MQTopic. Pobieranie komunikatów z kolejek lub tematów przy użyciu metody Get () klasy MQQueue lub MQTopic. W przeciwieństwie do interfejsu proceduralnego, w którym tabele MQPUT i MQGET są wstawiane i otrzymujemy tablice bajtów, klasy WebSphere MQ dla środowiska .NET wstawiają i pobiera instancje klasy MQMessage. Klasa MQMessage hermetykuje bufor danych, który zawiera rzeczywiste dane komunikatu, wraz ze wszystkimi parametrami MQMD (deskryptor komunikatu) opisujących ten komunikat.

Aby zbudować nowy komunikat, należy utworzyć nową instancję klasy MQMessage i użyć metod WriteXXX w celu umieszczenia danych w buforze komunikatów.

Po utworzeniu nowej instancji komunikatu wszystkie parametry MQMD są automatycznie ustawiane na wartości domyślne, zgodnie z definicją w sekcji Wartości początkowe i deklaracje języków dla deskryptora MQMD . Metoda Put () kolejki MQQueue pobiera również instancję klasy opcji MQPutMessagejako parametr. Ta klasa reprezentuje strukturę MQPMO. W poniższym przykładzie tworzony jest komunikat i umieszcza go w kolejce:

```
// Build a new message containing my age followed by my name
MQMessage myMessage = new MQMessage();
myMessage.WriteInt(25);

String name = "Charlie Jordan";
myMessage.WriteUTF(name);

// Use the default put message options...
MQPutMessageOptions pmo = new MQPutMessageOptions();

// put the message!
queue.Put(myMessage,pmo);
```

Metoda Get () kolejki MQQueue zwraca nową instancję komunikatu MQMessage, która reprezentuje komunikat, który został właśnie odebrany z kolejki. Pobiera ona również instancję klasy opcji MQGetMessagejako parametr. Ta klasa reprezentuje strukturę MQGMO.

Nie ma potrzeby określania maksymalnej wielkości komunikatu, ponieważ metoda Get () automatycznie dostosowuje wielkość swojego wewnętrznego buforu, aby pasował do komunikatu przychodzącego. Aby uzyskać dostęp do danych w zwróconej wiadomości, należy użyć metod ReadXXX klasy MQMessage.

W poniższym przykładzie przedstawiono sposób pobrania komunikatu z kolejki:

```
// Get a message from the queue
MQMessage theMessage    = new MQMessage();
MQGetMessageOptions gmo = new MQGetMessageOptions();
queue.Get(theMessage,gmo);  // has default values

// Extract the message data
int age = theMessage.ReadInt();
String name1 = theMessage.ReadUTF();
```

Istnieje możliwość zmiany formatu liczb, którego używają metody odczytu i zapisu, ustawiając zmienną elementu *encoding* .

Zestaw znaków używany do odczytu i zapisu łańcuchów można zmienić, ustawiając zmienną składową *characterSet* .

Więcej informacji na ten temat zawiera sekcja KlasaMQMessage .NET .

Uwaga: Metoda WriteUTF() obiektu MQMessage automatycznie koduje długość łańcucha, a także bajty Unicode, które zawiera. Gdy komunikat zostanie odczytany przez inny program .NET (za pomocą komendy ReadUTF()), jest to najprostszy sposób wysyłania informacji łańcuchowych.

Obsługa właściwości komunikatu

Właściwości komunikatu umożliwiają wybór komunikatów lub pobieranie informacji na temat komunikatu bez uzyskiwania dostępu do jego nagłówków. Klasa MQMessage zawiera metody pobierania i ustawiania właściwości.

Za pomocą właściwości komunikatu można zezwolić aplikacji na wybieranie komunikatów do przetwarzania lub pobieranie informacji o komunikacie bez uzyskiwania dostępu do nagłówków MQMD lub MQRFH2 . Ułatwiają one także komunikację między aplikacjami WebSphere MQ i JMS. Więcej informacji na temat właściwości komunikatu w produkcie WebSphere MQmożna znaleźć w sekcji Właściwości komunikatu.

Klasa MQMessage udostępnia wiele metod pobierania i ustawiania właściwości, zgodnie z typem danych właściwości. Metody get mają nazwy formatu Get * Property, a metody set mają nazwy w formacie Set * Property, gdzie gwiazdka (*) reprezentuje jeden z następujących łańcuchów:

- wartość boolowska
- Bajt
- Bajty
- Podwójna
- Liczba zmiennopozycyjna
- Wew.
- Int2
- Int4
- Int8
- Długa liczba całkowita
- Obiekt
- Krótki
- Łańcuch

Na przykład, aby uzyskać właściwość myproperty (łańcuch znaków) WebSphere MQ , należy użyć wywołania `message.GetStringProperty('myproperty')`. Opcjonalnie można przekazać deskryptor właściwości, który WebSphere MQ zakończy działanie.

Obsługa błędów

Obsługa błędów wynikających z klas produktu WebSphere MQ dla środowiska .NET przy użyciu bloków `try` i `catch` .

Metody w interfejsie .NET nie zwracają kodu zakończenia i kodu przyczyny. Zamiast tego zgłaszają one wyjątek, gdy kod zakończenia i kod przyczyny wynikające z wywołania WebSphere MQ nie są jednocześnie zerami. Upraszcza to logikę programu, tak aby nie było konieczne sprawdzanie kodów powrotu po każdym wywołaniu funkcji WebSphere MQ. Możesz zdecydować, w jakich punktach w programie chcesz poradzić

sobie z możliwością awarii. W tych punktach można surround kodu za pomocą bloków `try` i `catch` , jak w następującym przykładzie:

```
try
{
  myQueue.Put(messageA,PutMessageOptionsA);
  myQueue.Put(messageB,PutMessageOptionsB);
}
catch (MQException ex)
{
  // This block of code is only executed if one of
  // the two put methods gave rise to a non-zero
  // completion code or reason code.
  Console.WriteLine("An error occurred during the put operation:" +
                        "CC = " + ex.CompletionCode +
                        "RC = " + ex.ReasonCode);
  Console.WriteLine("Cause exception:" + ex );
}
```

Pobieranie i ustawianie wartości atrybutów

Klasy MQManagedObject, MQQueue i MQQueueManager zawierają metody, które umożliwiają uzyskanie i ustawienie ich wartości atrybutów. Należy zwrócić uwagę, że w przypadku kolejki MQQueue metody działają tylko wtedy, gdy podczas otwierania kolejki użytkownik określi odpowiednie opcje zapytania i ustawienia.

W przypadku wspólnych atrybutów klasy MQQueueManager i MQQueue dziedziczą z klasy o nazwie MQManagedObject. Ta klasa definiuje interfejsy Inquire () i Set ().

Po utworzeniu nowego obiektu menedżera kolejek za pomocą operatora *nowy* jest on automatycznie otwierany na potrzeby zapytania. Jeśli do uzyskania dostępu do obiektu kolejki używana jest metoda AccessQueue(), obiekt ten *nie* jest automatycznie otwierany na potrzeby operacji sprawdzania lub ustawiania, może to spowodować problemy z niektórymi typami kolejek zdalnych. Aby użyć metod Inquire i Set oraz ustawić właściwości w kolejce, należy określić odpowiednie opcje zapytania i ustawienia w parametrze openOptions metody AccessQueue().

Metody zapytania i ustawiania przyjmują trzy parametry:

- tablica selektorów
- Tablica intAttrs
- Tablica charAttrs

Nie są potrzebne parametry SelectorCount, IntAttrCount i CharAttrLength, które znajdują się w tabeli MQINQ, ponieważ długość tablicy jest zawsze znana. W poniższym przykładzie przedstawiono sposób wykonania zapytania w kolejce:

```
//inquire on a queue
int [ ] selectors = new int [2] ;
int [ ] intAttrs = new int [1] ;
byte [ ] charAttrs = new byte [MQC.MQ_Q_DESC_LENGTH];
selectors [0] = MQC.MQIA_DEF_PRIORITY;
selectors [1] = MQC.MQCA_Q_DESC;
queue.Inquire(selectors,intAttrs,charAttrs);
ASCIIEncoding enc = new ASCIIEncoding();
String s1 = "";
s1 = enc.GetString(charAttrs);
```

Wszystkie atrybuty tych obiektów mogą być zapytania. Podzbiór atrybutów jest prezentowany jako właściwości obiektu. Lista atrybutów obiektów znajduje się w sekcji Atrybuty obiektów. Informacje o właściwościach obiektu można znaleźć w opisie odpowiedniej klasy.

Programy wielowątkowe

Środowisko wykonawcze .NET jest z natury wielowątkowe. Klasy produktu WebSphere MQ dla środowiska .NET umożliwiają współużytkowanie obiektu menedżera kolejek w wielu wątkach, ale zapewnia, że wszystkie uprawnienia dostępu do docelowego menedżera kolejek są synchronizowane.

Rozważmy prosty program, który łączy się z menedżerem kolejek i otwiera kolejkę przy starcie. Program wyświetla na ekranie pojedynczy przycisk. Gdy użytkownik kliknie ten przycisk, program pobierze komunikat z kolejki. W takiej sytuacji inicjowanie aplikacji odbywa się w jednym wątku, a kod wykonywany w odpowiedzi na naciśnięcie przycisku jest wykonywany w osobnym wątku (wątek interfejsu użytkownika).

Implementacja produktu WebSphere MQ .NET zapewnia, że dla określonego połączenia (instancja obiektuMQQueueManager) wszystkie uprawnienia dostępu do docelowego menedżera kolejek produktu WebSphere MQ są synchronizowane. Domyślne zachowanie polega na tym, że wątek, który chce wywołać połączenie z menedżerem kolejek, jest blokowany do momentu zakończenia wszystkich innych wywołań w toku dla tego połączenia. Jeśli wymagany jest jednoczesny dostęp do tego samego menedżera kolejek z wielu wątków w programie, należy utworzyć nowy obiekt MQQueueManager dla każdego wątku, który wymaga współbieżnego dostępu. (Jest to równoznaczne z wywołaniem oddzielnego wywołania MQCONN dla każdego wątku).

Jeśli domyślne opcje połączenia są nadpisywane przez MQC.MQCNO_HANDLE_SHARE_NONE lub MQC.MQCNO_SHARE_NO_BLOCK : menedżer kolejek nie jest już zsynchronizowany.

Korzystanie z tabeli definicji kanału klienta w środowisku .NET

Za pomocą tabeli definicji kanału klienta (CCDT) można użyć klas .NET dla produktu WebSphere MQ. Położenie tabeli definicji kanału klienta można określić na różne sposoby, w zależności od tego, czy używane jest połączenie zarządzane, czy niezarządzane.

Typ połączenia niezarządzanego klienta, który nie jest zarządzany przez XA lub XA

W przypadku niezarządzanego typu połączenia można określić położenie tabeli definicji kanału klienta na dwa sposoby:

- Za pomocą zmiennych środowiskowych MQCHLLIB należy określić katalog, w którym znajduje się tabela, oraz wartość MQCHLTAB, aby określić nazwę pliku tabeli.
- Korzystanie z pliku konfiguracyjnego klienta. W sekcji CHANNELS użyj atrybutów ChannelDefinitionDirectory, aby określić katalog, w którym znajduje się tabela, oraz plik ChannelDefinition, aby określić nazwę pliku.

Jeśli położenie jest określone zarówno w pliku konfiguracyjnym klienta, jak i przy użyciu zmiennych środowiskowych, to zmienne środowiskowe mają pierwszeństwo. Za pomocą tej funkcji można określić standardowe położenie w pliku konfiguracyjnym klienta i przesłonić je za pomocą zmiennych środowiskowych, gdy jest to konieczne.

Typ połączenia zarządzanego klienta

W przypadku typu połączenia zarządzanego można określić położenie tabeli definicji kanału klienta na trzy sposoby:

- Korzystanie z pliku konfiguracyjnego aplikacji .NET. W sekcji CHANNELS użyj kluczy ChannelDefinitionDirectory, aby określić katalog, w którym znajduje się tabela, oraz plik ChannelDefinition, aby określić nazwę pliku.
- Za pomocą zmiennych środowiskowych MQCHLLIB należy określić katalog, w którym znajduje się tabela, oraz wartość MQCHLTAB, aby określić nazwę pliku tabeli.
- Korzystanie z pliku konfiguracyjnego klienta. W sekcji CHANNELS użyj atrybutów ChannelDefinitionDirectory, aby określić katalog, w którym znajduje się tabela, oraz plik ChannelDefinition, aby określić nazwę pliku.

Jeśli położenie jest określone w więcej niż jednym z tych sposobów, zmienne środowiskowe mają pierwszeństwo przed plikiem konfiguracyjnym klienta, a plik konfiguracyjny aplikacji .NET ma pierwszeństwo przed obydwoma innymi metodami. Za pomocą tej funkcji można określić standardowe położenie w pliku konfiguracyjnym klienta i przesłonić je za pomocą zmiennych środowiskowych lub pliku konfiguracyjnego aplikacji, gdy jest to konieczne.

W jaki sposób aplikacja .NET określa definicję kanału, która ma być używana

W środowisku klienta WebSphere MQ .NET definicja kanału, która ma być używana, może być określona na wiele różnych sposobów. Może istnieć wiele specyfikacji definicji kanału. Aplikacja wywodzi definicję kanału z jednego lub większej liczby źródeł.

Jeśli istnieje więcej niż jedna definicja kanału, to jest on wybierany w następującej kolejności priorytetów:

1. Właściwości określone w konstruktorze MQQueueManager , jawnie lub poprzez włączenie opcji *MQC.CHANNEL_PROPERTY* we właściwościach hashtable
2. Właściwość *MQC.CHANNEL_PROPERTY* w tabeli mieszającej MQEnvironment.properties
3. Właściwość *Kanał* w środowisku MQEnvironment
4. Plik konfiguracyjny aplikacji .NET, nazwa sekcji CHANNELS, klucz ServerConnectionParms (ma zastosowanie tylko do połączeń zarządzanych)
5. Zmienna środowiskowa *MQSERVER*
6. Plik konfiguracyjny klienta, sekcja CHANNELS, atrybut ServerConnectionParms
7. Tabela definicji kanału klienta (CCDT). Położenie tabeli definicji kanału klienta jest określone w pliku konfiguracyjnym aplikacji .NET (dotyczy tylko połączeń zarządzanych).
8. Tabela definicji kanału klienta (CCDT). Położenie tabeli definicji kanału klienta jest określane przy użyciu zmiennych środowiskowych *MQCHLIB* i *MQCHLTAB* .
9. Tabela definicji kanału klienta (CCDT). Położenie tabeli definicji kanału klienta jest określane przy użyciu pliku konfiguracyjnego klienta.

W przypadku pozycji 1-3, definicja kanału jest wbudowana w pole według wartości z wartości udostępnionych przez aplikację. Te wartości mogą być udostępniane przy użyciu różnych interfejsów i dla każdego z nich może istnieć wiele wartości. Wartości pól są dodawane do definicji kanału zgodnie z podanym porządkiem priorytetowym:

1. Wartość parametru *connName* w konstruktorze MQQueueManager
2. Wartości właściwości z tabeli mieszającej MQQueueManager.properties
3. Wartości właściwości z tabeli mieszającej MQEnvironment.properties
4. Wartości ustawione jako pola MQEnvironment (na przykład MQEnvironment.Hostname, MQEnvironment.Port)

W przypadku pozycji 4-6, cała definicja kanału jest dostarczana jako wartość. Nieokreślone pola w definicji kanału przyjmują wartości domyślne systemowe. Żadne wartości z innych metod definiowania kanałów i ich pól nie są scalane z tymi specyfikacjami.

W przypadku pozycji 7-9 cała definicja kanału jest pobierana z tabeli definicji kanału klienta. Pola, które nie zostały określone jawnie, gdy kanał został zdefiniowany, przyjmują wartości domyślne systemowe. Żadne wartości z innych metod definiowania kanałów i ich pól nie są scalane z tymi specyfikacjami.

Korzystanie z wyjść kanału w programie IBM WebSphere MQ .NET

Jeśli używane są powiązania klienta, można użyć wyjść kanału, tak jak w przypadku innych połączeń klienckich. Jeśli używane są powiązania zarządzane, należy napisać program obsługi wyjścia, który implementuje odpowiedni interfejs.

Powiązania klienta

Jeśli używane są powiązania klienta, można użyć wyjść kanału zgodnie z opisem w sekcji Wyjścia kanałów. Nie można używać wyjść kanału napisanych dla powiązań zarządzanych.

Powiązania zarządzane

W przypadku korzystania z połączenia zarządzanego w celu zaimplementowania wyjścia należy zdefiniować nową klasę .NET, która implementuje odpowiedni interfejs. W pakiecie WebSphere MQ zdefiniowane są trzy interfejsy wyjścia:

- MQSendExit
- MQReceiveExit
- MQSecurityExit

Uwaga: Wyjścia użytkownika napisane przy użyciu tych interfejsów nie są obsługiwane jako wyjścia kanału w środowisku niezarządzanym.

Poniższy przykład definiuje klasę, która implementuje wszystkie trzy:

```
class MyMQExits : MQSendExit, MQReceiveExit, MQSecurityExit
{
   // This method comes from the send exit
  byte[] SendExit(MQChannelExit       channelExitParms,
                  MQChannelDefinition channelDefinition,
                  byte[]              dataBuffer
                  ref int             dataOffset
                  ref int             dataLength
                  ref int             dataMaxLength)
  {
     // complete the body of the send exit here
  }


  // This method comes from the receive exit
  byte[] ReceiveExit(MQChannelExit       channelExitParms,
                     MQChannelDefinition channelDefinition,
                     byte[]              dataBuffer
                     ref int             dataOffset
                     ref int             dataLength
                     ref int             dataMaxLength)
  {
     // complete the body of the receive exit here
  }


  // This method comes from the security exit
  byte[] SecurityExit(MQChannelExit       channelExitParms,
                      MQChannelDefinition channelDefParms,
                      byte[]              dataBuffer
                      ref int             dataOffset
                      ref int             dataLength
                      ref int             dataMaxLength)
  {
     // complete the body of the security exit here
  }

}
```

Każde wyjście jest przekazywane przez obiekt MQChannelExit i instancję obiektu MQChannelDefinition . Obiekty te reprezentują struktury MQCXP i MQCD zdefiniowane w interfejsie proceduralnym.

Dane, które mają być wysyłane przez wyjście wysyłania, oraz dane odebrane w zabezpieczeniach lub wyjściu odbieranym są określane przy użyciu parametrów wyjścia.

W przypadku pozycji dane w pozycji *dataOffset* o długości *dataLength* w tablicy bajtów *dataBuffer* to dane, które mają zostać wysłane przez wyjście wysyłania, a także dane odebrane w ramach wyjścia bezpieczeństwa lub wyjścia odbierania. Parametr *dataMaxLength* podaje maksymalną długość (od *dataOffset*) dostępną dla wyjścia w *dataBuffer*. Uwaga: W przypadku wyjścia zabezpieczeń dataBuffer może mieć wartość NULL, jeśli jest to pierwsze wywołanie wyjścia lub gdy partner nie został wybrany do wysłania żadnych danych.

Po powrocie wartość *dataOffset* i *dataLength* powinny być ustawione tak, aby wskazywały na przesunięcie i długość w zwróconej tablicy bajtów, która powinna być następnie używana przez klasy .NET.
W przypadku wyjścia wysyłania oznacza to, że dane, które powinny zostać wysłane, a także dla wyjścia bezpieczeństwa lub wyjścia odbierania, powinny być interpretowane. Wyjście powinno zwykle zwracać tablicę bajtów; wyjątki to wyjście zabezpieczeń, które może zostać wybrane do wysłania bez danych, a każde wyjście wywołane z przyczyn INIT lub TERM. Najprostszą formą wyjścia, która może zostać zapisana, jest jedna, która nie wykonuje więcej niż zwraca dataBuffer:

Najprostszym możliwym organem wyjściowym jest:

```
{
  return dataBuffer;
}
```

Klasa MQChannelDefinition

V 7.5.0.6 W produkcie Version 7.5.0, Fix Pack 6identyfikator użytkownika i hasło określone za pomocą aplikacji klienta zarządzanego .NET są ustawiane w klasie IBM WebSphere MQ .NET MQChannelDefinition , która jest przekazywana do wyjścia zabezpieczeń klienta. Wyjście zabezpieczeń kopiuje identyfikator użytkownika i hasło do katalogu MQCD.RemoteUserIdentifier i MQCD.RemotePassword (patrz “Zapisywanie wyjścia zabezpieczeń” na stronie 416).

Określanie wyjść kanału (klient zarządzany)

Jeśli podczas tworzenia obiektu MQQueueManager (w środowisku MQEnvironment lub w konstruktorze MQQueueManager) zostanie określona nazwa kanału i nazwa połączenia, można określić wyjścia kanału na dwa sposoby.

W kolejności, w jakiej są one stosowane, są to:

1. Przekazywanie właściwości hashtable MQC.SECURITY_EXIT_PROPERTY, MQC.SEND_EXIT_PROPERTY lub MQC.RECEIVE_EXIT_PROPERTY w konstruktorze MQQueueManager .
2. Ustawianie właściwości MQEnvironment SecurityExit, SendExit lub ReceiveExit .

Jeśli nazwa kanału i nazwa połączenia nie zostaną określone, wyjście kanału do użycia pochodzi z definicji kanału odezanej z tabeli definicji kanału klienta (CCDT). Nie jest możliwe przesłonięcie wartości zapisanych w definicji kanału. Więcej informacji na temat tabel definicji kanału zawiera sekcja Tabela definicji kanału klienta i “Korzystanie z tabeli definicji kanału klienta w środowisku .NET” na stronie 600 .

W każdym przypadku specyfikacja przyjmuje formę łańcucha o następującym formacie:

```
Assembly_name(Class_name)
```

nazwa_klasy to pełna nazwa, w tym specyfikacja przestrzeni nazw, klasy .NET, która implementuje IBM.WMQ.MQSecurityExit, IBM.WMQ.MQSendExit lub IBM.WMQ.MQReceiveExit (w zależności od przypadku). *nazwa_zespołu* to pełna nazwa zespołu, w tym rozszerzenie nazwy pliku, w którym znajduje się klasa. Długość łańcucha jest ograniczona do 999 znaków, jeśli używane są właściwości obiektu MQEnvironment lub MQQueueManager. Jeśli jednak nazwa wyjścia kanału jest określona w tabeli definicji kanału klienta, to jest ona ograniczona do 128 znaków. Jeśli jest to konieczne, kod klienta .NET jest ładowany i tworzy instancję określonej klasy, analizując specyfikację łańcucha.

Określanie danych użytkownika wyjścia kanału (klient zarządzany)

Wyjścia kanału mogą zawierać powiązane z nimi dane użytkownika. Jeśli podczas tworzenia obiektu MQQueueManager zostanie określona nazwa kanału i nazwa połączenia (w środowisku MQEnvironment lub w konstruktorze MQQueueManager), dane użytkownika można określić na dwa sposoby.

W kolejności, w jakiej są one stosowane, są to:

1. Przekazywanie właściwości hashtable MQC.SECURITY_USERDATA_PROPERTY, MQC.SEND_USERDATA_PROPERTY lub MQC.RECEIVE_USERDATA_PROPERTY w konstruktorze MQQueueManager .
2. Ustawianie właściwości danych SecurityUserprogramu MQEnvironment, danych SendUserlub ReceiveUserdanych.

Jeśli nazwa kanału i nazwa połączenia nie zostaną określone, wartości danych użytkownika wyjścia, które mają być używane, pochodzą z definicji kanału odezanej z tabeli definicji kanału klienta (CCDT). Nie jest możliwe przesłonięcie wartości zapisanych w definicji kanału. Więcej informacji na temat tabel definicji kanału zawiera sekcja Tabela definicji kanału klienta i “Korzystanie z tabeli definicji kanału klienta w środowisku .NET” na stronie 600 .

W każdym przypadku specyfikacja jest łańcuchem, ograniczonym do 32 znaków.

Automatyczne ponowne połączenie klienta w środowisku .NET

Klient może automatycznie ponownie nawiązać połączenie z menedżerem kolejek podczas nieoczekiwanego przerwania połączenia.

Klient może nieoczekiwanie zostać odłączony od menedżera kolejek, jeśli na przykład menedżer kolejek zostanie zatrzymany lub nastąpi awaria sieci lub serwera.

Bez automatycznego ponownego nawiązywania połączenia z klientem generowany jest błąd, gdy połączenie nie powiedzie się. Można użyć kodu błędu, aby pomóc w ponownym nawiązaniu połączenia.

Klient korzystający z funkcji automatycznego ponownego połączenia klienta jest nazywany klientem z możliwością ponownego połączenia. Aby utworzyć klienta z możliwością ponownego połączenia, należy określić określone opcje o nazwie opcje ponownego połączenia podczas nawiązywania połączenia z menedżerem kolejek.

Jeśli aplikacja kliencka jest klientem WebSphere MQ .NET, może on zdecydować się na automatyczne ponowne połączenie klienta, określając odpowiednią wartość dla CONNECT_OPTIONS_PROPERTY, jeśli do utworzenia menedżera kolejek używana jest klasa `MQQueueManager` . Szczegółowe informacje na temat wartości CONNECT_OPTIONS_PROPERTY zawiera sekcja Opcje rekolekcji .

Użytkownik może wybrać, czy aplikacja kliencka zawsze łączy się i ponownie łączy z menedżerem kolejek o tej samej nazwie, z tym samym menedżerem kolejek lub w dowolnym zestawie menedżerów kolejek, które są zdefiniowane z tą samą nazwą QMNAME w tabeli połączeń klienta (szczegółowe informacje zawiera sekcja Grupy menedżera kolejek w tabeli CCDT).

Obsługa protokołu SSL (Secure Sockets Layer)

Poniższa sekcja nie ma zastosowania do klienta zarządzanego.

Klasy WebSphere MQ dla aplikacji klienckich .NET obsługują szyfrowanie Secure Sockets Layer (SSL). Protokół SSL zapewnia szyfrowanie komunikacji, uwierzytelnianie i integralność komunikatów. Jest on zwykle używany do zabezpieczania komunikacji między dowolnymi dwoma równorzędnymi płatnikami w sieci Internet lub w intranecie.

Włączanie protokołu SSL

Protokół SSL jest obsługiwany tylko dla połączeń klienckich. Aby włączyć protokół SSL, należy określić parametr CipherSpec , który ma być używany podczas komunikowania się z menedżerem kolejek. Musi on być zgodny z wartością atrybutu CipherSpec ustawioną w kanale docelowym.

Aby włączyć protokół SSL, należy określić wartość parametru CipherSpec przy użyciu statycznej zmiennej składowej SSLCipherSpec produktu MQEnvironment. Poniższy przykład przyłącza się do kanału SVRCONN o nazwie SECURE.SVRCONN.CHANNEL, który został skonfigurowany tak, aby wymagał połączenia SSL z atrybutem CipherSpec o wartości NULL_MD5:

```
MQEnvironment.Hostname         = "your_hostname";
MQEnvironment.Channel          = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.SSLCipherSpec    = "NULL_MD5";
MQEnvironment.SSLKeyRepository = "C:\mqm\key";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Listę CipherSpecszawiera sekcja Określanie specyfikacji CipherSpecs .

Właściwość SSLCipherSpec można także ustawić przy użyciu parametru MQC.SSL_CIPHER_SPEC_PROPERTY w tabeli mieszającej właściwości połączenia.

Aby pomyślnie nawiązać połączenie przy użyciu protokołu SSL, magazyn kluczy klienta musi być skonfigurowany z łańcuchem certyfikatów głównych ośrodka certyfikacji, z którego certyfikat prezentowany przez menedżer kolejek może zostać uwierzytelniony. Podobnie, jeśli parametr SSLClientAuth w kanale SVRCONN został ustawiony na wartość MQSSL_CLIENT_AUTH_REQUIRED,

magazyn kluczy klienta musi zawierać identyfikujące certyfikat osobisty, który jest zaufany przez menedżer kolejek.

Korzystanie z nazwy wyróżniającej menedżera kolejek

Menedżer kolejek identyfikuje się za pomocą certyfikatu SSL, który zawiera *nazwę wyróżniającą* (DN).

Aplikacja kliencka WebSphere MQ .NET może używać tej nazwy wyróżniającej, aby zapewnić komunikację z poprawnym menedżerem kolejek. Wzorzec nazwy wyróżniającej jest określany przy użyciu zmiennej nazwy sslPeernazwy środowiska MQEnvironment. Na przykład:

```
MQEnvironment.SSLPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

Umożliwia nawiązanie połączenia tylko wtedy, gdy menedżer kolejek wyświetli certyfikat o nazwie Common Name rozpoczynający się od QMGR., i co najmniej dwie nazwy jednostek organizacyjnych, z których pierwszym musi być IBM i drugi WEBSPHERE.

Właściwość SSLPeerName można także ustawić przy użyciu parametru MQC.SSL_PEER_NAME_PROPERTY w tabeli mieszającej właściwości połączenia. Więcej informacji na temat nazw wyróżniających i reguł ustawiania nazw węzłów sieci można znaleźć w sekcji Zabezpieczenia.

Jeśli parametr SSLPeerName jest ustawiony, połączenia powiodą się tylko wtedy, gdy zostanie ustawiony na poprawny wzorzec, a menedżer kolejek wyświetli pasujący certyfikat.

Obsługa błędów podczas korzystania z protokołu SSL

Następujące kody przyczyny mogą być wydawane przez klasy WebSphere MQ classes for .NET podczas nawiązywania połączenia z menedżerem kolejek przy użyciu protokołu SSL:

MQRC_SSL_NOT_ALLOWED
Właściwość SSLCipherSpec została ustawiona, ale używane były połączenia powiązań. Tylko połączenie klienta obsługuje protokół SSL.

MQRC_SSL_PEER_NAME_MISMATCH
Wzorzec nazwy wyróżniającej określony we właściwości SSLPeerName nie jest zgodny z nazwą wyróżniającą podaną przez menedżer kolejek.

MQRC_SSL_PEER_NAME_ERROR-BŁĄD
Wzorzec nazwy wyróżniającej określony we właściwości SSLPeerName nie jest poprawny.

Korzystanie z programu .NET Monitor

Informacje na temat ważnych informacji można znaleźć w sekcji Funkcje, które mogą być używane tylko z instalacją podstawową w systemie Windows .

Monitor .NET jest aplikacją podobną do monitora wyzwalacza WebSphere MQ . Można utworzyć komponenty .NET, których instancje są tworzone za każdym razem, gdy komunikat jest odbierany w monitorowanej kolejce, a następnie przetwarzali ten komunikat. Monitor .NET jest uruchamiany za pomocą komendy `runmqdnm` i zatrzymany za pomocą komendy `endmqdnm` . Szczegółowe informacje na temat tych komend można znaleźć w sekcji runmqdnm i endmqdnm.

Aby użyć programu .NET Monitor, należy napisać komponent implementujący interfejs IMQObjectTrigger , który jest zdefiniowany w pliku amqmdnm.dll.

Komponenty mogą być transakcyjne lub nietransakcyjne. Komponent transakcyjny musi dziedziczyć z pliku System.EnterpriseServices.ServicedComponent i musi być zarejestrowany jako RequiresTransaction lub SupportsTransaction. Nie może ona być zarejestrowana jako RequiresNew , ponieważ program .NET Monitor zainicjował już transakcję.

Komponent otrzymuje obiekty MQQueueManager, MQQueue i MQMessage z `runmqdnm`. Może on również odbierać łańcuch parametru użytkownika, jeśli został określony, przy użyciu opcji wiersza komend *-u* , gdy uruchomiono `runmqdnm` . Należy zauważyć, że komponent odbiera treść komunikatu, który dotarł do monitorowanej kolejki w obiekcie MQMessage. Nie ma potrzeby nawiązywania połączenia z menedżerem

kolejek, otwierania kolejki lub pobierania samego komunikatu. Komponent musi następnie przetworzyć komunikat jako odpowiedni i zwrócić kontrolę do monitora .NET.

Jeśli komponent został zapisany jako komponent transakcyjny, zarejestruje się w celu zatwierdzenia lub wycofania transakcji przy użyciu narzędzi udostępnianych przez komponent System.EnterpriseServices.ServicedComponent.

Ponieważ komponent odbiera zarówno obiekty MQQueueManager , jak i obiekty MQQueue, a także komunikat, ma pełne informacje kontekstowe dla tego komunikatu i może na przykład otworzyć inną kolejkę w tym samym menedżerze kolejek bez konieczności osobnego łączenia się z produktem WebSphere MQ.

Przykładowe fragmenty kodu

Ten temat zawiera dwa przykłady komponentów, które uzyskują komunikat z monitora .NET Monitor i drukują go, jeden z nich korzysta z przetwarzania transakcyjnego, a drugi z przetwarzania nietransakcyjnego. Trzeci przykład przedstawia wspólne procedury narzędziowe, które mają zastosowanie do dwóch pierwszych przykładów. Wszystkie przykłady znajdują się w C#.

Przykład 1: Przetwarzanie transakcyjne

```
/*********************************************************************/
/* Licensed materials, property of IBM                               */
/* 63H9336                                                           */
/* (C) Copyright IBM Corp. 2005, 2024.                               */
/*********************************************************************/
using System;
using System.EnterpriseServices;

using IBM.WMQ;
using IBM.WMQMonitor;

[assembly: ApplicationName("dnmsamp")]

// build:
//
// csc -target:library -reference:amqmdnet.dll;amqmdnm.dll TranAssembly.cs
//
// run (with dotnet monitor)
//
// runmqdnm -m <QMNAME> -q <QNAME> -a dnmsamp.dll -c Tran

namespace dnmsamp
{
  [TransactionAttribute(TransactionOption.Required)]
  public class Tran : ServicedComponent, IMQObjectTrigger
  {
    Util util = null;

    [AutoComplete(true)]
    public void Execute(MQQueueManager qmgr, MQQueue queue,
        MQMessage message, string param)
    {
      util = new Util("Tran");

      if (param != null)
        util.Print("PARAM: '" +param.ToString() + "'");

      util.PrintMessage(message);

      //System.Console.WriteLine("SETTING ABORT");
      //ContextUtil.MyTransactionVote = TransactionVote.Abort;

      System.Console.WriteLine("SETTING COMMIT");
      ContextUtil.SetComplete();
      //ContextUtil.MyTransactionVote = TransactionVote.Commit;
    }
  }
}
```

Przykład 2: Przetwarzanie nietransakcyjne

```
/*********************************************************************/
/* Licensed materials, property of IBM                               */
/* 63H9336                                                           */
/* (C) Copyright IBM Corp. 2005, 2024.                               */
/*********************************************************************/

using System;

using IBM.WMQ;
using IBM.WMQMonitor;

// build:
//
// csc -target:library -reference:amqmdnet.dll;amqmdnm.dll NonTranAssembly.cs
//
// run (with dotnet monitor)
//
// runmqdnm -m <QMNAME> -q <QNAME> -a dnmsamp.dll -c NonTran
namespace dnmsamp
{
  public class NonTran : IMQObjectTrigger
  {
    Util util = null;

    public void Execute(MQQueueManager qmgr, MQQueue queue,
        MQMessage message, string param)
    {
      util = new Util("NonTran");

      try
      {
        util.PrintMessage(message);
      }

      catch (Exception ex)
      {
        System.Console.WriteLine(">>> NonTran\n{0}", ex.ToString());
      }
    }
  }
}
```

Przykład 3: procedury wspólne

```
/*********************************************************************/
/* Licensed materials, property of IBM                               */
/* 63H9336                                                           */
/* (C) Copyright IBM Corp. 2005, 2024.                               */
/*********************************************************************/

using System;

using IBM.WMQ;

namespace dnmsamp
{
    /// <summary>
    /// Summary description for Util.
    /// </summary>
    public class Util
    {
    /* ------------------------------------------------------------------ */
    /* Default prefix string of the namespace.                             */
    /* ------------------------------------------------------------------ */
    private string prefixText = "dnmsamp";

    /* ------------------------------------------------------------------ */
    /* Constructor that takes the replacement prefix string to use.       */
    /* ------------------------------------------------------------------ */
    public Util(String text)
    {
      prefixText = text;
    }
```

```
    /* ------------------------------------------------------------------ */
    /* Display an arbitrary string to the console.                        */
    /* ------------------------------------------------------------------ */
    public void Print(String text)
    {
      System.Console.WriteLine("{0} {1}\n", prefixText, text);
    }


 /* ------------------------------------------------------------------ */
    /* Display the content of the message passed to the console.          */
    /* ------------------------------------------------------------------ */
    public void PrintMessage(MQMessage message)
    {
      if (message.Format.CompareTo(MQC.MQFMT_STRING) == 0)
      {
        try
        {
          string messageText = message.ReadString(message.MessageLength);

          Print(messageText);
        }

        catch(Exception ex)
        {
          Print(ex.ToString());
        }
      }
      else
      {
        Print("UNRECOGNISED FORMAT");
      }
    }

    /* ------------------------------------------------------------------ */
    /* Convert the byte array into a hex string.                          */
    /* ------------------------------------------------------------------ */
    static public string ToHexString(byte[] byteArray)
    {
      string hex = "0123456789ABCDEF";

      string retString = "";

      for(int i = 0; i < byteArray.Length; i++)
      {
        int h = (byteArray[i] & 0xF0)>>4;
        int l = (byteArray[i] & 0x0F);

        retString += hex.Substring(h,1) + hex.Substring(l,1);
      }

      return retString;
    }

    }
}
```

Kompilowanie programów WebSphere MQ .NET

Przykładowe komendy służące do kompilowania aplikacji .NET napisanych w różnych językach.

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

Aby zbudować aplikację C# przy użyciu klas WebSphere MQ dla środowiska .NET, należy użyć następującej komendy:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib:MQ_INSTALLATION_PATH\bin /out:MyProg.exe MyProg.cs
```

Aby zbudować aplikację Visual Basic za pomocą klas produktu WebSphere MQ dla platformy .NET, należy użyć następującej komendy:

```
vbc /r:System.dll /r:MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:MyProg.exe MyProg.vb
```

Aby zbudować aplikację Zarządzaną C++ przy użyciu klas WebSphere MQ dla platformy .NET, należy użyć następującej komendy:

```
cl /clr MQ_INSTALLATION_PATH\bin Myprog.cpp
```

W przypadku innych języków należy zapoznać się z dokumentacją dostarczonej przez dostawcę języka.

Śledzenie programów WebSphere MQ .NET

W produkcie WebSphere MQ .NET można uruchamiać i sterować narzędziem śledzenia, tak jak w programach WebSphere MQ za pomocą interfejsu MQI.

Jednak parametry -i i -p komendy strmqtrc, które pozwalają na określenie identyfikatorów procesów i wątków oraz nazwanych procesów, nie mają żadnego efektu.

Funkcja śledzenia jest zwykle używana tylko na żądanie usługi IBM .

Informacje na temat komend śledzenia można znaleźć w sekcji Korzystanie ze śledzenia w systemie Windows .

Kanał niestandardowy produktu IBM WebSphere MQ dla produktu Microsoft Windows Communication Foundation (WCF)

Kanał niestandardowy produktu Microsoft Windows Communication Foundation (WCF) dla produktu IBM WebSphere MQ wysyła i odbiera komunikaty między klientami i usługami WCF.

Pojęcia pokrewne

“Wprowadzenie do korzystania z kanału niestandardowego produktu WebSphere MQ dla systemu WCF z platformą .NET 3” na stronie 609
Przegląd informacji dostępnych dla programistów korzystających z niestandardowego kanału WebSphere MQ dla platformy Windows Communication Foundation (WCF) z platformą .NET 3.

“Korzystanie z niestandardowych kanałów produktu WebSphere MQ dla produktu WCF” na stronie 613
Przegląd informacji dostępnych dla programistów korzystających z niestandardowych kanałów WebSphere MQ V7 for Windows Communication Foundation (WCF).

“Korzystanie z przykładów WCF” na stronie 631
Przykłady Windows Communication Foundation (WCF) zawierają kilka prostych przykładów, w jaki sposób można użyć niestandardowego kanału WebSphere MQ .

“Określanie problemu w kanale niestandardowym WCF dla produktu WebSphere MQ” na stronie 638
Za pomocą funkcji śledzenia produktu WebSphere MQ można gromadzić szczegółowe informacje na temat różnych części kodu produktu WebSphere MQ . W przypadku korzystania z programu Windows Communication Foundation (WCF) dla niestandardowego śledzenia kanału WCF zintegrowanego ze śledzeniem infrastruktury Microsoft WCF generowane jest oddzielne dane wyjściowe śledzenia.

Wprowadzenie do korzystania z kanału niestandardowego produktu WebSphere MQ dla systemu WCF z platformą .NET 3

Przegląd informacji dostępnych dla programistów korzystających z niestandardowego kanału WebSphere MQ dla platformy Windows Communication Foundation (WCF) z platformą .NET 3.

Jaki jest niestandardowy kanał produktu WebSphere MQ dla WCF?

Kanał niestandardowy dla produktu WebSphere MQ jest kanałem transportowym korzystaniem z ujednoliconego modelu programowania WCF (Microsoft Windows Communication Foundation-WCF).

Środowisko Microsoft Windows Communication Foundation, wprowadzone w środowisku Microsoft .NET 3, umożliwia tworzenie aplikacji i usług .NET niezależnie od transportu i protokołów używanych do ich łączenia, co umożliwia użycie alternatywnych transportów lub konfiguracji w zależności od środowiska, w którym wdrożono usługę lub aplikację.

Połączenia są zarządzane w czasie wykonywania przez system WCF, budując stos kanałów zawierający wymaganą kombinację następujących elementów:

- Elementy protokołu: opcjonalny zestaw elementów, w których żaden, jeden lub więcej nie może być dodany do protokołów obsługi, takich jak normy WS-*.
- Koder komunikatów: obowiązkowy element w stosie sterujący serializacją komunikatu do jego formatu łącznika.
- Kanał transportowy: obowiązkowy element w stosie odpowiedzialny za transport serializowanego komunikatu do jego punktu końcowego.

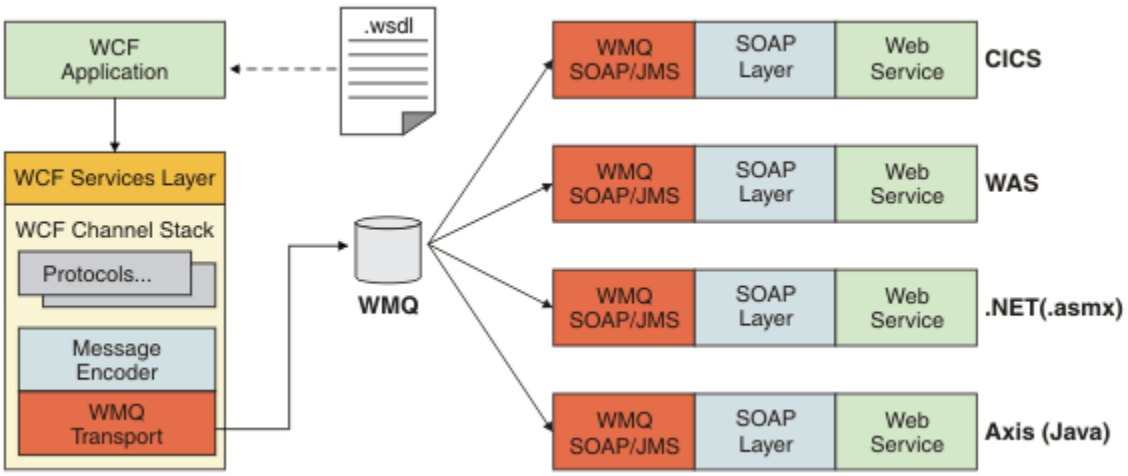
Kanał niestandardowy dla produktu WebSphere MQ jest kanałem transportowym i jako taki musi być sparowany z koderem komunikatów i opcjonalnymi protokołami, które są wymagane przez aplikację przy użyciu niestandardowego powiązania WCF. W ten sposób aplikacje, które zostały opracowane w celu użycia produktu WCF, mogą używać kanału niestandardowego dla produktu WebSphere MQ do wysyłania i odbierania danych w taki sam sposób, w jaki korzystają z wbudowanych transportów udostępnianych przez firmę Microsoft, co umożliwia prostą integrację z asynchronicznymi, skalowalną i niezawodną funkcją przesyłania komunikatów w produkcie WebSphere MQ. Pełną listę obsługiwanych funkcji można znaleźć pod adresem: “Niestandardowe funkcje i możliwości kanału WCF” na stronie 614.

Kiedy i dlaczego używany jest niestandardowy kanał produktu WebSphere MQ dla WCF?

Kanał niestandardowy produktu WebSphere MQ może być używany do wysyłania i odbierania komunikatów między klientami i usługami WCF w taki sam sposób, jak wbudowane transporty udostępniane przez firmę Microsoft, umożliwiając aplikacjom uzyskiwanie dostępu do funkcji produktu WebSphere MQ w ramach ujednoliconego modelu programistycznego WCF.

Typowy scenariusz wzorca użycia kanału niestandardowego produktu WebSphere MQ dla produktu WCF jest jako interfejs dla usług Web Service udostępnianych w produkcie WebSphere MQ (SOAP/JMS).

Komunikaty są przenoszone przy użyciu formatu komunikatu SOAP over JMS produktu WebSphere MQ, umożliwiając klientom i usługom WCF również wywołanie lub wywołanie innych aplikacji produktu WebSphere MQ lub środowisk usług serwerowych zgodnych z tym formatem, w tym usług Web Service i klientów działających w produkcie WebSphere Application Server, CICS, Axis v1 (Java), i .asmx (.NET), jak pokazano na poniższym diagramie:



Szczegółowe informacje na temat protokołu SOAP-JMS zawiera sekcja: “Transport produktu WebSphere MQ dla protokołu SOAP” na stronie 973

Przykładem typowego scenariusza z diagramu jest:

1. Usługa Web Service udostępniana na serwerze WebSphere Application Server i udostępniana za pośrednictwem produktu WebSphere MQ przy użyciu obsługi protokołu SOAP korzystający z usługi JMS w produkcie WebSphere Application Server.
2. Dokument WSDL opisujący usługę może być następnie używany przez narzędzie WCF do generowania proxy klienta i konfiguracji, które następnie tworzą odpowiedni stos kanałów WCF, w tym kanał niestandardowy.
3. Aplikacja kliencka może następnie użyć proxy w celu uruchomienia usługi Web Service w taki sam sposób, jak każda inna usługa Web Service.

Kanał ten jest zwykle używany z koderem komunikatów tekst/SOAP WCF, ale kanał może być w parze z innymi koderami komunikatów WCF, jeśli jest to wymagane. Użycie alternatywnych enkoderów może również zapewnić ograniczoną integrację z rodzimymi aplikacjami produktu WebSphere MQ , które nie obsługują protokołu SOAP przez JMS, ale nie jest to podstawowa rola kanału.

Kluczowe korzyści wynikające z używania niestandardowego kanału w środowisku WCF są następujące:

- Wywołanie asynchroniczne: Obsługa ognia i zapominania o operacjach klienta, w których klient jest oddzielony od dostępności usługi i funkcji, takich jak przekierowywanie odpowiedzi i wieloprzeskokowe.
- Charakterystyka skalowania niezawodnego: przesyłanie komunikatów oparte na kolejce umożliwia przewidywanie możliwości przewidywania wielkości w systemie.
- Jakość usługi: Wiadomości są namacalne i możliwe do śledzenia i mogą być łatwo zarządzane i administrowane.

Wymagania dotyczące oprogramowania i instrukcje instalacji dla niestandardowego kanału produktu WebSphere MQ dla produktu WCF

W tym temacie przedstawiono wymagania dotyczące oprogramowania oraz informacje dotyczące instalacji kanału niestandardowego produktu WebSphere MQ dla produktu WCF.

Kanał niestandardowy produktu WebSphere MQ dla produktu WCF może łączyć się tylko z menedżerami kolejek produktu WebSphere MQ V7 lub wyższymi.

Wymagania dotyczące oprogramowania dla niestandardowego kanału WCF dla produktu WebSphere MQ

Ta sekcja zawiera listę wymagań programowych dla niestandardowego kanału WCF dla produktu WebSphere MQ.

Środowisko wykonawcze

- Produkt Microsoft .NET Framework w wersji v3.0 lub nowszej musi być zainstalowany na komputerze hosta.
- Usługi *Java i .NET Messaging and Web Services* są instalowane domyślnie jako część instalatora produktu WebSphere MQ 7.0.1 . Instaluje zespoły .NET wymagane dla kanału niestandardowego w pamięci podręcznej zespołu Global Assembly Cache.

Uwaga: Jeśli środowisko Microsoft .NET Framework w wersji v2.0 lub nowszej nie jest zainstalowane przed zainstalowaniem produktu WebSphere MQ V7.0.1, instalacja produktu WebSphere MQ będzie kontynuowana bez błędów, ale kanał niestandardowy produktu WebSphere MQ jest niedostępny. Jeśli środowisko .NET Framework jest zainstalowane po zainstalowaniu produktu WebSphere MQ 7.0.1, wówczas niestandardowy kanał produktu WebSphere MQ musi zostać aktywowany przez uruchomienie skryptu *WMQInstallDir*\bin\amqiRegisterdotNet.cmd , gdzie *WMQInstallDir* to katalog, w którym zainstalowano produkt WebSphere MQ 7.0.1 . Ten skrypt zainstaluje wymagane zespoły w globalnej pamięci podręcznej zespołu (Global Assembly Cache-GAC). W katalogu %TEMP% tworzony jest zestaw plików produktu amqi*.log , które rejestrują działania. Nie jest konieczne ponowne uruchomienie skryptu amqiRegisterdotNet.cmd , jeśli platforma .NET została zaktualizowana do wersji v3.0 lub nowszej z wcześniejszej wersji, na przykład z platformy .NET v2.0.

Środowisko programistyczne

- Microsoft Visual Studio 2008 lub Windows Software Development Kit for .NET 3.0 lub nowszy.
- Aby można było zbudować przykładowe pliki rozwiązania, na komputerze hosta musi być zainstalowany produkt Microsoft .NET Framework w wersji V3.5 lub nowszej.

Uwaga: Jeśli środowisko Microsoft .NET Framework w wersji v2.0 lub nowszej nie jest zainstalowane przed zainstalowaniem produktu WebSphere MQ V7.0.1, instalacja produktu WebSphere MQ będzie kontynuowana bez błędów, ale kanał niestandardowy produktu WebSphere MQ jest niedostępny. Jeśli środowisko .NET Framework jest zainstalowane po zainstalowaniu produktu WebSphere MQ 7.0.1, wówczas niestandardowy kanał produktu WebSphere MQ musi zostać aktywowany przez uruchomienie skryptu *WMQInstallDir*\bin\amqiRegisterdotNet.cmd , gdzie *WMQInstallDir* to katalog, w którym zainstalowano produkt WebSphere MQ 7.0.1 . Ten skrypt zainstaluje wymagane zespoły w globalnej pamięci podręcznej zespołu (Global Assembly Cache-GAC). W katalogu %TEMP% tworzony jest zestaw plików produktu amqi*.log , które rejestrują działania. Nie jest konieczne ponowne uruchomienie skryptu amqiRegisterdotNet.cmd , jeśli platforma .NET została zaktualizowana do wersji v3.0 lub nowszej z wcześniejszej wersji, na przykład z platformy .NET v2.0.

Kanał niestandardowy produktu WebSphere MQ dla produktu WCF: What's installed?

Kanał niestandardowy dla produktu WebSphere MQ jest kanałem transportowym korzystaniem z ujednoliconego modelu programowania WCF (Microsoft Windows Communication Foundation-WCF). Kanał niestandardowy jest instalowany domyślnie jako część instalacji produktu WebSphere MQ 7.0.1 .

Kanał niestandardowy produktu WebSphere MQ dla produktu WCF

Niestandardowy kanał produktu WebSphere MQ dla produktu WCF jest instalowany domyślnie jako część instalacji produktu WebSphere MQ 7.0.1 . Kanał niestandardowy i jego zależności są zawarte w komponencie Java and .NET Messaging and Web Services , który jest instalowany domyślnie. W przypadku aktualizacji do wersji WebSphere MQ 7.0.1 z wcześniejszej wersji aktualizacja kanału niestandardowego produktu WebSphere MQ jest instalowana domyślnie dla WCF, jeśli komponent Java and .NET Messaging and Web Services był wcześniej zainstalowany we wcześniejszej instalacji.

Komponent Java and .NET Messaging and Web Services zawiera plik IBM.XMS.WCF.dll , a plik IBM.XMS.WCF.dll jest głównym niestandardowym zespołem kanałów, który zawiera klasy interfejsu WCF. Ten plik jest instalowany w globalnej pamięci podręcznej zespołu (Global Assembly Cache-GAC) i jest również dostępny w następującym katalogu: *MQ_INSTALLATION_PATH*\bin , gdzie *MQ_INSTALLATION_PATH* jest katalogiem, w którym zainstalowano produkt WebSphere MQ 7.0.1 .

Klasy kluczy wymagane do korzystania z kanału niestandardowego znajdują się w obszarze *Przestrzeń nazw: IBM.XMS.WCF* oraz:

Nazwa powiązania transportu	IBM.XMS.WCF.SoapJmsIbmTransportBindingElement
Importer powiązania transportu	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementImporter
Konfiguracja powiązania transportu	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementConfig

Niestandardowe przykłady kanałów produktu WebSphere MQ

Przykłady zawierają kilka prostych przykładów użycia niestandardowego kanału produktu WebSphere MQ dla WCF. Przykłady i powiązane z nimi pliki znajdują się w katalogu *MQ_INSTALLATION_PATH*\tools\wcf\samples\ , gdzie *MQ_INSTALLATION_PATH* to katalog instalacyjny produktu WebSphere MQ. Więcej informacji na temat przykładów kanału niestandardowego produktu WebSphere MQ zawiera sekcja “Korzystanie z przykładów WCF” na stronie 631 .

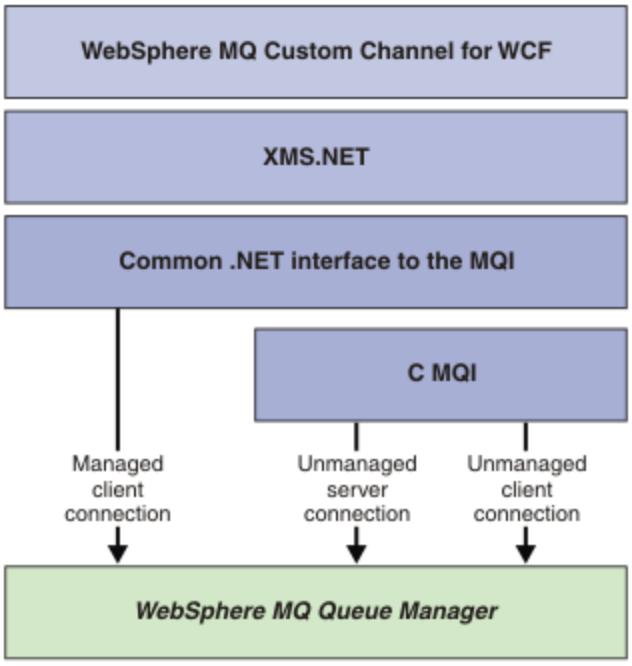
svcutil.exe.config

`svcutil.exe.config` jest przykładem ustawień konfiguracyjnych wymaganych do włączenia narzędzia do generowania proxy klienta Microsoft WCF `svcutil` w celu rozpoznania kanału niestandardowego. Plik `svcutil.exe.config` znajduje się w katalogu *MQ_INSTALLATION_PATH*`\tools\wcf\docs\examples\` , gdzie *MQ_INSTALLATION_PATH* jest katalogiem instalacyjnym produktu WebSphere MQ. Więcej informacji na temat korzystania z `svcutil.exe.config`zawiera sekcja: “Generowanie proxy klienta WCF i plików konfiguracyjnych aplikacji przy użyciu narzędzia svcutil z metadanymi pochodzącym z uruchomionej usługi” na stronie 628.

Architektura WCF

Niestandardowy kanał produktu WebSphere MQ dla WCF jest zintegrowany z interfejsem API produktu `IBM Message Service Client for .NET (XMS .NET)`.

Architektura WCF jest przedstawiona na poniższym diagramie:



Wszystkie wymagane komponenty są instalowane domyślnie przy użyciu instalacji produktu WebSphere MQ V7.0.1 .

Są to następujące trzy połączenia: Połączenia klienta zarządzanego, Połączenia z serwerem niezarządzanym i Połączenia klienta niezarządzanego. Więcej informacji na temat tych połączeń zawiera sekcja “Opcje połączenia WCF” na stronie 618.

Korzystanie z niestandardowych kanałów produktu WebSphere MQ dla produktu WCF

Przegląd informacji dostępnych dla programistów korzystających z niestandardowych kanałów WebSphere MQ V7 for Windows Communication Foundation (WCF).

Program Microsoft Windows Communication Foundation stanowi podstawę do obsługi usług Web Service i obsługi przesyłania komunikatów w środowisku Microsoft .NET Framework 3. Produkt WebSphere MQ V7 może być teraz używany jako kanał niestandardowy w środowisku WCF w środowisku .NET Framework 3 w taki sam sposób, jak wbudowane kanały oferowane przez firmę Microsoft.

Komunikaty transportowane w kanale niestandardowym są formatowane zgodnie z implementacją protokołu SOAP-JMS w produkcie WebSphere MQ V7. Aplikacje mogą następnie komunikować się z usługami udostępnianą przez system WCF lub za pomocą infrastruktury usługi WebSphere SOAP

przez JMS. Szczegółowe informacje na temat protokołu SOAP-JMS zawiera sekcja: “Transport produktu WebSphere MQ dla protokołu SOAP” na stronie 973

Niestandardowe funkcje i możliwości kanału WCF

Poniższe tematy zawierają informacje na temat niestandardowych funkcji i możliwości kanału WCF.

Niestandardowe kształty kanałów WCF

Przegląd niestandardowych kształtów kanałów, które mogą być używane w produkcie WebSphere MQ , jak w kanałach niestandardowych Microsoft Windows Communication Foundation (WCF).

Kanał niestandardowy produktu WebSphere MQ dla WCF obsługuje dwa kształty kanałów:

- Jednokierunkowa
- Żądanie-odpowiedź

WCF automatycznie wybiera kształt kanału zgodnie z udostępnianym kontraktem serwisowym.

Kontrakty zawierające metody, które używają tylko parametru **IsOneWay** , są obsługiwane przez jednokierunkowy kształt kanału, na przykład:

```
    [OperationContract(IsOneWay = true)]
    void printString(String text);
```

Kontrakty zawierające mieszaninę metod jednokierunkowych i żądanie-odpowiedź lub wszystkie metody żądanie-odpowiedź są obsługiwane przez kształt kanału odpowiedzi na żądanie. Na przykład:

```
    [OperationContract]
    int subtract(int a, int b);

    [OperationContract(IsOneWay = true)]
    void printString(string text);
```

Uwaga: Podczas mieszania metod jednokierunkowych i żądanie-odpowiedź w tej samej umowie należy upewnić się, że zachowanie jest zgodne z przeznaczeniem, szczególnie w przypadku pracy w środowisku mieszanym, ponieważ metody jednokierunkowe oczekują na odebranie odpowiedzi o wartości NULL z usługi.

Kanał jednokierunkowy

W produkcie WebSphere MQ niestandardowy kanał niestandardowy dla WCF jest używany na przykład do wysyłania komunikatów z klienta WCF przy użyciu jednokierunkowego kształtu kanału. Kanał może wysyłać komunikaty tylko w jednym kierunku, na przykład z menedżera kolejek klienta do kolejki w usłudze WCF.

Kanał żądanie-odpowiedź

Niestandardowy kanał WebSphere MQ żądanie-odpowiedź dla WCF jest używany na przykład w celu asynchronicznego wysyłania komunikatów w dwóch kierunkach; ta sama instancja klienta musi być używana na potrzeby asynchronicznego przesyłania komunikatów. Kanał może wysyłać komunikaty w jednym kierunku, na przykład z menedżera kolejek klienta do kolejki w usłudze WCF, a następnie wysłać komunikat odpowiedzi z WCF do kolejki w menedżerze kolejek klienta.

Nazwy i wartości parametrów URI WCF

connectionFactory
Parametr connectionFactory jest wymagany. Informacje na temat składni tego parametru zawiera sekcja Składnia i parametry URI dla wdrożenia usługi Web Service .

InitialContextFactory

Parametr initialContextFactory jest wymagany i musi być ustawiony na wartość "com.ibm.mq.jms.Nojndi" w celu zapewnienia zgodności z serwerem WebSphere Application Server i innymi produktami (patrz sekcja “Wdrażanie usługi na serwerze WebSphere Application Server w celu używania produktu WebSphere Transport for SOAP” na stronie 1034).

Zapewniony kanał niestandardowy WCF

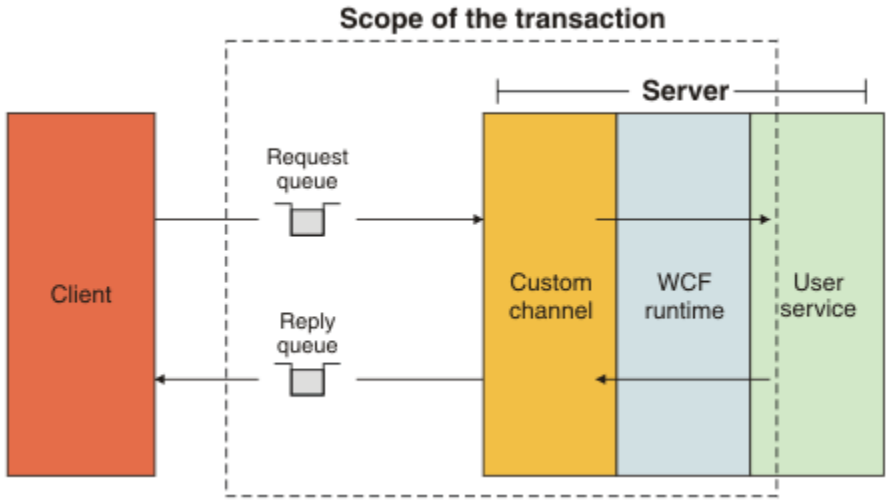
Zapewnione dostarczanie gwarantuje, że żądanie usługi lub odpowiedź są przydzielane i nie są tracone.

Odebrano komunikat z żądaniem, a każdy komunikat odpowiedzi jest wysyłany w lokalnym punkcie synchronizacji transakcji, który może zostać wycofany w przypadku niepowodzenia w czasie wykonywania. Przykłady tych awarii to: nieobsługiwany wyjątek zgłoszony przez usługę, niepowodzenie wysłania komunikatu do usługi lub niepowodzenie dostarczenia komunikatu odpowiedzi.

`AssuredDelivery` to zapewniony atrybut dostarczania, który można określić w umowie o świadczenie usług, aby zagwarantować, że wszystkie komunikaty żądania odebrane przez usługę oraz wszystkie komunikaty odpowiedzi wysłane z usługi nie zostaną utracone w przypadku niepowodzenia w czasie wykonywania.

Aby komunikaty były zachowywane również w przypadku awarii systemu lub wyłączenia zasilania, komunikaty muszą być wysyłane jako trwałe. Aby można było używać komunikatów trwałych, aplikacja kliencka musi mieć tę opcję określoną w identyfikatorze URI punktu końcowego. Więcej informacji na temat ustawiania właściwości identyfikatora URI zawiera sekcja Składnia i parametry identyfikatora URI dla wdrożenia usługi Web Service.

Rozproszone transakcje nie są obsługiwane, a zasięg transakcji nie wykracza poza przetwarzanie komunikatów żądania i odpowiedzi wykonywane przez produkt WebSphere MQ. Każda praca wykonana w ramach usługi może zostać ponownie uruchomiona w wyniku awarii, która spowoduje ponowne odebraną wiadomość. Na poniższym diagramie przedstawiono zakres transakcji:



Zapewnione dostarczanie jest włączone przez zastosowanie atrybutu `AssuredDelivery` do klasy usługi, jak pokazano w poniższym przykładzie:

```
[AssuredDelivery]
class TestCalculatorService : IWMQSampleCalculatorContract
{
    public int add(int a, int b)
    {
        int ans = a + b;
        return ans;
    }
}
```

Jeśli używany jest atrybut `AssuredDelivery` , należy pamiętać o następujących punktach:

- Jeśli kanał wykryje, że niepowodzenie może się powtórzyć, jeśli komunikat został wycofany i odebrany ponownie, komunikat jest traktowany jako komunikat nieprzetwarzalny i nie jest zwracany do kolejki

żądań w celu ponownego przetworzenia. Na przykład: jeśli odebrany komunikat nie jest poprawnie sformatowany lub nie może zostać wysłany do usługi. Nieobsłużone wyjątki zgłaszane przez operację usługi są zawsze powtarzane aż do momentu, gdy komunikat zostanie ponownie dostarczony przez maksymalną liczbę razy określoną przez właściwość progu wycofania kolejki żądań. Więcej informacji na ten temat zawiera sekcja “Niestandardowe komunikaty nieprzetwarzalne kanału WCF” na stronie 616 .

- Kanał wykonuje odczytywanie, przetwarzanie i odpowiadanie na każdy komunikat żądania jako operację atomową przy użyciu pojedynczego wątku wykonania w celu wymuszenia integralności transakcyjnej. Aby umożliwić współbieżne uruchamianie operacji serwisowych, kanał WCF umożliwia tworzenie wielu instancji kanału. Liczba instancji kanału dostępnych do przetwarzania żądań jest sterowana przez właściwość powiązania `MaxConcurrentCalls`. Więcej informacji na ten temat zawiera sekcja: “Opcje konfiguracji powiązania WCF” na stronie 624
- Zapewniona funkcja dostarczania korzysta zarówno z punktów rozszerzalności IOperationInvoker , jak i IErrorHandler WCF. Jeśli te punkty rozszerzalności są używane zewnętrznie przez aplikację, aplikacja musi upewnić się, że wszystkie zarejestrowane wcześniej punkty rozszerzalności są wywoływane. Niezastosowanie się do procedury IErrorHandler może spowodować, że błędy nie zostaną zgłoszone. Niepowodzenie operacji IOperationInvoker może spowodować, że WCF przestanie odpowiadać.

Niestandardowe zabezpieczenia kanału WCF

Niestandardowy kanał produktu WebSphere MQ dla WCF obsługuje użycie protokołu SSL tylko dla niezarządzanych połączeń klientów z menedżerem kolejek.

Protokół SSL można określić na jeden z dwóch sposobów:

- Określ protokół SSL bezpośrednio w identyfikatorze URI protokołu SOAP over JMS. Pełny opis opcji SSL można znaleźć w sekcji Transport SSL i transport WebSphere MQ dla protokołu SOAP .
- Określ protokół SSL przy użyciu pozycji w tabeli definicji kanału klienta (CCDT). Więcej informacji na temat tabel CCDTs zawiera sekcja Tabela definicji kanału klienta .

Tabele definicji kanału klienta WCF (CCDT)

Kanał niestandardowy produktu WebSphere MQ dla produktu WCF obsługuje użycie tabel definicji kanałów klienta (CCDT) w celu skonfigurowania informacji o połączeniu dla połączeń klienckich.

CCDTs są kontrolowane przez te dwie zmienne środowiskowe:

- *MQCHLLIB* określa katalog, w którym znajduje się tabela.
- *MQCHLTAB* określa nazwę pliku tabeli.

Nie można określić tabeli definicji kanału bezpośrednio w identyfikatorze URI protokołu SOAP over JMS. Jeśli te zmienne środowiskowe są zdefiniowane, mają one pierwszeństwo przed wszelkimi szczegółami połączenia klienta określonymi w identyfikatorze URI.

Więcej informacji na temat tabel definicji kanału klienta zawiera sekcja Tabela definicji kanału klienta .

Pojęcia pokrewne

Tabela definicji kanału klienta

Niestandardowe komunikaty nieprzetwarzalne kanału WCF

Gdy usługa nie może przetworzyć komunikatu żądania lub nie dostarczy komunikatu odpowiedzi do kolejki odpowiedzi, komunikat jest traktowany jako komunikat trucizny.

Komunikaty żądania trucizny

Jeśli komunikat żądania nie może zostać przetworzony, jest traktowany jako komunikat nieprzetwarzalny. To działanie uniemożliwia usłudze ponowne odbieranie tego samego komunikatu nieprzetwarzalnego. W przypadku nieprzetwarzalnego komunikatu żądania, który ma być traktowany jako komunikat nieprzetwarzalny, jedna z następujących sytuacji musi być prawdziwa:

- Liczba wycofań komunikatów przekroczyła próg wycofania określony w kolejce żądań, który ma miejsce tylko wtedy, gdy dla usługi została określona zapewniona dostawa. Więcej informacji na temat

gwarantowanego dostarczania zawiera sekcja “Zapewniony kanał niestandardowy WCF” na stronie 615 .

- Komunikat nie został poprawnie sformatowany i nie mógł zostać zinterpretowany jako komunikat SOAP over JMS.

Komunikaty odpowiedzi trucizny

Jeśli usługa nie dostarczy komunikatu odpowiedzi do kolejki odpowiedzi, komunikat odpowiedzi jest traktowany jako komunikat nieprzetwarzalny. W przypadku komunikatów odpowiedzi to działanie umożliwia późniejsze odtworzenie komunikatów odpowiedzi w celu określenia problemu z pomocą.

Obsługa komunikatów trujących

Działanie podjęte dla komunikatu nieprzetwarzalnego zależy od konfiguracji menedżera kolejek oraz od wartości ustawionych w opcjach raportu komunikatu. W przypadku protokołu SOAP-JMS następujące opcje raportu są domyślnie ustawiane w komunikatach żądań i nie można ich konfigurować:

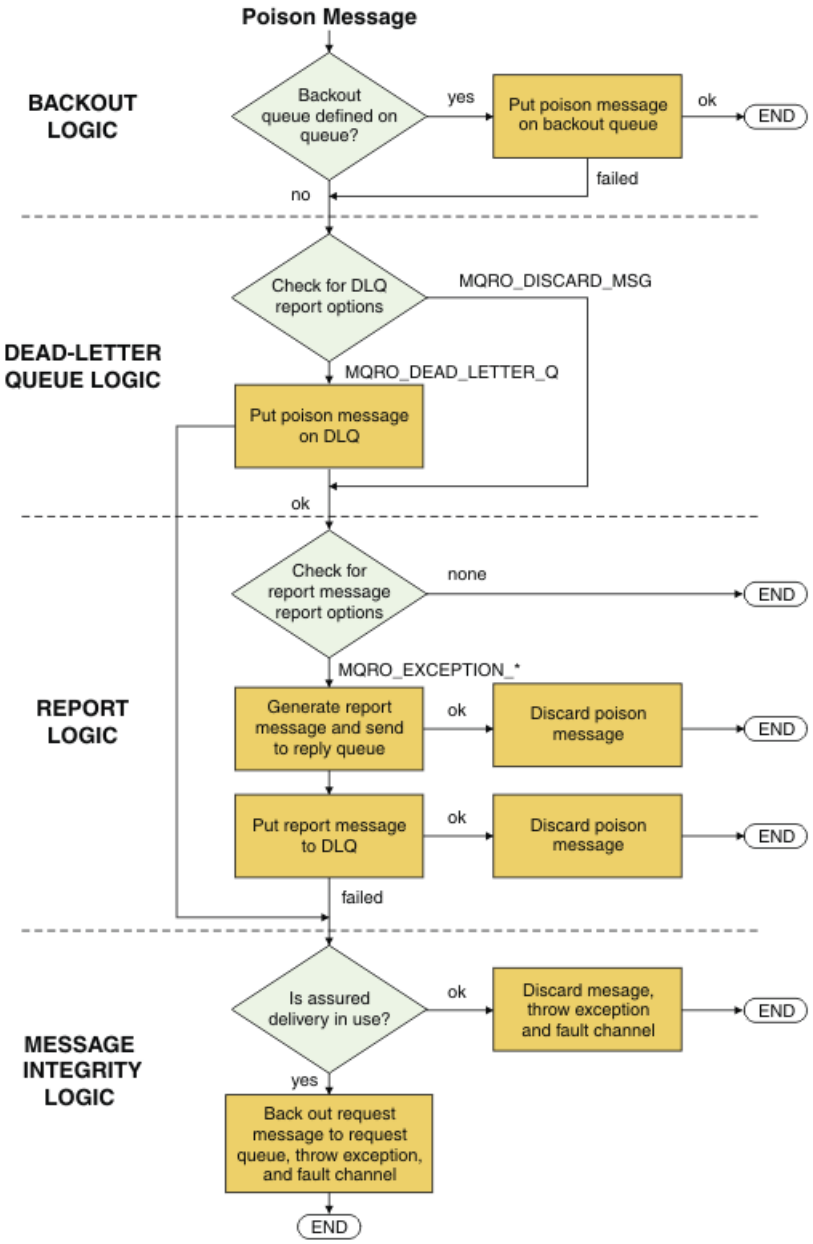
- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_DISCARD_MSG

W przypadku protokołu SOAP over JMS następujące opcje raportu są domyślnie ustawione na komunikaty odpowiedzi i nie można ich konfigurować:

- MQRO_DEAD_LETTER_Q

Jeśli komunikaty pochodzą ze źródła innego niż WCF, należy zapoznać się z dokumentacją dla tego źródła.

Na poniższym diagramie przedstawiono możliwe działania i kroki podjęte w przypadku niepowodzenia obsługi komunikatów nieprzetwarzalnych:



Opcje połączenia WCF

Istnieją trzy tryby łączenia kanału niestandardowego produktu WebSphere MQ dla WCF z menedżerem kolejek. Należy wziąć pod uwagę, który typ połączenia najlepiej odpowiada wymaganiom użytkownika.

Więcej informacji na temat opcji połączenia zawiera sekcja “Różnice między połączeniami” na stronie 589 .

Więcej informacji na temat architektury WCF można znaleźć pod adresem: “Architektura WCF” na stronie 613

Niezarządzane połączenie klienta

Połączenie wykonane w tym trybie łączy się jako klient WebSphere MQ z serwerem WebSphere MQ działającym na komputerze lokalnym lub na komputerze zdalnym.

Aby użyć niestandardowego kanału produktu WebSphere MQ dla WCF jako klienta WebSphere MQ , można go zainstalować przy użyciu klienta MQI produktu WebSphere MQ na serwerze WebSphere MQ lub na osobnym komputerze.

Połączenie z serwerem niezarządzanym

W przypadku użycia w trybie powiązań serwera kanał niestandardowy produktu WebSphere MQ dla WCF korzysta z interfejsu API menedżera kolejek, a nie komunikując się za pośrednictwem sieci. Korzystanie z połączeń powiązań zapewnia lepszą wydajność aplikacji WebSphere MQ niż korzystanie z połączeń sieciowych.

Aby korzystać z połączenia powiązań, należy zainstalować niestandardowy kanał produktu WebSphere MQ dla produktu WCF na serwerze WebSphere MQ .

Połączenie z klientem zarządzanym

Połączenie wykonane w tym trybie łączy się jako klient WebSphere MQ z serwerem WebSphere MQ działającym na komputerze lokalnym lub na komputerze zdalnym.

Niestandardowe klasy kanału WebSphere MQ dla środowiska .NET 3 łączące się w tym trybie pozostają w kodzie zarządzanym .NET i nie wywołują żadnych połączeń z usługami rodzimymi. Więcej informacji na temat kodu zarządzanego zawiera dokumentacja firmy Microsoft .

Istnieje wiele ograniczeń dotyczących korzystania z zarządzanego klienta. Więcej informacji na temat tych ograniczeń można znaleźć w sekcji “Połączenia zarządzane przez klienta” na stronie 589.

Tworzenie i konfigurowanie niestandardowego kanału produktu WebSphere MQ dla produktu WCF

Niestandardowe kanały WebSphere MQ V7 dla WCF działają w taki sam sposób, jak kanały transportowe WCF oferowane przez firmę Microsoft. Kanał niestandardowy produktu WebSphere MQ dla WCF można utworzyć na jeden z dwóch sposobów.

O tym zadaniu

Kanał niestandardowy produktu WebSphere MQ integruje się z systemem WCF jako kanał transportowy WCF i jako taki musi być sparowany z koderem komunikatów i opcjonalnymi kanałami protokołu, dzięki czemu może utworzyć kompletny stos kanałów, który może być używany przez aplikację. W celu pomyślnego utworzenia pełnego stosu kanału wymagane są dwa elementy:

1. Definicja powiązania: określa, które elementy są wymagane do zbudowania stosu kanału aplikacji, w tym kanału transportowego, kodera komunikatów i wszystkich protokołów, a także wszelkich ogólnych ustawień konfiguracyjnych. W przypadku kanału niestandardowego definicja powiązania musi zostać utworzona w postaci powiązania niestandardowego WCF.
2. Definicja punktu końcowego: łączy umowę o świadczenie usług z definicją powiązania, a także udostępnia rzeczywisty identyfikator URI połączenia, który opisuje miejsce, w którym aplikacja może nawiązać połączenie. W przypadku kanału niestandardowego identyfikator URI jest w postaci identyfikatora URI protokołu SOAP over JMS.

Definicje te można utworzyć na jeden z dwóch sposobów:

- Administracyjnie; definicje są tworzone przez podanie szczegółów w pliku konfiguracyjnym aplikacji (na przykład: `app.config`).
- Programowo; definicje są tworzone bezpośrednio z poziomu kodu aplikacji.

Decyzja, nad którą metoda powinna zostać użyta do utworzenia definicji, musi być oparta na wymaganiach aplikacji w następujący sposób:

- Metoda administracyjna konfiguracji zapewnia elastyczność w zakresie zmiany szczegółów usługi i klienta po wdrożeniu bez odbudowowania aplikacji.
- Programowa metoda konfiguracji zapewnia większą ochronę przed błędami konfiguracji oraz możliwość dynamicznego generowania konfiguracji w czasie wykonywania.

Tworzenie niestandardowego kanału WCF administracyjnego przez dostarczanie informacji o powiązaniach i punktach końcowych w pliku konfiguracyjnym aplikacji

Kanał niestandardowy produktu WebSphere MQ dla WCF jest kanałem WCF na poziomie transportu. Aby można było używać kanału niestandardowego, należy zdefiniować punkt końcowy i powiązanie. Definicje te można wykonać, podając informacje o powiązaniach i punktach końcowych w pliku konfiguracyjnym aplikacji.

Aby skonfigurować i używać niestandardowego kanału produktu WebSphere MQ dla systemu WCF, który jest kanałem WCF na poziomie transportu, należy zdefiniować powiązanie i definicję punktu końcowego. Powiązanie przechowuje informacje konfiguracyjne dla kanału, a definicja punktu końcowego zawiera szczegóły połączenia. Definicje te mogą być tworzone na dwa sposoby:

- Programowo bezpośrednio z kodu aplikacji, zgodnie z opisem w tym miejscu: "Tworzenie kanału niestandardowego WCF przez programowe informacje o powiązaniach i punktach końcowych" na stronie 622
- Administracyjnie, podając szczegółowe informacje w pliku konfiguracyjnym aplikacji, zgodnie z opisem w poniższej procedurze.

Plik konfiguracyjny klienta lub aplikacji usługi nosi nazwę *yourappname*`.exe.config` , gdzie *nazwa_aplikacje_użytkownika* jest nazwą aplikacji. Plik konfiguracyjny aplikacji jest najłatwiej modyfikowany za pomocą narzędzia Microsoft Service Configuration Tool o nazwie `SvcConfigEditor.exe` w następujący sposób:

- Uruchom narzędzie edytora konfiguracji produktu `SvcConfigEditor.exe` . Domyślne miejsce instalacji dla tego narzędzia to: *`Drive:`*`\Program Files\Microsoft SDKs\Windows\v6.0\Bin\SvcConfigEditor.exe` , gdzie *Napęd:* jest nazwą napędu instalacyjnego.

Krok 1: Dodawanie rozszerzenia elementu binding w celu włączenia WCF w celu znalezienia kanału niestandardowego

1. Kliknij prawym przyciskiem myszy opcję **Zaawansowane** > **Rozszerzenie** > **element powiązania** , aby otworzyć menu, a następnie wybierz opcję **Nowy** .
2. Wypełnij pola zgodnie z poniższą tabelą:

Tabela 73. Nowe pola elementu powiązania

Pole	**Wartość**
Nazwa	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`
Typ	Przejdź do `IBM.XMS.WCF.dll` w globalnej pamięci podręcznej zespołu (Global Assembly Cache-GAC) i wybierz `IBM.XMS.WCFSoapJmsIbmTransportBindingElementConfig`

Krok 2: tworzenie niestandardowej definicji powiązania, która paruje kanał niestandardowy z koderem komunikatów WCF

1. Kliknij prawym przyciskiem myszy opcję **Powiązania** , aby otworzyć menu, a następnie wybierz opcję **Nowa konfiguracja powiązania** .
2. Wypełnij pola zgodnie z poniższą tabelą:

Tabela 74. Nowe pola konfiguracji powiązania

Pole	**Wartość**
Nazwa	`CustomBinding_WMQ`

Tabela 74. Nowe pola konfiguracji powiązania (kontynuacja)	
Pole	**Wartość**
BindingElement 1	`textMessageEncoding (MessageVersion: Soap11)`
BindingElement 2	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`

Krok 3: Określanie właściwości powiązania

1. Wybierz opcję *IBM.XMS.WCF.SoapJmsIbmTransportChannel* Powiązanie transportu z powiązania, które zostało utworzone w: "Krok 2: tworzenie niestandardowej definicji powiązania, która paruje kanał niestandardowy z koderem komunikatów WCF" na stronie 620
2. Wprowadź wszystkie wymagane zmiany w wartościach domyślnych właściwości zgodnie z opisem w: "Opcje konfiguracji powiązania WCF" na stronie 624

Krok 4: Tworzenie definicji punktu końcowego

Utwórz definicję punktu końcowego, która odwołuje się do powiązania niestandardowego, które zostało utworzone w: "Krok 2: tworzenie niestandardowej definicji powiązania, która paruje kanał niestandardowy z koderem komunikatów WCF" na stronie 620 , i udostępnia szczegóły połączenia usługi. Sposób określania tych informacji jest zależny od tego, czy definicja dotyczy aplikacji klienckiej, czy aplikacji usługowej.

W przypadku aplikacji klienckiej dodaj definicję punktu końcowego do sekcji klienta w następujący sposób:

1. Kliknij prawym przyciskiem myszy opcję **Klient** > **Punkty końcowe** , aby otworzyć menu, a następnie wybierz opcję **Nowy punkt końcowy klienta** .
2. Wypełnij pola zgodnie z poniższą tabelą:

Tabela 75. Pola nowego punktu końcowego klienta	
Pole	**Wartość**
Nazwa	`Endpoint_WMQ`
Adres	*Identyfikator URI SOAP/JMS opisujący szczegóły połączenia WMQ wymagane w celu uzyskania dostępu do usługi.* Więcej informacji na ten temat zawiera sekcja "Kanał niestandardowy produktu WebSphere MQ dla formatu adresu URI punktu końcowego WCF" na stronie 623 .
Łączy	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Contract (kontrakt)	*Nazwa interfejsu umowy usługi*

W przypadku aplikacji usługi należy dodać definicję usługi do sekcji usług w następujący sposób:

1. Kliknij prawym przyciskiem myszy opcję **Usługi** , aby otworzyć menu, a następnie wybierz opcję **Nowa usługa**, a następnie wybierz klasę usługi, która ma być udostępniana.
2. Dodaj definicję punktu końcowego do sekcji **Punkty końcowe** dla nowej usługi i wypełniaj pola zgodnie z poniższą tabelą:

Tabela 76. Nowe pola punktu końcowego usługi	
Pole	**Wartość**
Nazwa	`Endpoint_WMQ`

Tabela 76. Nowe pola punktu końcowego usługi (kontynuacja)	
Pole	**Wartość**
Adres	*Identyfikator URI SOAP/JMS opisujący szczegóły połączenia WMQ wymagane w celu uzyskania dostępu do usługi.* Więcej informacji na ten temat zawiera sekcja “Kanał niestandardowy produktu WebSphere MQ dla formatu adresu URI punktu końcowego WCF” na stronie 623 .
Łączy	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Contract (kontrakt)	*Nazwa klasy implementacji usługi*

Tworzenie kanału niestandardowego WCF przez programowe informacje o powiązaniach i punktach końcowych

Kanał niestandardowy produktu WebSphere MQ dla WCF jest kanałem WCF na poziomie transportu. Aby można było używać kanału niestandardowego, należy zdefiniować punkt końcowy i powiązanie, a te definicje można wykonać programowo bezpośrednio z poziomu kodu aplikacji.

Aby skonfigurować i używać niestandardowego kanału produktu WebSphere MQ dla systemu WCF, który jest kanałem WCF na poziomie transportu, należy zdefiniować powiązanie i definicję punktu końcowego. Powiązanie przechowuje informacje konfiguracyjne dla kanału, a definicja punktu końcowego zawiera szczegóły połączenia. Więcej informacji na ten temat zawiera sekcja “Korzystanie z przykładów WCF” na stronie 631 .

Definicje te mogą być tworzone na dwa sposoby:

- Administracyjnie, podając szczegółowe informacje w pliku konfiguracyjnym aplikacji, zgodnie z opisem w następującym miejscu: “Tworzenie niestandardowego kanału WCF administracyjnego przez dostarczanie informacji o powiązaniach i punktach końcowych w pliku konfiguracyjnym aplikacji” na stronie 620
- Programowo bezpośrednio z kodu aplikacji, zgodnie z opisem podanym w poniższym przykładzie.

Krok 1: tworzenie instancji elementu powiązania transportu dla kanału

Dodaj do aplikacji następujący kod:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
```

Krok 2: Ustawianie właściwości powiązania

Ustaw wszystkie wymagane właściwości powiązania, na przykład poprzez dodanie do aplikacji następującego kodu w celu ustawienia `ClientConnectionMode`.

```
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.AS_URI;
```

Krok 3: tworzenie niestandardowego powiązania, które służy do parowania kanału transportowego za pomocą programu kodującego komunikaty

Aby utworzyć powiązanie niestandardowe, należy dodać następujący kod do aplikacji:

```
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
```

Krok 4: Tworzenie identyfikatora URI SOAP/JMS

Identyfikator URI SOAP/JMS, który opisuje szczegóły połączenia WebSphere MQ wymagane do uzyskania dostępu do usługi, musi być podany jako adres punktu końcowego. Zależy to od tego, czy kanał jest używany dla aplikacji usługowej, czy aplikacji klienckiej.

W przypadku aplikacji klienckich identyfikator URI SOAP/JMS musi zostać utworzony jako `EndpointAddress` w następujący sposób:

```
EndpointAddress address = new EndpointAddress("jms:/queue?
destination=SampleQ@QM1&connectionFactory=connectQueueManager(QM1)&initialContextFactory=com.ibm
.mq.jms.Nojndi");
```

W przypadku aplikacji usługowych identyfikator URI SOAP/JMS musi zostać utworzony jako identyfikator URI w następujący sposób:

```
Uri address = new Uri("jms:/queue?
destination=SampleQ@QM1&connectionFactory=connectQueueManager(QM1)&initialContextFactory=com.ibm
.mq.jms.Nojndi");
```

Więcej informacji na temat adresu punktu końcowego zawiera sekcja “Kanał niestandardowy produktu WebSphere MQ dla formatu adresu URI punktu końcowego WCF” na stronie 623 .

Kanał niestandardowy produktu WebSphere MQ dla formatu adresu URI punktu końcowego WCF

Identyfikator URI (Universal Resource Identifier) udostępnia położenie i szczegóły połączenia w celu określenia usługi Web Service. Ten format identyfikatora URI pozwala na uzyskanie kompleksowego poziomu kontroli nad parametrami i opcjami specyficznymi dla protokołu SOAP/ WebSphere MQpodczas uzyskiwania dostępu do usług docelowych.

Usługa Web Service jest określana przy użyciu identyfikatora URI (Universal Resource Identifier). Ta sekcja określa format identyfikatora URI, który jest obsługiwany w transporcie WebSphere MQ dla protokołu SOAP. Ten format identyfikatora URI pozwala na uzyskanie kompleksowego poziomu kontroli nad parametrami i opcjami specyficznymi dla protokołu SOAP/WebSphere MQpodczas uzyskiwania dostępu do usług docelowych. Ten format jest zgodny z serwerem WebSphere Application Server (WAS) i z produktem CICS ułatwiającym integrację produktu WebSphere MQ z obydwoma tymi produktami.

Składnia identyfikatora URI jest następująca:

```
jms:/queue?name=value&name=value...
```

gdzie `name` jest nazwą parametru, a *wartość* jest odpowiednią wartością, a element `name`=*wartość* może być powtórzony dowolną liczbę razy z drugim, a kolejne wystąpienia poprzedzane ampersand (&).

Więcej informacji na temat ustawiania właściwości identyfikatora URI zawiera sekcja Składnia i parametry URI dla wdrożenia usługi Web Service .

W nazwach parametrów rozróżniana jest wielkość liter, ponieważ są to nazwy obiektów WebSphere MQ . Jeśli dowolny parametr zostanie określony więcej niż jeden raz, końcowe wystąpienie parametru będzie miało wpływ na to, że aplikacje klienckie mogą przesłonić wartości parametrów, dołączając do identyfikatora URI. Jeśli zostaną uwzględnione dodatkowe nierozpoznane parametry, zostaną one zignorowane.

Jeśli identyfikator URI jest przechowywany w łańcuchu XML, należy reprezentować znak ampersand jako "&". Podobnie, jeśli identyfikator URI jest zakodowany w skrypcie, należy zwrócić uwagę na znaki zmiany znaczenia, takie jak **&** , które w przeciwnym razie zostałyby zinterpretowane przez powłokę.

Poniżej przedstawiono przykład prostego identyfikatora URI dla usługi Axis:

```
jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Poniżej znajduje się przykład prostego identyfikatora URI dla usługi .NET:

```
jms:/queue?destination=myQ&connectionFactory=()&targetService=MyService.asmx
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Dostarczane są tylko wymagane parametry (`targetService` jest wymagany tylko w przypadku usług .NET), a `connectionFactory` nie ma żadnych opcji.

W tym przykładzie osi `connectionFactory` zawiera wiele opcji:

```
jms:/queue?destination=myQ@myRQM&connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

W tym przykładzie przedstawiono również opcję `sslPeerName` produktu `connectionFactory` . Wartość parametru sslPeerNazwa zawiera pary nazwa-wartość i znaczące odstępy wewnętrzne:

```
jms:/queue?destination=myQ@myRQM&connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
sslPeerName(CN=MQ Test 1,O=IBM,S=Hampshire,C=GB)
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Opcje konfiguracji powiązania WCF

W tym temacie opisano, w jaki sposób można zastosować opcje konfiguracyjne dla informacji o powiązaniu kanałów niestandardowych, a także listę dostępnych opcji.

Opcje konfiguracji powiązania można ustawić na jeden z dwóch różnych sposobów:

1. Administracyjnie: ustawienia właściwości powiązania muszą być określone w sekcji transportu definicji powiązania niestandardowego w pliku konfiguracyjnym aplikacji, na przykład: `app.config`
2. Programowo: kod aplikacji musi zostać zmodyfikowany, aby można było określić właściwość podczas inicjowania powiązania niestandardowego.

Ustawianie właściwości powiązania administracyjnie

Ustawienia właściwości powiązania można również określić w pliku konfiguracyjnym aplikacji, na przykład: `app.config`. Plik konfiguracyjny jest generowany przez program **svcutil**, na przykład:

```
<customBinding>

...

     <IBM.XMS.WCF.SoapJmsIbmTransportChannel maxBufferPoolSize="524288"
      maxMessageSize="4000000" clientConnectionMode="0" maxConcurrentCalls="16"/>

...

</customBinding>
```

Programowo ustawianie właściwości powiązania

Aby dodać właściwość powiązania WCF w celu określenia trybu połączenia klienckiego, należy zmodyfikować kod usługi, aby określić właściwość podczas inicjowania powiązania niestandardowego.

Użyj następującego przykładu, aby określić tryb niezarządzanego połączenia klienckiego:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.CLIENT_UNMANAGED;

Binding sampleBinding = new CustomBinding(new TextMessageEncodingBindingElement(),
                                                    transportBindingElement);
```

Właściwości powiązania WCF

Nazwa właściwości	Aplikacja kliencka lub usługowa	Wartość administracyjna	Wartość programowa	Opis
maxBufferPoolSize	Obie	Od 0 do 64 bitowej liczby całkowitej	Od 0 do 64 bitowej liczby całkowitej	Określa maksymalną wielkość pamięci, która może być używana do przechowywania buforów komunikatów WCF dla instancji kanału.
maxMessage, Wielkość	Obie	Od 1 do 32-bitowej liczby całkowitej ze znakiem	Od 1 do 32-bitowej liczby całkowitej ze znakiem	Określa maksymalną ilość pamięci, która może być użyta dla pojedynczego komunikatu WCF.
Tryb clientConnection	Obie	0 (wartość domyślna) 1	AS_URI (wartość domyślna) KLIENT_UNMANAGED	Określa tryb połączenia klienta w kanale transportowym. Wartość 0 oznacza, że tryb połączenia klienta jest określony w identyfikatorze URI. Używany tylko wtedy, gdy używane jest połączenie klienta. Określa, że tryb połączenia klienta jest określony w identyfikatorze URI. Wartość 0 jest wartością domyślną, jeśli nie jest ustawiony żaden tryb połączenia klienckiego. 1 oznacza, że tryb połączenia klienta jest klientem niezarządzanym. Używany tylko wtedy, gdy używane jest połączenie klienta.

Nazwa właściwości	Aplikacja kliencka lub usługowa	Wartość administracyjna	Wartość programowa	Opis
Wywołania MaxConcurrent	Klient	Zakres wartości to 0-2 147 483 647 16 to wartość domyślna	Zakres wartości to 0-2 147 483 647 16 to wartość domyślna	Ta właściwość definiuje maksymalną liczbę współbieżnych operacji, które mogą być wykonywane na pojedynczym serwerze proxy klienta w dowolnym momencie. W przypadku uruchomienia większej liczby operacji są one umieszczane w kolejce do momentu zakończenia lub zakończenia operacji w toku. To ustawienie może być używane do sterowania maksymalną liczbę wątków i zasobów, które mogą być wykorzystywane przez pojedynczego proxy. `0` usuwa ten limit, umożliwiając jednoczesną próbę wykonania wszystkich operacji.
Wywołania MaxConcurrent	Usługa	Zakres ten wynosi 1-2 147 483 647 16 to wartość domyślna	Zakres ten wynosi 1-2 147 483 647 16 to wartość domyślna	Ta właściwość jest używana tylko wtedy, gdy funkcja gwarantowanego dostarczania jest włączona (więcej informacji na temat gwarantowanego dostarczania zawiera sekcja “Zapewniony kanał niestandardowy WCF” na stronie 615). Określa on maksymalną liczbę współbieżnych operacji, które mogą być jednocześnie w toku dla danego punktu końcowego. Podczas zmiany tego ustawienia należy zachować ostrożność. Każda operacja współbieżna wymaga dodatkowych zasobów, w szczególności nowej instancji kanału niestandardowego i powiązanych wątków z puli wątków w celu działania żądań. Nadmierna alokacja może być bardzo wydajna i poważnie wpływa na wydajność. W celu obsługi tej właściwości należy utworzyć odpowiednią konfigurację puli wątków.

Usługi budowlane i usługi serwerowe dla WCF

Przegląd usług produktu Microsoft Windows Communication Foundation (WCF) wyjaśniających, w jaki sposób można tworzyć i konfigurować usługi WCF.

Kanał niestandardowy produktu IBM WebSphere MQ dla usług WCF i WCF, które korzystają z tego kanału, może być udostępniany za pomocą następujących metod:

- Self-hosting
- Usługa Windows

Niestandardowy kanał produktu IBM WebSphere MQ dla systemu WCF nie może być udostępniany w usłudze aktywacji procesów systemu Windows.

W poniższych tematach przedstawiono kilka prostych przykładów samodzielnego udostępniania, aby zademonstrować kroki związane z tym tematem. Dokumentacja elektroniczna produktu Microsoft WCF, zawierająca dalsze informacje oraz najnowsze informacje szczegółowe, znajduje się na stronie WWW MSDN produktu Microsoft pod adresem https://msdn.microsoft.com.

Budowanie aplikacji usług WCF przy użyciu metody 1: Samodzielne udostępnianie administracyjnie przy użyciu pliku konfiguracyjnego aplikacji

Po utworzeniu pliku konfiguracyjnego aplikacji otwórz instancję usługi i dodaj określony kod do aplikacji.

Zanim rozpoczniesz

Utwórz lub dokonaj edycji pliku konfiguracyjnego aplikacji dla usługi zgodnie z opisem w: "Tworzenie niestandardowego kanału WCF administracyjnego przez dostarczanie informacji o powiązaniach i punktach końcowych w pliku konfiguracyjnym aplikacji" na stronie 620

O tym zadaniu

1. Utwórz instancję i otwórz instancję usługi na hoście usługi. Typ usługi musi być taki sam, jak typ usługi określony w pliku konfiguracyjnym usługi.
2. Dodaj do aplikacji następujący kod:

```
ServiceHost service = new ServiceHost(typeof(MyService));
service.Open();
...
service.Close();
```

Budowanie aplikacji usługowych WCF przy użyciu metody 2: Samodzielność programowo bezpośrednio z aplikacji

Dodaj właściwości powiązania, utwórz host usługi z instancją wymaganej klasy usługi i otwórz usługę.

Zanim rozpoczniesz

1. Dodaj odwołanie do niestandardowego pliku kanału `IBM.XMS.WCF.dll` do projektu. `IBM.XMS.WCF.dll` znajduje się w katalogu *`WMQInstallDir`*`\bin` , gdzie *WMQInstallDir* jest katalogiem, w którym jest zainstalowany program WebSphere MQ 7.
2. Dodaj instrukcję *using* do przestrzeni nazw `IBM.XMS.WCF` , na przykład: `using IBM.XMS.WCF`
3. Utwórz instancję elementu powiązania kanałów i punktu końcowego zgodnie z opisem w: "Tworzenie kanału niestandardowego WCF przez programowe informacje o powiązaniach i punktach końcowych" na stronie 622

O tym zadaniu

Jeśli wymagane są zmiany właściwości powiązania kanału, wykonaj następujące kroki:

1. Dodaj właściwości powiązania do produktu `transportBindingElement` , jak pokazano w poniższym przykładzie:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
Uri address = new Uri("jms:/queue?destination=SampleQ@QM1&connectionFactory=

connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
```

2. Utwórz host usługi z instancją wymaganej klasy usługi:

```
ServiceHost service = new ServiceHost(typeof(MyService));
```

3. Otwórz usługę:

```
service.AddServiceEndpoint(typeof(IMyServiceContract), binding, address);
service.Open();
    ...
service.Close();
```

Ujawnianie metadanych przy użyciu punktu końcowego HTTP

Instrukcje dotyczące ujawniania metadanych usługi, która jest skonfigurowana do używania niestandardowego kanału WebSphere MQ dla WCF.

O tym zadaniu

Jeśli metadane usług muszą być ujawnione (aby narzędzia, takie jak produkt `svcutil` , mogły uzyskać do niego dostęp bezpośrednio z działającej usługi, a nie z pliku WSDL działającego w trybie bez połączenia), należy to zrobić, ujawniając metadane usług za pomocą punktu końcowego HTTP. Aby dodać ten dodatkowy punkt końcowy, można użyć następujących kroków.

1. Dodaj adres bazowy, w którym metadane muszą być ujawnione na hoście ServiceHost, na przykład:

```
ServiceHost service = new ServiceHost(typeof(TestService),
                      new Uri("http://localhost:8000/MyService"));
```

2. Dodaj następujący kod do obiektu ServiceHost przed otwarniem usługi:

```
ServiceMetadataBehavior metadataBehavior = new ServiceMetadataBehavior();
  metadataBehavior.HttpGetEnabled = true;
  service.Description.Behaviors.Add(metadataBehavior);
  service.AddServiceEndpoint(typeof(IMetadataExchange),
     MetadataExchangeBindings.CreateMexHttpBinding(), "mex");
```

Wyniki

Metadane są teraz dostępne pod następującym adresem: `http://localhost:8000/MyService`

Budowanie aplikacji klienckich dla WCF

Przegląd generowania i budowania aplikacji klienckich Microsoft Windows Communication Foundation (WCF).

Aplikacja kliencka może zostać utworzona dla usługi WCF. Aplikacje klienckie są zwykle generowane przy użyciu narzędzia Microsoft ServiceModel Metadata Utility Tool (Svcutil.exe) w celu utworzenia wymaganych plików konfiguracyjnych i proxy, które mogą być używane bezpośrednio przez aplikację.

Generowanie proxy klienta WCF i plików konfiguracyjnych aplikacji przy użyciu narzędzia svcutil z metadanymi pochodzącym z uruchomionej usługi

Instrukcje dotyczące używania narzędzia Microsoft svcutil.exe do generowania klienta dla usługi, która jest skonfigurowana do używania niestandardowego kanału WebSphere MQ dla WCF.

Zanim rozpoczniesz

Istnieją trzy wymagania wstępne dotyczące korzystania z narzędzia svcutil do tworzenia wymaganych plików konfiguracyjnych i proxy, które mogą być używane bezpośrednio przez aplikację:

- Usługa WCF musi być uruchomiona przed uruchomieniem narzędzia svcutil.
- Usługa WCF musi ujawniać swoje metadane przy użyciu portu HTTP, oprócz odwołań do punktów końcowych kanału niestandardowego produktu WebSphere MQ w celu wygenerowania klienta bezpośrednio z uruchomionej usługi.
- Kanał niestandardowy musi być zarejestrowany w danych konfiguracji dla usługi svcutil.

O tym zadaniu

W poniższych krokach wyjaśniono sposób generowania klienta dla usługi, która jest skonfigurowana do używania kanału niestandardowego produktu WebSphere MQ , ale również udostępnia swoje metadane w czasie wykonywania za pośrednictwem osobnego portu HTTP:

1. Uruchom usługę WCF (usługa musi być uruchomiona przed uruchomieniem narzędzia svcutil).
2. Dodaj szczegóły z pliku konfiguracyjnego produktu `svcutil.exe` z katalogu głównego instalacji do aktywnego pliku konfiguracyjnego svcutil, zwykle produkt `C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\svcutil.exe.config` tak, aby program svcutil rozpoznał kanał niestandardowy produktu WebSphere MQ .
3. Uruchom komendę svcutil z wiersza komend, na przykład:

```
svcutil /language:C# /r:<installlocation>\bin\IBM.XMS.WCF.dll
               /config:app.config http://localhost:8000/IBM.XMS.WCF/samples
```

4. Skopiuj wygenerowane pliki `app.config` i `YourService.cs` do projektu klienta Microsoft Visual Studio.

Co dalej

Jeśli nie można bezpośrednio pobrać metadanych usług, można użyć usługi svcutil w celu wygenerowania plików klienta z pliku wsdl. Więcej informacji na ten temat zawiera sekcja “Generowanie proxy klienta WCF i plików konfiguracyjnych aplikacji przy użyciu narzędzia svcutil przy użyciu pliku WSDL” na stronie 629 .

Generowanie proxy klienta WCF i plików konfiguracyjnych aplikacji przy użyciu narzędzia svcutil przy użyciu pliku WSDL

Instrukcje dotyczące generowania klientów WCF na podstawie pliku WSDL, jeśli metadane usługi są niedostępne.

Jeśli nie można bezpośrednio pobrać metadanych usługi w celu wygenerowania klienta na podstawie metadanych z działającej usługi, można użyć usługi svcutil w celu wygenerowania plików klienta z pliku WSDL. Aby określić, że kanał niestandardowy produktu WebSphere MQ ma być używany, należy wprowadzić następujące modyfikacje w pliku WSDL:

1. Dodaj następujące definicje przestrzeni nazw i informacje o strategii:

```
<wsdl:definitions
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">

        <wsp:Policy wsu:Id="CustomBinding_IWMQSampleContract_policy">
            <wsp:ExtactlyOne>
                <wsp:All>
                    <xms:xms xmlns:xms="http://sample.schemas.ibm.com/policy/xms" />
                </wsp:All>
            </wsp:ExactlyOne>
        </wsp:Policy>

...

</wsdl:definitions>
```

2. Zmodyfikuj sekcję powiązań w taki sposób, aby odwoływała się do nowej sekcji strategii, a następnie usuń dowolną definicję `transport` z bazowego elementu powiązania:

```
<wsdl:definitions ...>

        <wsdl:binding ...>
                <wsp:PolicyReference URI="#CustomerBinding_IWMQSampleContract_policy" />
                <[soap]:binding ... transport="" />
                ...
        </wsdl:binding>
</wsdl:definitions>
```

3. Uruchom komendę svcutil z wiersza komend, na przykład:

```
svcutil /language:C# /r:MQ_INSTALLATION_PATH\bin\IBM.XMS.WCF.dll
        /config:app.config
MQ_INSTALLATION_PATH\src\samples\WMQAxis\default\service\soap.server.stockQuoteAxis_Wmq.wsdl
```

Gdzie *MQ_INSTALLATION_PATH* to katalog instalacyjny produktu WebSphere MQ.

Budowanie aplikacji klienckich WCF przy użyciu proxy klienta z plikiem konfiguracyjnym aplikacji

Zanim rozpoczniesz

Utwórz lub edytuj plik konfiguracyjny aplikacji dla klienta zgodnie z opisem w: "Tworzenie niestandardowego kanału WCF administracyjnego przez dostarczanie informacji o powiązaniach i punktach końcowych w pliku konfiguracyjnym aplikacji" na stronie 620

O tym zadaniu

Utwórz instancję serwera proxy klienta i otwórz instancję serwera proxy klienta. Parametr przekazany do wygenerowanego serwera proxy musi być taki sam, jak nazwa punktu końcowego określona w pliku konfiguracyjnym klienta, na przykład `Endpoint_WMQ`:

```
MyClientProxy myClient = new MyClientProxy("Endpoint_WMQ");
            try {
                   myClient.myMethod("HelloWorld!");
                   myClient.Close();
            }
            catch (TimeoutException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (CommunicationException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (Exception e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
```

Budowanie aplikacji klienckich WCF przy użyciu proxy klienta z konfiguracją programową

Zanim rozpoczniesz

1. Dodaj odwołanie do niestandardowego pliku kanału `IBM.XMS.WCF.dll` do projektu. Katalog `IBM.XMS.WCF.dll` znajduje się w katalogu *WMQInstallDir*`\bin` , gdzie *WMQInstallDir* to katalog, w którym jest zainstalowany produkt WebSphere MQ 7.
2. Dodaj instrukcję *using* do przestrzeni nazw `IBM.XMS.WCF` , na przykład: `using IBM.XMS.WCF`
3. Utwórz instancję elementu wiążącego i punktu końcowego kanału zgodnie z opisem w: "Tworzenie kanału niestandardowego WCF przez programowe informacje o powiązaniach i punktach końcowych" na stronie 622

O tym zadaniu

Jeśli wymagane są zmiany właściwości powiązania kanału, wykonaj następujące kroki:

1. Dodaj właściwości powiązania do składnika `transportBindingElement` , tak jak pokazano na poniższym rysunku:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
EndpointAddress address =
  new EndpointAddress("jms:/queue?destination=SampleQ@QM1&connectionFactory=

connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
```

2. Utwórz proxy klienta w sposób przedstawiony na poniższej ilustracji, gdzie *powiązanie* i *adres punktu końcowego* są powiązaniem i adresem punktu końcowego skonfigurowanym w kroku “1” na stronie 631 i przekazanego w:

```
 MyClientProxy myClient = new MyClientProxy(binding, endpoint address);
            try {
                   myClient.myMethod("HelloWorld!");
                   myClient.Close();
            }
            catch (TimeoutException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (CommunicationException e) {
                Console.Out.WriteLine(e);
                                 myClient.Abort();
            }
            catch (Exception e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
```

Korzystanie z przykładów WCF

Przykłady Windows Communication Foundation (WCF) zawierają kilka prostych przykładów, w jaki sposób można użyć niestandardowego kanału WebSphere MQ .

Aby zbudować przykładowe projekty, konieczne jest albo pakiet Microsoft .NET 3.5 SDK, albo Microsoft Visual Studio 2008.

Przykład prostego klienta jednokierunkowego i serwera WCF

Ten przykład demonstruje niestandardowy kanał produktu WebSphere MQ używany do uruchamiania usługi WCF (Windows Communication Foundation) z klienta WCF przy użyciu jednokierunkowego kształtu kanału.

O tym zadaniu

Usługa implementuje pojedynczą metodę, która wyprowadza łańcuch do konsoli. Klient został wygenerowany przy użyciu narzędzia `svcutil` w celu pobrania metadanych usługi z oddzielnie ujawnionego punktu końcowego HTTP zgodnie z opisem w sekcji “Generowanie proxy klienta WCF i plików konfiguracyjnych aplikacji przy użyciu narzędzia svcutil z metadanymi pochodzącym z uruchomionej usługi” na stronie 628 .

Przykład został skonfigurowany z określonymi nazwami zasobów zgodnie z opisem w poniższej procedurze. Jeśli konieczna jest zmiana nazw zasobów, należy również zmienić odpowiednią wartość w aplikacji klienckiej w pliku`\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\app.config` produktu *MQ_INSTALLATION_PATH*, a w aplikacji usługi w pliku`\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\TestServices.cs` produktu *MQ_INSTALLATION_PATH*, gdzie *MQ_INSTALLATION_PATH* jest katalogiem instalacyjnym produktu IBM WebSphere MQ. Więcej informacji na temat formatowania identyfikatora URI punktu

końcowego JMS można znaleźć w sekcji *WebSphere MQ Transport for SOAP* w dokumentacji produktu WebSphere MQ . Jeśli konieczne jest zmodyfikowanie przykładowego rozwiązania i źródła, potrzebne jest środowisko IDE, na przykład Microsoft Visual Studio 8 lub nowsze.

Procedura

1. Utwórz menedżer kolejek o nazwie *QM1*
2. Utwórz miejsce docelowe kolejki o nazwie *SampleQ*
3. Uruchom usługę, aby program nasłuchujący oczekiwał na komunikaty: Uruchom plik *MQ_INSTALLATION_PATH*`\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\bin\Release\TestService.exe` , gdzie *MQ_INSTALLATION_PATH* jest katalogiem instalacyjnym produktu IBM WebSphere MQ.
4. Uruchom klienta raz: uruchom plik`\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\bin\Release\TestClient.exe` produktu *MQ_INSTALLATION_PATH*, gdzie *MQ_INSTALLATION_PATH* jest katalogiem instalacyjnym produktu IBM WebSphere MQ.

   Aplikacja kliencka pętle pięciokrotnie wysyłając pięć komunikatów do *SampleQ*

Wyniki

Aplikacja usługi pobiera komunikaty z katalogu *SampleQ* i wyświetla `Hello World` na ekranie pięciokrotnie.

Co dalej

Przykład prostego klienta odpowiedzi żądania i serwera WCF

Ten przykład demonstruje niestandardowy kanał produktu WebSphere MQ używany do uruchamiania usługi WCF (Windows Communication Foundation) z klienta WCF przy użyciu kształtu kanału odpowiedzi na żądanie.

O tym zadaniu

Ta usługa udostępnia kilka prostych metod kalkulatora w celu dodania i odjęcia dwóch liczb, a następnie zwrócenia wyniku. Klient został wygenerowany przy użyciu narzędzia `svcutil` w celu pobrania metadanych usługi z oddzielnie ujawnionego punktu końcowego HTTP zgodnie z opisem w sekcji “Generowanie proxy klienta WCF i plików konfiguracyjnych aplikacji przy użyciu narzędzia svcutil z metadanymi pochodzącym z uruchomionej usługi” na stronie 628 .

Przykład został skonfigurowany z określonymi nazwami zasobów w sposób opisany w poniższej procedurze. Jeśli konieczna jest zmiana nazw zasobów, należy również zmienić odpowiednią wartość w aplikacji klienckiej w pliku`\Tools\wcf\samples\WCF\requestreply\client\app.config` produktu *MQ_INSTALLATION_PATH*, a w aplikacji usługi w pliku`\Tools\wcf\samples\WCF\requestreply\service\RequestReplyService.cs` produktu *MQ_INSTALLATION_PATH*, gdzie *MQ_INSTALLATION_PATH* jest katalogiem instalacyjnym produktu WebSphere MQ. Więcej informacji na temat formatowania identyfikatora URI punktu końcowego JMS można znaleźć w sekcji *WebSphere MQ Transport for SOAP* w dokumentacji produktu WebSphere MQ . Jeśli konieczne jest zmodyfikowanie przykładowego rozwiązania i źródła, potrzebne jest środowisko IDE, na przykład Microsoft Visual Studio 8 lub nowsze.

Procedura

1. Utwórz menedżer kolejek o nazwie *QM1*
2. Utwórz miejsce docelowe kolejki o nazwie *SampleQ*
3. Utwórz miejsce docelowe kolejki o nazwie *SampleReplyQ*
4. Uruchom usługę, aby program nasłuchujący oczekiwał na komunikaty: Uruchom plik *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\service\bin\Release\`

SimpleRequestReply_Service.exe , gdzie *MQ_INSTALLATION_PATH* jest katalogiem instalacyjnym produktu WebSphere MQ.

5. Uruchom klienta raz: uruchom plik\Tools\wcf\samples\WCF\requestreply\client\bin\Release\SimpleRequestReply_Client.exe produktu *MQ_INSTALLATION_PATH*, gdzie *MQ_INSTALLATION_PATH* to katalog instalacyjny produktu WebSphere MQ.

Wyniki

Po uruchomieniu klienta następujący proces jest uruchamiany i powtarzany cztery razy, więc w sumie wysyłane są pięć komunikatów:

1. Klient umieszcza komunikat żądania w pliku *SampleQ* i oczekuje na odpowiedź.
2. Usługa pobiera komunikat żądania z katalogu *SampleQ*.
3. Usługa dodaje i odejmuje niektóre wartości, korzystając z treści komunikatu.
4. Następnie usługa umieszcza wyniki w komunikacie w kolejce *SampleReplyQ*i oczekuje na umieszczenie nowego komunikatu przez klienta.
5. Klient pobiera komunikat z kolejki *SampleReplyQ* i wyświetla wyniki na ekranie.

Co dalej

Klient WCF do usługi .NET udostępnianej przez produkt WebSphere MQ

Przykładowe aplikacje klienckie i przykładowe aplikacje proxy usługi są dostarczane zarówno dla środowiska .NET, jak i środowiska Java. Przykłady są oparte na usłudze wyceny papierów wartościowych, która pobiera żądanie dotyczące wyceny akcji, a następnie udostępnia wycenę akcji.

Zanim rozpoczniesz

Przykład wymaga, aby środowisko usług serwerowych .NET SOAP over JMS zostało poprawnie zainstalowane i skonfigurowane w produkcie WebSphere MQ i było dostępne z lokalnego menedżera kolejek. Więcej informacji na temat instalowania i konfigurowania środowiska zawiera sekcja: “Instalowanie produktu WebSphere MQ Web transport dla protokołu SOAP” na stronie 984

Jeśli środowisko usług serwerowych .NET SOAP over JMS jest poprawnie zainstalowane i skonfigurowane w produkcie WebSphere MQ i jest dostępne z lokalnego menedżera kolejek, konieczne jest wykonanie dodatkowych kroków konfiguracyjnych.

1. Ustaw zmienną środowiskową WMQSOAP_HOME na katalog instalacyjny produktu WebSphere MQ , na przykład: C:\Program Files\IBM\WebSphere MQ
2. Upewnij się, że kompilator języka Java javac jest dostępny i znajduje się na ścieżce PATH.
3. Skopiuj plik axis.jar z katalogu prereqs/axis instalacyjnego dysku CD WebSphere do katalogu produkcyjnego produktu WebSphere MQ , na przykład: C:\Program Files\IBM\WebSphere MQ\java\lib\soap
4. Dodaj do zmiennej PATH: *MQ_INSTALLATION_PATH*\Java\lib , gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog, w którym zainstalowany jest produkt WebSphere MQ , na przykład: C:\Program Files\IBM\WebSphere MQ
5. Upewnij się, że położenie środowiska .NET zostało poprawnie podane w produkcie *MQ_INSTALLATION_PATH*\bin\amqwcallWSDL.cmd , gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog, w którym zainstalowany jest produkt WebSphere MQ , na przykład: C:\Program Files\IBM\WebSphere MQ. Położenie środowiska .NET można określić na przykład: set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin

Gdy poprzednie kroki są kompletne, przetestuj i uruchom usługę:

1. Przejdź do katalogu roboczego protokołu SOAP over JMS.
2. Wprowadź jedną z następujących komend, aby uruchomić test weryfikujący i pozostaw uruchomiony program nasłuchujący usługi:

- Dla środowiska .NET: *MQ_INSTALLATION_PATH*\Tools\soap\samples\runivt dotnet hold , gdzie *MQ_INSTALLATION_PATH* oznacza katalog, w którym zainstalowany jest produkt WebSphere MQ .
- W przypadku produktu AXIS:*MQ_INSTALLATION_PATH*\Tools\soap\samples\runivt Dotnet2AxisClient hold , gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog, w którym zainstalowany jest produkt WebSphere MQ .

Argument hold powoduje, że obiekty nasłuchiwania są uruchamiane po zakończeniu testu.

Jeśli podczas tej konfiguracji zostaną zgłoszone błędy, można usunąć wszystkie zmiany, tak aby procedura mogła zostać zrestartowana w następujący sposób:

1. Usuń wygenerowany katalog SOAP over JMS.
2. Usuń menedżer kolejek.

O tym zadaniu

Ten przykład demonstruje połączenie z klientem WCF z przykładową usługą .NET SOAP korzystającym z usługi JMS udostępnionej w produkcie WebSphere MQ przy użyciu jednokierunkowego kształtu kanału. Usługa implementuje prosty przykład StockQuote , który powoduje wyjście z łańcucha tekstowego do konsoli.

Klient został wygenerowany przy użyciu pliku WSDL w celu wygenerowania plików klienta zgodnie z opisem w sekcji “Generowanie proxy klienta WCF i plików konfiguracyjnych aplikacji przy użyciu narzędzia svcutil przy użyciu pliku WSDL” na stronie 629

Przykład został skonfigurowany z określonymi nazwami zasobów zgodnie z opisem w poniższej procedurze. Jeśli konieczna jest zmiana nazw zasobów, należy również zmienić odpowiednią wartość w aplikacji klienckiej w pliku *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQNET\default\client\app.config , a następnie w aplikacji usługi w pliku\tools\wcf\samples\WMQNET\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl produktu *MQ_INSTALLATION_PATH*, gdzie *MQ_INSTALLATION_PATH* oznacza katalog instalacyjny produktu WebSphere MQ. Więcej informacji na temat formatowania identyfikatora URI punktu końcowego JMS można znaleźć w sekcji *WebSphere MQ Transport for SOAP* w dokumentacji produktu WebSphere MQ .

Procedura

Uruchom klienta raz: uruchom plik\tools\wcf\samples\WMQNET\default\client\bin\Release\TestClient.exe produktu *MQ_INSTALLATION_PATH*, gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog instalacyjny produktu WebSphere MQ.

Aplikacja kliencka pęka pięć razy, wysyłając pięć komunikatów do kolejki próbkowania.

Wyniki

Aplikacja usługi pobiera komunikaty z kolejki przykładowej i wyświetla pięć razy Hello World ekranie pięć razy.

Klient WCF do usługi Java Axis obsługiwanej przez przykład produktu WebSphere MQ

Przykładowe aplikacje klienckie i przykładowe aplikacje proxy usługi są dostarczane zarówno dla środowiska Java, jak i dla środowiska .NET. Przykłady są oparte na usłudze wyceny papierów wartościowych, która pobiera żądanie dotyczące wyceny akcji, a następnie udostępnia wycenę akcji.

Zanim rozpoczniesz

Ten przykład wymaga, aby środowisko usług serwerowych .NET SOAP przez JMS zostało poprawnie zainstalowane i skonfigurowane w produkcie WebSphere MQ i było dostępne z lokalnego menedżera

kolejek. Więcej informacji na temat instalowania i konfigurowania środowiska zawiera sekcja: “Instalowanie produktu WebSphere MQ Web transport dla protokołu SOAP” na stronie 984

Jeśli środowisko usług serwerowych .NET SOAP over JMS jest poprawnie zainstalowane i skonfigurowane w produkcie WebSphere MQ i jest dostępne z lokalnego menedżera kolejek, konieczne jest wykonanie dodatkowych kroków konfiguracyjnych.

1. Ustaw zmienną środowiskową WMQSOAP_HOME na katalog instalacyjny produktu WebSphere MQ , na przykład: `C:\Program Files\IBM\WebSphere MQ`
2. Upewnij się, że kompilator języka Java `javac` jest dostępny i znajduje się na ścieżce PATH.
3. Skopiuj plik `axis.jar` z katalogu `prereqs/axis` instalacyjnego dysku CD WebSphere do katalogu instalacyjnego produktu WebSphere MQ .
4. Dodaj do zmiennej PATH: *MQ_INSTALLATION_PATH*`\Java\lib` , gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog, w którym zainstalowany jest produkt WebSphere MQ , na przykład: `C:\Program Files\IBM\WebSphere MQ`
5. Upewnij się, że położenie środowiska .NET zostało poprawnie podane w produkcie *MQ_INSTALLATION_PATH*`\bin\amqwcallWSDL.cmd` , gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog, w którym zainstalowany jest produkt WebSphere MQ , na przykład: `C:\Program Files\IBM\WebSphere MQ`. Położenie środowiska .NET można określić na przykład: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Gdy poprzednie kroki są kompletne, przetestuj i uruchom usługę:

1. Przejdź do katalogu roboczego protokołu SOAP over JMS.
2. Wprowadź jedną z następujących komend, aby uruchomić test weryfikujący i pozostaw uruchomiony program nasłuchujący usługi:
   - Dla środowiska .NET: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt dotnet hold` , gdzie *MQ_INSTALLATION_PATH* oznacza katalog, w którym zainstalowany jest produkt WebSphere MQ .
   - W przypadku produktu AXIS:*MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt Dotnet2AxisClient hold` , gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog, w którym zainstalowany jest produkt WebSphere MQ .

Argument `hold` powoduje, że obiekty nasłuchiwania są uruchamiane po zakończeniu testu.

Jeśli podczas tej konfiguracji zostaną zgłoszone błędy, można usunąć wszystkie zmiany, tak aby procedura została zrestartowana w następujący sposób:

1. Usuń wygenerowany katalog SOAP over JMS.
2. Usuń menedżer kolejek.

O tym zadaniu

Przykład demonstruje połączenie z klientem WCF z przykładową usługą Java SOAP korzystającym z usługi JMS firmy Axis udostępnianej w produkcie WebSphere MQ przy użyciu jednokierunkowego kształtu kanału. Usługa implementuje prosty przykład StockQuote , który wyprowadza łańcuch tekstowy do pliku, który jest zapisywany w bieżącym katalogu.

Klient został wygenerowany przy użyciu pliku WSDL w celu wygenerowania plików klienta zgodnie z opisem w sekcji “Generowanie proxy klienta WCF i plików konfiguracyjnych aplikacji przy użyciu narzędzia svcutil przy użyciu pliku WSDL” na stronie 629

Próbka została skonfigurowana z określonymi nazwami zasobów, zgodnie z opisem w niniejszym akapicie. Jeśli konieczna jest zmiana nazw zasobów, należy również zmienić odpowiednią wartość w aplikacji klienckiej w pliku *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WMQAxis\default\client\app.config` , a następnie w aplikacji usługi w pliku`\tools\wcf\samples\WMQAxis\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl` produktu *MQ_INSTALLATION_PATH*, gdzie *MQ_INSTALLATION_PATH* oznacza katalog instalacyjny produktu WebSphere MQ.

Procedura

Uruchom klienta raz: uruchom plik`\tools\wcf\samples\WMQAxis\default\client\bin\Release\TestClient.exe` produktu *MQ_INSTALLATION_PATH*, gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog instalacyjny produktu WebSphere MQ.

Aplikacja kliencka pęka pięć razy, wysyłając pięć komunikatów do kolejki próbkowania.

Wyniki

Aplikacja usługi pobiera komunikaty z kolejki przykładowej i dodaje `Hello World` razy do pliku w bieżącym katalogu.

Odsyłacze pokrewne

“Obsługa różnych nazw elementów odpowiedzi SOAP” na stronie 644

WCF oczekuje domyślnie, że nazwa zwróconej wartości ma być domyślnie w określonym formacie, ale usługa może nie zwrócić elementu o jego nazwie w oczekiwanym formacie.

Klient WCF do usługi Java udostępnianej przez serwer WebSphere Application Server

Przykładowe aplikacje klienckie i przykładowe aplikacje proxy usługi są dostarczane dla serwera WebSphere Application Server (WAS) 6. Udostępniana jest również usługa żądanie-odpowiedź.

Zanim rozpoczniesz

Ten przykład wymaga, aby używana była następująca konfiguracja produktu WebSphere MQ :

Tabela 77. Wymagana konfiguracja produktu WebSphere MQ

Obiekt	**Wymagana nazwa**
Menedżer kolejek	QM1
Kolejka lokalna	HelloWorld
Kolejka lokalna	HelloWorldOdpowiedz

Ten przykład wymaga również, aby środowisko usług serwerowych WebSphere Application Server V6 zostało poprawnie zainstalowane i skonfigurowane. Serwer WebSphere Application Server V6 domyślnie używa połączenia z trybem powiązań do nawiązywania połączenia z WebSphere MQ . Z tego powodu serwer WebSphere Application Server V6 musi być zainstalowany na tym samym komputerze, co menedżer kolejek.

Po skonfigurowaniu środowiska WAS należy wykonać następujące dodatkowe kroki konfiguracyjne:

1. W repozytorium JNDI serwera WebSphere Application Server należy utworzyć następujące obiekty JNDI:
   a. Miejsce docelowe kolejki JMS o nazwie `HelloWorld`
      - Ustaw nazwę JNDI na `jms/HelloWorld`
      - Ustaw nazwę kolejki na `HelloWorld`
   b. Fabryka połączeń kolejki JMS o nazwie `HelloWorldQCF`
      - Ustaw nazwę JNDI na `jms/HelloWorldQCF`
      - Ustaw nazwę menedżera kolejek na `QM1`
   c. Fabryka połączeń kolejki JMS o nazwie `WebServicesReplyQCF`
      - Ustaw nazwę JNDI na `jms/WebServicesReplyQCF`
      - Ustaw nazwę menedżera kolejek na `QM1`

2. Utwórz port nasłuchiwania komunikatów o nazwie `HelloWorldPort` na serwerze WebSphere Application Server z następującą konfiguracją:
   - Ustaw nazwę JNDI fabryki połączeń na wartość `jms/HelloWorldQCF` .
   - Ustaw nazwę JNDI miejsca docelowego na `jms/HelloWorld`
3. Zainstaluj aplikację usługi Web Service `HelloWorldEJBEAR.ear` na serwerze WebSphere Application Server w następujący sposób:
   a. Kliknij opcję **Aplikacje** > **Nowa aplikacja** > **Nowa aplikacja korporacyjna**.
   b. Przejdź do katalogu *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` , gdzie *MQ_INSTALLATION_PATH* to katalog instalacyjny produktu WebSphere MQ.
   c. Nie należy zmieniać żadnej z opcji domyślnych w kreatorze i restartować serwer aplikacji po zainstalowaniu aplikacji.

Po zakończeniu konfigurowania serwera WAS przetestuj usługę, uruchamiając ją jeden raz:

1. Przejdź do katalogu roboczego Soap over JMS.
2. Wprowadź tę komendę, aby uruchomić przykład: *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\TestClient.exe` , gdzie *MQ_INSTALLATION_PATH* to katalog instalacyjny produktu WebSphere MQ.

O tym zadaniu

Przykład demonstruje połączenie klienta WCF z przykładową usługą SOAP korzystającym z usługi JMS serwera WebSphere Application Server dostarczonym w próbkach WCF dołączonych do produktu WebSphere MQ V7za pomocą kształtu kanału żądanie-odpowiedź. Komunikaty przepływa między WCF i serwerem WebSphere Application Server za pomocą kolejek produktu WebSphere MQ . Usługa implementuje metodę `HelloWorld(...)` , która pobiera łańcuch i zwraca pozdrowienie do klienta.

Klient został wygenerowany przy użyciu narzędzia svcutil w celu pobrania metadanych usługi z oddzielnie ujawnionego punktu końcowego HTTP zgodnie z opisem w sekcji “Generowanie proxy klienta WCF i plików konfiguracyjnych aplikacji przy użyciu narzędzia svcutil z metadanymi pochodzącym z uruchomionej usługi” na stronie 628 .

Przykład został skonfigurowany z określonymi nazwami zasobów zgodnie z opisem w poniższej procedurze. Jeśli konieczna jest zmiana nazw zasobów, należy również zmienić odpowiednią wartość w aplikacji klienckiej w pliku *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\default\client\app.config` , a także w aplikacji usługi w *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` , gdzie *MQ_INSTALLATION_PATH* jest katalogiem instalacyjnym produktu WebSphere MQ. Więcej informacji na temat formatowania identyfikatora URI punktu końcowego JMS zawiera sekcja Składnia i parametry identyfikatora URI dla wdrożenia usługi Web Service .

The service and client are based upon the service and client outlined in the IBM Developer article *Budowanie usługi Web Service JMS przy użyciu protokołu SOAP over JMS i produktu WebSphere Studio*. Aby dowiedzieć się więcej na temat projektowania usług Web Service SOAP przez JMS, które są zgodne z kanałem niestandardowym produktu WebSphere MQ WCF, odpowiedni artykuł można znaleźć pod adresem: https://www.ibm.com/developerworks/websphere/library/techarticles/0402_du/0402_du.html.

Procedura

Uruchom klienta raz: uruchom plik`\tools\wcf\samples\WAS\default\client\bin\Release\TestClient.exe` produktu *MQ_INSTALLATION_PATH*, gdzie *MQ_INSTALLATION_PATH* to katalog instalacyjny produktu WebSphere MQ.

Aplikacja kliencka uruchamia jednocześnie obie metody usługi, wysyłając dwa komunikaty do kolejki przykładowej.

Wyniki

Aplikacja usługi pobiera komunikaty z kolejki przykładowej i udostępnia odpowiedź na wywołanie metody `HelloWorld(...)` , które jest wysyłane przez aplikację kliencką do konsoli.

Określanie problemu w kanale niestandardowym WCF dla produktu WebSphere MQ

Za pomocą funkcji śledzenia produktu WebSphere MQ można gromadzić szczegółowe informacje na temat różnych części kodu produktu WebSphere MQ . W przypadku korzystania z programu Windows Communication Foundation (WCF) dla niestandardowego śledzenia kanału WCF zintegrowanego ze śledzeniem infrastruktury Microsoft WCF generowane jest oddzielne dane wyjściowe śledzenia.

Pełne włączenie śledzenia dla niestandardowego kanału WCF powoduje utworzenie dwóch plików wyjściowych:

1. Niestandardowy śledzenie kanału WCF jest zintegrowane ze śledzeniem infrastruktury Microsoft WCF.
2. Niestandardowy śledzenie kanału WCF jest zintegrowane z platformą XMS .NET.

Dzięki dwóm wyjściom śledzenia problemy mogą być śledzone w każdym interfejsie za pomocą odpowiednich narzędzi, na przykład:

- Określanie problemu WCF przy użyciu odpowiednich narzędzi Microsoft .
- Problemy z klientem MQI produktu WebSphere MQ przy użyciu formatu śledzenia XMS .

Aby uprościć włączenie śledzenia, sterowany jest stos wywołań .NET 3 TraceSource i XMS .NET przy użyciu jednego interfejsu, zgodnie z opisem w: “Konfiguracja śledzenia WCF i nazwy plików śledzenia” na stronie 638.

Hierarchia wyjątków kanału niestandardowego WCF

Typy wyjątków zgłaszane przez kanał niestandardowy są spójne z produktem WCF i zazwyczaj są to wyjątek TimeoutException lub CommunicationException (lub podklasa CommunicationException).

Dodatkowe szczegóły dotyczące warunku błędu, jeśli są dostępne, są udostępniane przy użyciu wyjątków powiązanych lub wewnętrznych. Następujące wyjątki są typowymi przykładami, a każda warstwa w architekturze kanału wnosi dodatkowy powiązany wyjątek, na przykład: CommunicationsException ma powiązany wyjątek XMSException, który ma powiązany wyjątek MQException:

1. System.serviceModel.CommunicationsExceptions
2. IBM.XMS.XMSException
3. IBM.WMQ.MQException

Informacje o kluczu są przechwytywane i udostępniane w gromadzeniu danych o najwyższym wyjątku CommunicationException w hierarchii. Dzięki temu przechwytywanie i udostępnianie danych uniemożliwia aplikacjom łączenie się z każdą warstwą w architekturze kanału w celu interrogate powiązanych wyjątków oraz wszelkich dodatkowych informacji, które mogą zawierać. Zdefiniowane są następujące nazwy kluczy:

- IBM.XMS.WCF.ErrorCode: kod komunikatu o błędzie bieżącego niestandardowego wyjątku kanału.
- IBM.XMS.ErrorCode: Komunikat o błędzie z pierwszego wyjątku XMS w stosie.
- IBM.WMQ.ReasonCode: bazowy kod przyczyny produktu WebSphere MQ .
- IBM.WMQ.CompletionCode: bazowy kod zakończenia produktu WebSphere MQ .

Konfiguracja śledzenia WCF i nazwy plików śledzenia

Gdy śledzenie jest w pełni włączone, generuje dwa pliki wyjściowe, jeden do diagnozowania problemów WCF oraz jeden szczegółowy plik dla wewnętrznego materiału diagnostycznego śledzenia. Aby uprościć śledzenie, w stosach śledzenia .NET 3 TraceSource i XMS .NET używany jest jeden interfejs.

Dla niestandardowego kanału WCF dostępne są dwie różne metody śledzenia. Dwie metody śledzenia są aktywowane niezależnie lub razem. Każda metoda tworzy swój własny plik śledzenia, dlatego po aktywowaniu obu metod śledzenia generowane są dwa pliki wyjściowe śledzenia.

Aby zachować konfigurację i włączenie w możliwie prosty sposób, ten sam interfejs jest używany do kontrolowania obu metod śledzenia. Plik `app.config` musi być edytowany w taki sposób, aby zawierał odpowiednią konfigurację śledzenia zgodnie z opisem w poniższej sekcji. Użytkownicy mogą następnie dodać własne odpowiedniki, aby połączyć dane wyjściowe ze śledzeniem z własnej aplikacji.

Niestandardowe śledzenie kanału WCF nie jest domyślnie włączone. Najpierw należy utworzyć program nasłuchujący śledzenia, a następnie ustawić wymagany poziom śledzenia dla wybranego źródła śledzenia w pliku `app.config` .

Konfigurowanie niestandardowego kanału WCF za pomocą śledzenia infrastruktury WCF

Dodaj następującą sekcję kodu do sekcji `<system.diagnostics><sources>` w pliku `app.config` :

```
<source name="IBM.XMS.WCF" switchValue="Verbose,ActivityTracing">
    <listeners>
        <remove name="Default"/>
        <add name="NewListener"/>
    </listeners>
</source>
```

Poprzedni fragment kodu powoduje, że śledzenie kanału jest używane w środowisku .NET 3 TraceSource. Wszystkie wywołania plików konfiguracyjnych powiązanych z plikami wykonywalnymi są kontrolowane przez ten fragment kodu.

Konfigurowanie niestandardowego kanału WCF za pomocą śledzenia XMS .NET

Skonfigurowanie śledzenia środowiska .NET XMS wymaga dodania sekcji kodu do sekcji `<system.diagnostics><sources>` w pliku `app.config` . Fragment kodu jest jednak dodawany do rozszerzalnego elementu `<source>` , który jest wyświetlany w sekcji Konfigurowanie niestandardowego kanału WCF z śledzeniem infrastruktury WCF . Pomimo tego, że kod śledzenia infrastruktury WCF musi być obecny dla śledzenia XMS .NET, śledzenie infrastruktury WCF może być wyłączone, jeśli nie jest wymagane, zgodnie z opisem w sekcji Włączanie śledzenia WCF .

```
<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing"
 xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path"
 xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced">
    <listeners>
        <remove name="Default"/>
        <add name="NewListener"/>
    </listeners>
</source>
```

Zmienne konfiguracyjne śledzenia WCF

Tabela 78. Zmienne konfiguracyjne śledzenia WCF

Zmienna	**Opis**
nazwa	Podaj nazwę jako: `IBM.XMS.WCF`

Tabela 78. Zmienne konfiguracyjne śledzenia WCF (kontynuacja)

Zmienna	**Opis**
switchValue	Wartość switchValue steruje poziomem śledzenia. Jeśli wartość switchValue jest ustawiona na `Off`, to infrastruktura WCF TraceSource nie jest generowana. Każda inna wartość, taka jak `Verbose`, generuje TraceSource. Aby uzyskać szczegółowe informacje na temat poziomu śledzenia firmy Microsoft, należy zapoznać się z dokumentacją WCF lub przejść do strony WWW śledzenia produktu Microsoft WCF: https://msdn.microsoft.com/en-us/library/ms733025(vs.85).aspx .

xmsTraceSpecification =*ComponentName=typ=stan*	*ComponentName* to nazwa klasy, która ma być śledzona. W nazwie tej można użyć znaku wieloznacznego *. Na przykład:
	`*=all=enabled`
	określa, że mają być śledzeniem wszystkich klas, oraz
	`IBM.XMS.impl.*=all=enabled`
	określa, że wymagane jest tylko śledzenie interfejsu API.
	typ może być dowolnym z następujących typów śledzenia:
	• Wszystkie
	• debuguj
	• zdarzenie
	• EntryExit
	Stan może być włączony lub wyłączony.

xmsTraceFilePath= "*nazwa_pliku*"	Jeśli użytkownik nie określi xmsTraceFilePathlub jeśli xmsTraceFilePath jest obecny, ale zawiera pusty łańcuch, to plik śledzenia zostanie umieszczony w bieżącym katalogu. Aby zapisać plik śledzenia w nazwanym katalogu, należy podać nazwę katalogu w pliku xmsTraceFilePath, na przykład: `xmsTraceFilePath="c:\somepath"`
xmsTraceFileSize= "*wielkość*"	Maksymalna dozwolona wielkość pliku śledzenia. Gdy plik osiągnie tę wielkość, zostanie zarchiwizowany i jego nazwa zostanie zmieniona. Domyślna wartość maksymalna to 20 kB, która jest określona jako: `xmsTraceFileSize="20000000".`
xmsTraceFileNumber= "*liczba*"	Liczba plików śledzenia, które mają zostać zachowane. Wartością domyślną jest 4 (jeden aktywny plik i trzy pliki archiwum). Minimalna dozwolona liczba to dwa.

Tabela 78. Zmienne konfiguracyjne śledzenia WCF (kontynuacja)	
Zmienna	**Opis**

xmsTraceFormat="*format*"	Istnieją dwa poziomy formatu xmsTraceFormat: `basic` (podstawowy) i `advanced`(zaawansowany). Domyślny format śledzenia jest podstawowy, jeśli nie zostanie określony format xmsTrace, lub jeśli format xmsTracejest obecny, ale zawiera pusty łańcuch. Pliki śledzenia są tworzone w tym formacie, jeśli zostały określone:
	`xmsTraceFormat="basic"`
	Jeśli wymagane jest śledzenie, które jest zgodne z narzędziami analizatora śledzenia, należy określić:
	`traceFormat="advanced"`

Włączanie śledzenia WCF

Istnieją cztery kombinacje umożliwiające włączanie i wyłączanie dwóch różnych metod śledzenia. Cztery kombinacje wymagają edycji wartości sekcji zakodowanych opisanych w poprzednich sekcjach.

Istnieje również zmienna środowiskowa, która może zostać ustawiona. Więcej informacji na ten temat zawiera sekcja “Włączanie śledzenia WCF za pomocą zmiennej środowiskowej WCF_TRACE_ON” na stronie 642.

Ta tabela i przedstawione wartości są zależne od wcześniej pokazanych fragmentów kodu, które zostały już dodane do pliku `app.config` .

Tabela 79. Kombinacje włączania śledzenia WCF.		
Typ śledzenia	**Wartość zmieniona**	**Przykład**
Śledzenie XMS zostało włączone. Włączono narzędzie WCF TraceSource	Wartość `switchValue` nie jest ustawiona na `Wyłączone` .	`<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing" xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`
Śledzenie XMS zostało włączone. Narzędzie WCF TraceSource zostało wyłączone	Parametr `switchValue` jest ustawiony na wartość `Off` (Wyłączony) i podano `xmsTraceSpecification` (Wyłączone).	`<source name="IBM.XMS.WCF" switchValue="Off, ActivityTracing" xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`

Tabela 79. Kombinacje włączania śledzenia WCF. (kontynuacja)

Typ śledzenia	Wartość zmieniona	Przykład
Śledzenie XMS zostało wyłączone. Włączono narzędzie WCF TraceSource	Istnieją dwa sposoby osiągnięcia tego wyniku: • Zmienna `switchValue` nie jest ustawiona na `Wyłączone` , a`xmsTraceSpecification` nie została dodana. • Zmienna `switchValue` nie jest ustawiona na `Off` (Wyłączone), a parametr `xmsTraceSpecification` został ustawiony na `disabled` (wyłączone).	`<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing" xmsTraceSpecification="*=all=disabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`
Śledzenie XMS zostało wyłączone. Narzędzie WCF TraceSource zostało wyłączone	Istnieją trzy sposoby osiągnięcia tego wyniku: • Brak elementu `<source>` w pliku `app.config` • Zmienna `switchValue` jest ustawiona na `Wyłączone` , a`xmsTraceSpecification` nie została dodana. • Zmienna `switchValue` ma wartość `Wyłączone` , a właściwość `xmsTraceSpecification` została ustawiona na wartość `wyłączone` .	`<source name="IBM.XMS.WCF" switchValue="Off, ActivityTracing" xmsTraceSpecification="*=all=disabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`

Włączanie śledzenia WCF za pomocą zmiennej środowiskowej WCF_TRACE_ON

Podobnie jak w przypadku poprzednich metod opisanych w celu włączenia śledzenia WCF, śledzenie XMS .NET może być również włączone przy użyciu zmiennej środowiskowej WCF_TRACE_ON.

Ustawienie zmiennej środowiskowej WCF_TRACE_ON na dowolną wartość inną niż NULL jest równoznaczne z ustawieniem wartości `xmstraceSpecification` na `*=all=enabled`, na przykład: `"set WCF_TRACE_ON=true"`

Jeśli jednak parametr `xmstraceSpecification` jest jawnie ustawiony w pliku `app.config` , to zmienna środowiskowa WCF_TRACE_ON zostanie nadpisana.

Pliki wyjściowe śledzenia WCF i nazwy plików

Pliki śledzenia XMS są tradycyjnie nazwane przy użyciu nazwy podstawowej i formatu ID procesu: `xms_trace_`*`pid`*`.log`, gdzie *pid* jest identyfikatorem procesu.

Ponieważ pliki śledzenia XMS mogą być nadal tworzone równolegle z niestandardowymi plikami śledzenia kanału WCF, niestandardowe śledzenie kanału WCF zintegrowane z plikami wyjściowymi śledzenia środowiska XMS .NET ma następujący format, aby uniknąć pomyłek: `wcfxms_trace_`*`pid`*`.log`, gdzie *pid* jest identyfikatorem procesu.

Plik wyjściowy śledzenia jest domyślnie tworzony w bieżącym katalogu roboczym, ale w razie potrzeby można je ponownie zdefiniować.

WCF XMS pierwsza technologia obsługi awarii (FFST)

Istnieje możliwość zebrania szczegółowych informacji na temat różnych części kodu produktu WebSphere MQ za pomocą funkcji śledzenia produktu WebSphere MQ . XMS FFST ma własne pliki konfiguracyjne i wyjściowe dla niestandardowego kanału WCF.

Pliki śledzenia XMS FFST są tradycyjnie nazwane przy użyciu nazwy podstawowej i formatu ID procesu: `xmsffdc`*`pid_date`*`.txt`, gdzie *pid* jest identyfikatorem procesu, a *data* jest godziną i datą.

Ponieważ pliki śledzenia XMS FFST mogą być nadal tworzone równolegle z niestandardowymi plikami XMS FFST kanału WCF, pliki wyjściowe XMS FFST kanału WCF mają następujący format, aby uniknąć pomyłki: `wcfffdc`*`pid_date`*`.txt`, gdzie *pid* jest identyfikatorem procesu, a *data* jest godziną i datą.

Ten plik wyjściowy śledzenia jest domyślnie tworzony w bieżącym katalogu roboczym, ale to miejsce docelowe może zostać ponownie zdefiniowane, jeśli to konieczne.

Niestandardowy kanał WCF z nagłówkiem śledzenia środowiska .NET XMS jest podobny do następującego przykładu:

```
************* Start Display XMS WCF Environment *************
  Product Name  :- value
  WCF Version   :- value
  Level    :- value
*************  End Display XMS WCF Environment  *************
```

Pliki śledzenia FFST są formatowane w standardowy sposób, bez formatowania specyficznego dla kanału niestandardowego.

Informacje o wersji WCF

Informacje o wersji WCF z określeniem problemu i zawarte w metadanych asemblacji kanału niestandardowego.

Niestandardowy kanał produktu WebSphere MQ dla metadanych wersji WCF można pobrać na jeden z trzech sposobów:

- Korzystanie z narzędzia dspmqver programu narzędziowego WebSphere MQ . Informacje na temat korzystania z narzędzia dspmqver można znaleźć w sekcji: dspmqver
- Przy użyciu okna dialogowego właściwości programu Okna Explorer: w Eksploratorze Okna kliknij prawym przyciskiem myszy opcję **IBM.XMS.WCF.dll** > **Właściwości** > **Wersja**.
- Z poziomu informacji nagłówkowych dowolnego z kanałów FFST lub plików śledzenia. Więcej informacji na temat nagłówka FFST zawiera sekcja: “WCF XMS pierwsza technologia obsługi awarii (FFST)” na stronie 643

Wskazówki i wskazówki WCF

Następujące wskazówki i wskazówki nie są w znaczącym porządku i mogą zostać dodane do momentu zwolnienia nowych wersji dokumentacji. Są to tematy, które mogą zaoszczędzić czas, jeśli są istotne dla pracy, którą wykonujesz.

Eksternalizowanie wyjątków od hosta usługi WCF

W przypadku usług udostępnianych za pomocą hosta usługi WCF; nieobsługiwane wyjątki zgłaszane przez usługę, internałty WCF lub stos kanałów nie są domyślnie eksternalizowane. Aby można było uzyskać informacje o tych wyjątkach, należy zarejestrować procedurę obsługi błędów.

Poniższy kod przedstawia przykład definiowania zachowania usługi procedury obsługi błędów, która może zostać zastosowana jako atrybut usługi:

```
using System.ServiceModel.Dispatcher;
using System.Collections.ObjectModel;
....
    public class ErrorHandlerBehaviorAttribute : Attribute, IServiceBehavior, IErrorHandler
    {
```

```
        //
        // IServiceBehavior Interface
        //
        public void AddBindingParameters(ServiceDescription serviceDescription,
           ServiceHostBase serviceHostBase, Collection<ServiceEndpoint> endpoints,
           BindingParameterCollection bindingParameters)
        {
        }
        public void ApplyDispatchBehavior(ServiceDescription serviceDescription,
            ServiceHostBase serviceHostBase)
        {
            foreach (ChannelDispatcher channelDispatcher in serviceHostBase.ChannelDispatchers)
{
                channelDispatcher.ErrorHandlers.Add(this);
            }
        }
        public void Validate(ServiceDescription serviceDescription, ServiceHostBase
serviceHostBase)
        {
        }

        //
        // IErrorHandler Interface
        //
        public bool HandleError(Exception e)
        {
            // Process the exception in the required way, in this case just outputting to the
console
            Console.Out.WriteLine(e);

            // Always return false to allow any other error handlers to run
            return false;
        }
        public void ProvideFault(Exception error, MessageVersion version, ref Message fault)
        {
        }
    }
```

Obsługa różnych nazw elementów odpowiedzi SOAP

WCF oczekuje domyślnie, że nazwa zwróconej wartości ma być domyślnie w określonym formacie, ale usługa może nie zwrócić elementu o jego nazwie w oczekiwanym formacie.

WCF ma konwencję, w której oczekiwano, że zwrócona wartość zostanie nazwana w następującym formacie: *methodName*`Result` , gdzie *methodName* to nazwa operacji usługi. Na przykład w przypadku usługi o nazwie `getQuote`WCF oczekuje, że odpowiedź zostanie wywołana: `getQuoteResult`.

Jednak usługa może zwrócić element o nazwie, która nie jest zgodna z tym formatem.

W przypadku uruchamiania narzędzia scvutil w celu wygenerowania klienta proxy, jeśli plik WSDL określa inną nazwę, interfejs proxy dodaje parametry do instrukcji WCF o nazwie, która ma być poszukiwaana. Na przykład:

```
[System.ServiceModel.OperationContractAttribute(Action = "", ReplyAction = "*")]
[System.ServiceModel.XmlSerializerFormatAttribute(Style =
System.ServiceModel.OperationFormatStyle.Rpc,
                                                  Use =
System.ServiceModel.OperationFormatUse.Encoded)]
[return: System.ServiceModel.MessageParameterAttribute(Name = "getQuoteReturn")]
float getQuote(string in0);
```

Jeśli zostanie utworzony własny interfejs (na przykład przez dodanie metody żądanie-odpowiedź do istniejącego interfejsu proxy), należy upewnić się, że do interfejsu zostaną dodane te same parametry, jeśli usługa zwróci inną nazwę. Jeśli tego nie zrobicie, najczęstszym problemem jest to, że wywołanie metody service zawsze zwraca wartość null; jeśli obiekt jest zwracany, to metoda zwraca wartość null, ale jeśli zwracana jest wartość liczbowa, taka jak liczba całkowita, to metoda zwraca 0.

Używanie języka C++

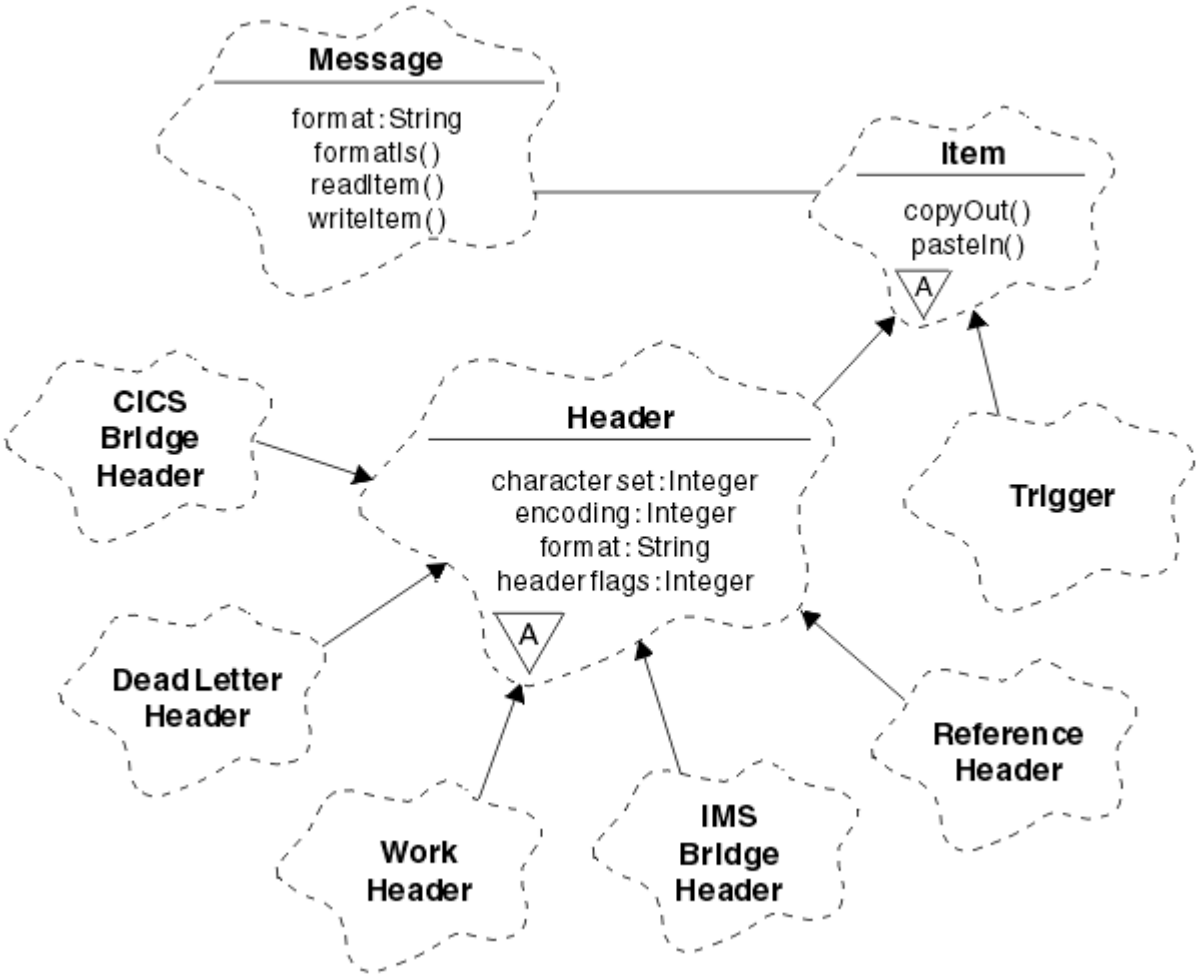
ProduktWebSphere MQ udostępnia klasy języka C++ odpowiadające obiektom WebSphere MQ oraz niektóre dodatkowe klasy równoważne z typami danych tablicowych. Udostępnia ona wiele funkcji, które nie są dostępne w interfejsie MQI.

W przypadku produktu WebSphere MQ w wersji 7.0rozszerzenia interfejsów programistycznych produktu WebSphere MQ nie będą stosowane w klasach C + +.

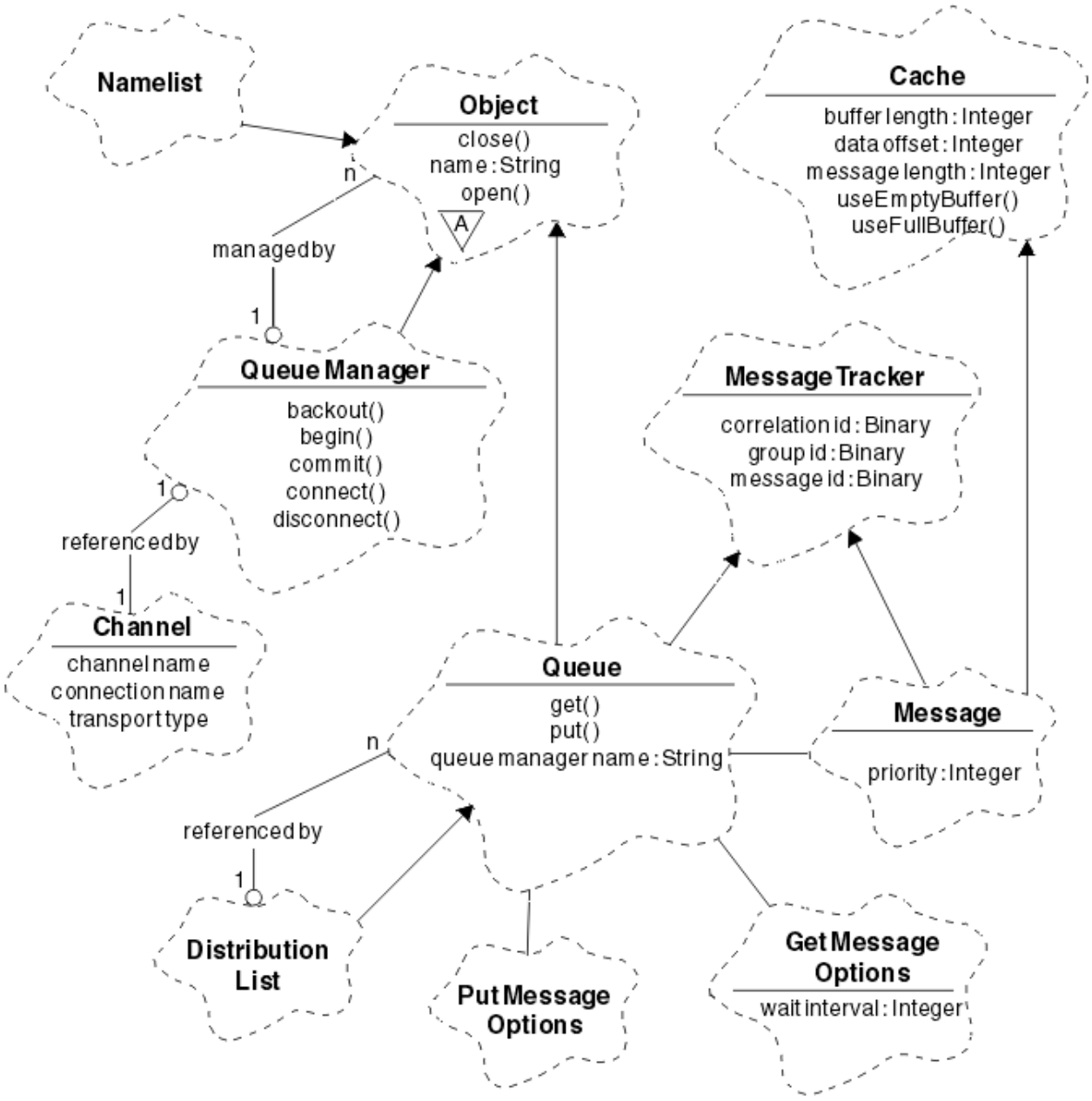
W programie WebSphere MQ C++ dostępne są następujące funkcje:

- Automatyczne inicjowanie struktur danych produktu WebSphere MQ .
- Połączenie z menedżerem kolejek i otwieranie kolejki w trybie just-in-time.
- Niejawne zamknięcie kolejki i rozłączenie menedżera kolejek.
- Przesłanie i przyjęcie nagłówka niedostarczanego listu.
- Przesyłanie i odbieranie nagłówka mostu IMS .
- Transmisja i przyjęcie nagłówka komunikatu referencyjnego.
- Wyzwalanie odbioru komunikatu.
- Przesyłanie i odbieranie nagłówka mostu CICS .
- Przesyłanie i odbieranie nagłówka pracy.
- Definicja kanału klienta.

Poniższe diagramy klasy Booch pokazują, że wszystkie klasy są zasadniczo równoległe do tych jednostek WebSphere MQ w proceduralnym MQI (na przykład przy użyciu C), które mają albo uchwyty, albo struktury danych. Wszystkie klasy dziedziczą z klasy ImqError (patrz ImqError klasa C++), co pozwala na powiązanie warunku błędu z każdym obiektem.



Rysunek 120. Klasy języka C++ w produkcie WebSphere MQ (obsługa elementów)



Rysunek 121. Klasy produktu WebSphere MQ C++ (zarządzanie kolejkami)

Aby poprawnie zinterpretować diagramy klas Booch, należy pamiętać o następujących konwencjach:

- Metody i godne uwagi atrybuty są wyświetlane pod nazwą *klasa* .
- Mały trójkąt w chmurze oznacza *klasę abstrakcyjną* .
- *Dziedziczenie* jest oznaczane strzałką w klasie nadrzędnej.
- Nieprzyozdobiona linia między chmurami oznacza *relację kooperatywną* między klasami.
- Linia urządzona z liczbą oznacza *relację referencyjną* między dwiema klasami. Liczba wskazuje liczbę obiektów, które mogą uczestniczyć w danej relacji w dowolnym momencie.

Następujące klasy i typy danych są używane w sygnaturach metod C + + w klasach zarządzania kolejkami (patrz sekcja Rysunek 121 na stronie 646) oraz w klasach obsługi elementów (patrz Rysunek 120 na stronie 645):

- Klasa ImqBinary (patrz Klasa ImqBinary C++), która hermetyzuje tablice bajtów, takie jak MQBYTE24.
- Typ danych ImqBoolean , który jest zdefiniowany jako **typedef unsigned char ImqBoolean**.

- Klasa ImqString (patrz Klasa ImqString C++), która hermetyzuje tablice znakowe, takie jak MQCHAR64.

Obiekty ze strukturami danych są podsumowane w ramach odpowiednich klas obiektów. Do poszczególnych pól struktury danych (patrz Skorowidz języka C++ i MQI) można uzyskać dostęp do metod.

Obiekty z uchwytami znajdują się w hierarchii klas ImqObject (patrz Klasa ImqObject C++) i udostępniają enkapsulowane interfejsy do interfejsu MQI. Obiekty tych klas wykazują inteligentne zachowanie, które może zredukować liczbę wywołań metod wymaganych w odniesieniu do proceduralnego interfejsu MQI. Na przykład można ustanowić i odrzucić połączenia menedżera kolejek zgodnie z wymaganiami lub otworzyć kolejkę z odpowiednimi opcjami, a następnie zamknąć ją.

Klasa ImqMessage (patrz klasa ImqMessage C++ class) hermetyzuje strukturę danych MQMD, a także działa jako punkt wstrzymujący dla danych użytkownika i *elementów* (patrz "Odczytywanie komunikatów w języku C++" na stronie 656), udostępniając buforowane obiekty buforu. Można podać bufory o stałej długości dla danych użytkownika i wiele razy używać buforu. Ilość danych znajdujących się w buforze może być różna od jednego do następnego. Alternatywnie system może udostępnić bufor o elastycznej długości i zarządzać nim. Zarówno wielkość bufora (kwota dostępna do odbioru wiadomości), jak i ilość rzeczywiście wykorzystana (ilość bajtów do transmisji lub liczba rzeczywiście odebranych bajtów) stają się ważnymi względami.

Pojęcia pokrewne

Przegląd techniczny

"Programy przykładowe C++" na stronie 647
Dostarczane są cztery programy przykładowe w celu zademonstrować pobieranie i umieszczanie komunikatów.

"Uwagi dotyczące języka C++" na stronie 651
Ta kolekcja tematów zawiera szczegóły dotyczące używania języka C++ oraz konwencji, które należy uwzględnić podczas pisania programów aplikacji, które korzystają z interfejsu MQI (Message Queue Interface).

"Przygotowywanie danych komunikatu w języku C++" na stronie 655
Dane komunikatu są przygotowywane w buforze, który może być dostarczony przez system lub aplikację. Istnieją zalety jednej z metod. Podano przykłady użycia buforu.

"Wybór języka programowania do użycia" na stronie 80
W tej sekcji znajdują się informacje na temat języków programowania i środowisk obsługiwanych przez produkt IBM WebSphere MQ, a także niektóre zagadnienia dotyczące ich używania.

"Projektowanie aplikacji" na stronie 7
Produkt IBM WebSphere MQ udostępnia kilka sposobów tworzenia aplikacji w celu wysyłania i odbierania komunikatów, które są potrzebne do obsługi procesów biznesowych. Istnieje również możliwość tworzenia aplikacji do zarządzania menedżerami kolejek i powiązanymi zasobami.

Odsyłacze pokrewne

"Budowanie programów w języku C++ w programie WebSphere MQ" na stronie 662
Zostanie wyświetlony adres URL obsługiwanych kompilatorów wraz z komendami, które mają być używane do kompilowania, łączenia i uruchamiania programów w języku C + + oraz przykładów na platformach WebSphere MQ .

Skorowidz języka C++ i MQI

Klasy języka C++ produktu WebSphere MQ

Programy przykładowe C++

Dostarczane są cztery programy przykładowe w celu zademonstrować pobieranie i umieszczanie komunikatów.

Programy przykładowe są następujące:

- HELLO WORLD (imqwrld.cpp)
- SPUT (imqsput.cpp)

- SGET (imqsget.cpp)
- DPUT (imqdput.cpp)

Przykładowe programy znajdują się w katalogach przedstawionych w sekcji Tabela 80 na stronie 648.

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

Tabela 80. Położenie przykładowych programów

Środowisko	Katalog zawierający źródło	Katalog zawierający zbudowany programy
AIX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/ia
HP-UX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/ samp/bin/ah (patrz uwaga “2” na stronie 648)
Solaris	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/as
Linux	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/
Windows	*MQ_INSTALLATION_PATH*\tools\cplus\ samples	*MQ_INSTALLATION_PATH*\tools\cplus\ przykłady \bin\vn (patrz uwaga “3” na stronie 648)

Uwagi:

1. Programy zbudowane przy użyciu kompilatora ILE C++ dla IBM i znajdują się w bibliotece QMQM. Pliki włączane znajdują się w katalogu /QIBM/ProdData/mqm/inc.
2. Programy zbudowane przy użyciu kompilatora HP ANSI C + + znajdują się w katalogu *MQ_INSTALLATION_PATH*/samp/bin/ah. Więcej informacji zawiera sekcja “Budowanie programów C++ w systemie HP-UX” na stronie 663.
3. Programy zbudowane przy użyciu programu Microsoft Visual Studio znajdują się w katalogu *MQ_INSTALLATION_PATH*\tools\cplus\samples\bin\vn. Więcej informacji na temat tych kompilatorów zawiera sekcja “Budowanie programów C++ w systemie Windows” na stronie 669.

Przykładowy program HELLO WORLD (imqwrld.cpp)

Ten przykładowy program C++ przedstawia sposób umieszczania i pobierania regularnego datagramu (struktury C) przy użyciu klasy ImqMessage .

W tym programie przedstawiono sposób umieszczania i pobierania regularnego datagramu (struktura C) przy użyciu klasy ImqMessage . W tym przykładzie użyto nielicznych wywołań metod, które wykorzystują niejawne wywołania metod, takie jak **open**, **close**, i **disconnect**.

Na wszystkich platformach z wyjątkiem systemu z/OS

Jeśli używane jest połączenie z serwerem WebSphere MQ, należy wykonać jedną z następujących procedur:

- Aby korzystać z istniejącej kolejki domyślnej, SYSTEM.DEFAULT.LOCAL.QUEUE, uruchom program **imqwrlds** bez przekazywania żadnych parametrów
- Aby użyć tymczasowej dynamicznie przypisanej kolejki, uruchom komendę **imqwrlds** przekazując nazwę domyślnej kolejki modelowej (SYSTEM.DEFAULT.MODEL.QUEUE.

Jeśli używane jest połączenie klienckie z produktem WebSphere MQ, należy wykonać jedną z następujących procedur:

- Skonfiguruj zmienną środowiskową MQSERVER (więcej informacji na ten temat zawiera sekcja MQSERVER), a następnie uruchom komendę **imqwrldc**, lub
- Uruchom **imqwrldc** przechodząc jako parametry **queue-name**, **queue-manager-name**i **channel-definition**, gdzie typowym **channel-definition** może być SYSTEM.DEF.SVRCONN/TCP/*nazwa_hosta*(1414)

Kod przykładowy

```
extern "C" {
#include <stdio.h>
}

#include <imqi.hpp> // WebSphere MQ C++

#define EXISTING_QUEUE "SYSTEM.DEFAULT.LOCAL.QUEUE"

#define BUFFER_SIZE 12

static char gpszHello[ BUFFER_SIZE ] = "Hello world" ;
int main ( int argc, char * * argv ) {
  ImqQueueManager manager ;
  int iReturnCode = 0 ;

  // Connect to the queue manager.
  if ( argc > 2 ) {
    manager.setName( argv[ 2 ] );
  }
  if ( manager.connect( ) ) {
    ImqQueue * pqueue = new ImqQueue ;
    ImqMessage * pmsg = new ImqMessage ;

    // Identify the queue which will hold the message.
    pqueue -> setConnectionReference( manager );
    if ( argc > 1 ) {
      pqueue -> setName( argv[ 1 ] );


      // The named queue can be a model queue, which will result in
      // the creation of a temporary dynamic queue, which will be
      // destroyed as soon as it is closed. Therefore we must ensure
      // that such a queue is not automatically closed and reopened.
      // We do this by setting open options which will avoid the need
      // for closure and reopening.
      pqueue -> setOpenOptions( MQOO_OUTPUT | MQOO_INPUT_SHARED |
                                MQOO_INQUIRE );
    } else {
      pqueue -> setName( EXISTING_QUEUE );

      // The existing queue is not a model queue, and will not be
      // destroyed by automatic closure and reopening. Therefore we
      // will let the open options be selected on an as-needed basis.
      // The queue will be opened implicitly with an output option
      // during the "put", and then implicitly closed and reopened
      // with the addition of an input option during the "get".
    }

    // Prepare a message containing the text "Hello world".
    pmsg -> useFullBuffer( gpszHello , BUFFER_SIZE );
    pmsg -> setFormat( MQFMT_STRING );

    // Place the message on the queue, using default put message
    // Options.
    // The queue will be automatically opened with an output option.
    if ( pqueue -> put( * pmsg ) ) {
      ImqString strQueue( pqueue -> name( ) );

      // Discover the name of the queue manager.
      ImqString strQueueManagerName( manager.name( ) );
      printf( "The queue manager name is %s.\n",
              (char *)strQueueManagerName );

      // Show the name of the queue.
      printf( "Message sent to %s.\n", (char *)strQueue );
```

```
      // Retrieve the data message just sent ("Hello world" expected)
      // from the queue, using default get message options. The queue
      // is automatically closed and reopened with an input option
      // if it is not already open with an input option. We get the
      // message just sent, rather than any other message on the
      // queue, because the "put" will have set the ID of the message
      // so, as we are using the same message object, the message ID
      // acts as in the message object, a filter which says that we
      // are interested in a message only if it has this
      // particular ID.

      if ( pqueue -> get( * pmsg ) ) {
        int iDataLength = pmsg -> dataLength( );

        // Show the text of the received message.
        printf( "Message of length %d received, ", iDataLength );

        if ( pmsg -> formatIs( MQFMT_STRING ) ) {
          char * pszText = pmsg -> bufferPointer( );

          // If the last character of data is a null, then we can
          // assume that the data can be interpreted as a text
          // string.
          if ( ! pszText[ iDataLength - 1 ] ) {
            printf( "text is \"%s\".\n", pszText );
          } else {
            printf( "no text.\n" );
          }

        } else {
          printf( "non-text message.\n" );
        }
      } else {
        printf( "ImqQueue::get failed with reason code %ld\n",
                pqueue -> reasonCode( ) );
        iReturnCode = (int)pqueue -> reasonCode( );
      }

    } else {
      printf( "ImqQueue::open/put failed with reason code %ld\n",
              pqueue -> reasonCode( ) );
      iReturnCode = (int)pqueue -> reasonCode( );
    }

    // Deletion of the queue will ensure that it is closed.
    // If the queue is dynamic then it will also be destroyed.
    delete pqueue ;
    delete pmsg ;

  } else {
    printf( "ImqQueueManager::connect failed with reason code %ld\n"
            manager.reasonCode( ) );
    iReturnCode = (int)manager.reasonCode( );
  }

  // Destruction of the queue manager ensures that it is
  // disconnected.  If the queue object were still available
  // and open (which it is not), the queue would be closed
  // prior to disconnection.

  return iReturnCode ;
}
```

Przykładowe programy SPUT (imqsput.cpp) i SGET (imqsget.cpp)

Te programy w języku C + + umieszczają komunikaty i pobierają komunikaty z kolejki o określonej nazwie.

Te przykłady przedstawiają użycie następujących klas:

- ImqError (patrz sekcja ImqError klasa C++)
- ImqMessage (patrz sekcja ImqMessage C++ class)
- ImqObject (patrz klasa ImqObject C++ class)
- ImqQueue (patrz klasa ImqQueue C++ class)
- Menedżer ImqQueue(patrz sekcja ImqQueue)

Postępuj zgodnie z odpowiednimi instrukcjami, aby uruchomić programy.

Na wszystkich platformach z wyjątkiem systemu z/OS

1. Uruchom komendę **imqsputs** *nazwa-kolejki*.
2. Wpisz linie tekstu w konsoli. Wiersze te są umieszczane jako komunikaty w określonej kolejce.
3. Wprowadź pusty wiersz, aby zakończyć wprowadzanie danych.
4. Uruchom komendę **imqsgets** *nazwa-kolejki* , aby pobrać wszystkie wiersze i wyświetlić je na konsoli.

Przykładowy program DPUT (imqdput.cpp)

Ten przykładowy program C++ umieszcza komunikaty do listy dystrybucyjnej składającej się z dwóch kolejek.

Funkcja DPUT pokazuje użycie klasy ImqDistributionList (patrz ImqDistributionList C++ class). Ten przykład nie jest obsługiwany w systemie z/OS.

1. Run **imqdputs** *queue-name-1 queue-name-2* to place messages on the two named queues.
2. Uruchom **imqsgets** *queue-name-1* i **imqsgets** *queue-name-2* , aby pobrać komunikaty z tych kolejek.

Uwagi dotyczące języka C++

Ta kolekcja tematów zawiera szczegóły dotyczące używania języka C++ oraz konwencji, które należy uwzględnić podczas pisania programów aplikacji, które korzystają z interfejsu MQI (Message Queue Interface).

Pliki nagłówkowe C++

Pliki nagłówkowe są udostępniane jako część definicji interfejsu MQI w celu ułatwienia pisania programów aplikacji WebSphere MQ w języku C + +.

Te pliki nagłówkowe są podsumowane w poniższej tabeli.

Tabela 81. Pliki nagłówkowe C/C++

Nazwa pliku	**Spis treści**
IMQI.HPP	Klasy MQI języka C++ (zawiera CMQC.H i IMQTYPE.H)
IMQTYPE.H	Definiuje typ danych **ImqBoolean**
CMQC.H	Struktury danych MQI i stałe manifestów

Aby zwiększyć przenośność aplikacji, należy zakodować nazwę pliku nagłówkowego małymi literami w dyrektywie preprocesora **#include** :

```
#include <imqi.hpp> // C++ classes
```

Metody i atrybuty w języku C++

Nazwy metod znajdują się w mieszanym przypadku. Do parametrów i zwracanych wartości mają zastosowanie różne zagadnienia. Dostęp do atrybutów można uzyskać, używając odpowiednio ustawionych metod i metod pobierania.

Parametry metod, które są *const* , są przeznaczone tylko dla danych wejściowych. Parametry z podpisami łącznie ze wskaźnikiem (*) lub odwołaniem (&) są przekazywane przez referencję. Wartości zwracane, które nie zawierają wskaźnika lub odwołania, są przekazywane przez wartość; w przypadku zwracanych obiektów są to nowe obiekty, które są odpowiedzialne za program wywołujący.

Niektóre sygnatury metod zawierają elementy, które przyjmują wartość domyślną, jeśli nie została określona. Takie elementy są zawsze pod koniec podpisów i są oznaczane znakiem równości (=); wartość po znaku równości wskazuje wartość domyślną, która ma zastosowanie, jeśli element jest pominięty.

Wszystkie nazwy metod w tych klasach są małymi literami, zaczynając od małej litery. Każde słowo, z wyjątkiem pierwszego w nazwie metody, rozpoczyna się od litery. Skróty nie są używane, chyba że ich znaczenie jest szeroko rozumiane. Używane skróty to *id* (dla tożsamości) i *sync* (dla synchronizacji).

Dostęp do atrybutów obiektu można uzyskać za pomocą zestawu i metod pobierania. Metoda set rozpoczyna się od słowa *set*; metoda get nie ma przedrostka. Jeśli atrybut ma wartość *tylko do odczytu*, nie ma ustawionej metody.

Atrybuty są inicjowane do poprawnych stanów podczas budowy obiektu, a stan obiektu jest zawsze spójny.

Typy danych w języku C++

Wszystkie typy danych są definiowane przez instrukcję C **typedef** .

Typ **ImqBoolean** jest zdefiniowany jako **unsigned char** w elemencie IMQTYPE.H i może mieć wartości PRAWDA i FAŁSZ. Obiektów klasy **ImqBinary** można używać w miejsce tablic **MQBYTE** , a obiekty klasy **ImqString** w miejsce **char ***. Wiele metod zwraca obiekty, a nie wskaźniki **char** i **MQBYTE** , aby ułatwić zarządzanie pamięcią masową. Wszystkie wartości zwracane stają się odpowiedzialnością programu wywołującego, a w przypadku zwracanego obiektu można usunąć pamięć masową przy użyciu usuwania.

Manipulowanie łańcuchami binarnymi w języku C++

Łańcuchy danych binarnych są deklarowane jako obiekty klasy **ImqBinary** . Obiekty tej klasy mogą być kopiowane, porównywane i ustawiane przy użyciu znanych operatorów języka C. Dostępny jest przykładowy kod.

Poniższy przykładowy kod przedstawia operacje na łańcuchu binarnym:

```
#include <imqi.hpp> // C++ classes

ImqMessage message ;
ImqBinary id, correlationId ;
MQBYTE24 byteId ;

correlationId.set( byteId, sizeof( byteId ) ); // Set.
id = message.id( );                            // Assign.
if ( correlationId == id ) {                   // Compare.
  ...
```

Manipulowanie łańcuchami znaków w języku C++

Dane znakowe są często zwracane w obiektach klasy **ImqString** , które mogą być rzutowane na **znak *** przy użyciu operatora konwersji. Klasa ImqString zawiera metody pomocne przy przetwarzaniu łańcuchów znaków.

Gdy dane znakowe są akceptowane lub zwracane za pomocą metod języka C++ w języku C++, dane znakowe są zawsze zakończone znakiem o kodzie zero i mogą mieć dowolną długość. Jednak niektóre limity są nakładane przez produkt WebSphere MQ , który może spowodować obcinanie informacji. Aby ułatwić zarządzanie pamięcią masową, dane znakowe są często zwracane w obiektach klasy **ImqString** . Obiekty te można rzutować na **znak *** za pomocą udostępnionego operatora konwersji, a używane w celach *tylko do odczytu* w wielu sytuacjach, w których wymagany jest znak **char *** .

Uwaga: Wynik konwersji **char *** z obiektu klasy **ImqString** może mieć wartość NULL.

Mimo że funkcje języka C mogą być używane na **znaku ***, istnieją specjalne metody klasy **ImqString** , które są preferowane; **długość operatora**() jest odpowiednikiem **strlen** i **storage**() wskazuje pamięć przydzieloną dla danych znakowych.

Początkowy stan obiektów w języku C++

Wszystkie obiekty mają spójny stan początkowy odzwierciedlający ich atrybuty. Wartości początkowe są zdefiniowane w opisach klas.

Korzystanie z C z C++

Jeśli używane są funkcje C z programu C++, należy dołączyć odpowiednie nagłówki.

W poniższym przykładzie przedstawiono program `string.h` dołączony do programu w języku C + +:

```
extern "C" {
#include <string.h>
}
```

Konwencje notacyjne języka C++

W tym przykładzie przedstawiono sposób wywoływania metod i deklarowania parametrów.

W tym przykładzie kodu używane są metody i parametry**ImqBoolean ImqQueue**::**get**(**ImqMessage &** *msg*)

Zadeklaruj i wykorzystaj parametry w następujący sposób:

```
ImqQueueManager * pmanager ;    // Queue manager
ImqQueue * pqueue ;             // Message queue
ImqMessage msg ;                // Message
char szBuffer[ 100 ];          // Buffer for message data

pmanager = new ImqQueueManager ;
pqueue = new ImqQueue ;
pqueue -> setName( "myreplyq" );
pqueue -> setConnectionReference( pmanager );

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );

if ( pqueue -> get( msg ) ) {
  long lDataLength = msg.dataLength( );

  ...
}
```

Operacje niejawne w języku C++

Kilka operacji może wystąpić w sposób niejawny, *właśnie w czasie*, w celu spełnienia wymagań wstępnych dla pomyślnego wykonania metody. Te niejawne operacje są połączone, otwarte, ponownie otwarte, zamykane i rozłączane. Za pomocą atrybutów klasy można sterować łączem i otwierać zachowanie niejawne.

Połączenie

Obiekt menedżera ImqQueuejest połączony automatycznie dla dowolnej metody, która powoduje wywołanie interfejsu MQI (patrz Skorowidz języka C++ i MQI).

Otwórz

Obiekt ImqObject jest otwierany automatycznie dla dowolnej metody, której wynikiem jest wywołanie MQGET, MQINQ, MQPUT lub MQSET. Użyj metody **openFor** , aby określić jedną lub większą liczbę odpowiednich wartości **open option** .

Otwórz ponownie

Obiekt ImqObject jest ponownie otwierany automatycznie dla dowolnej metody, której wynikiem jest wywołanie MQGET, MQINQ, MQPUT lub MQSET, w którym obiekt jest już otwarty, ale istniejące **opcje otwarcia** nie są odpowiednie, aby umożliwić pomyślne wywołanie MQI. Obiekt jest tymczasowo zamknięty przy użyciu tymczasowej wartości **close options** parametru MQCO_NONE. Użyj metody **openFor** , aby dodać odpowiedni**open option**.

Ponowne otwarcie może powodować problemy w określonych okolicznościach:

- Tymczasowa kolejka dynamiczna jest niszczona, gdy jest zamknięta i nigdy nie może zostać ponownie otwarta.
- Kolejka otwarta dla wejścia na wyłączność (jawnie lub domyślnie) może być dostępna dla innych osób w oknie potencjalnej transakcji podczas zamykania i ponownego otwierania.
- Pozycja kursora przeglądania jest tracona, gdy kolejka jest zamknięta. Ta sytuacja nie zapobiega zamknięciu i ponownym otwieraniu, ale uniemożliwia późniejsze użycie kursora, dopóki nie zostanie ponownie użyta wartość MQGMO_BROWSE_FIRST.
- Kontekst ostatniego pobranego komunikatu jest tracony, gdy kolejka jest zamknięta.

Jeśli którakolwiek z tych okoliczności wystąpi lub może być przewidziana, należy unikać ponownego otwierania, jawnie ustawiając odpowiednie **opcje otwarte** przed otwartymi obiektami (jawnie lub niejawnie).

Ustawienie opcji **open options** w sposób jawny dla złożonych sytuacji związanych z obsługą kolejek powoduje lepszą wydajność i pozwala uniknąć problemów związanych z korzystaniem z ponownie otwartych.

Zamknij

Obiekt ImqObject jest zamykany automatycznie w każdym punkcie, w którym stan obiektu nie jest już wykonalny, na przykład wtedy, gdy odniesienie do połączenia ImqObject zostanie zerwane lub jeśli obiekt ImqObject zostanie zniszczony.

Rozłącz

Menedżer ImqQueuejest odłączony automatycznie w dowolnym momencie, w którym połączenie nie jest już wykonalne, na przykład wtedy, gdy odniesienie do połączenia ImqObject zostanie zerwane lub jeśli obiekt ImqQueueManager zostanie zniszczony.

Łańcuchy binarne i łańcuchy znaków w języku C++

Klasa ImqString hermetyzuje tradycyjny format danych *char* * . Klasa ImqBinary hermetyzuje binarną tablicę bajtów. Niektóre metody, które ustawiają dane znakowe, mogą obcinać dane.

Metody, które ustawiają dane znakowe (**char ***), zawsze przyjmują kopię danych, ale niektóre metody mogą obciąć kopię, ponieważ niektóre limity są nakładane przez produkt WebSphere MQ.

Klasa ImqString (patrz Klasa ImqString C++) hermetyzuje tradycyjny produkt **znak *** i zapewnia obsługę następujących elementów:

- Porównanie
- Konkatenacja
- Kopiowanie
- Konwersja typu integer-to-text i text-to-integer
- Wyodrębnianie znacznika (słowa)
- Konwersja wielkich liter

Klasa ImqBinary (patrz Klasa ImqBinary C++) hermetyzuje binarne tablice bajtowe o dowolnej wielkości. W szczególności jest on używany do przechowywania następujących atrybutów:

- **token rozliczania** (MQBYTE32)

- **znacznik połączenia** (MQBYTE128)
- **identyfikator korelacji** (MQBYTE24)
- **facility token (znacznik obiektu)** (MQBYTE8)
- **identyfikator grupy** (MQBYTE24)
- **identyfikator instancji** (MQBYTE24)
- **ID komunikatu** (MQBYTE24)
- **token komunikatu** (MQBYTE16)
- **identyfikator instancji transakcji** (MQBYTE16)

Miejsce, w którym te atrybuty należą do obiektów następujących klas:

- Nagłówek ImqCICSBridge(patrz ImqCICSBridgeHeader C++ class)
- ImqGetMessageOptions (patrz ImqGetMessageOptions C++ class)
- Nagłówek ImqIMSBridge(patrz sekcja ImqIMSBridgeHeader C++ class)
- ImqMessageTracker (patrz ImqMessageTracker C++ class)
- Menedżer ImqQueue(patrz sekcja ImqQueue)
- Nagłówek ImqReference(patrz sekcja ImqReferenceHeader C++ class)
- ImqWorkNagłówek (patrz ImqWorkKlasa nagłówka C++)

Klasa ImqBinary udostępnia również obsługę porównywania i kopiowania.

Nieobsługiwane funkcje w języku C++

Klasy i metody języka C++ produktu WebSphere MQ są niezależne od platformy WebSphere MQ . Mogą one zatem oferować niektóre funkcje, które nie są obsługiwane na określonych platformach.

W przypadku próby użycia funkcji na platformie, na której nie jest ona obsługiwana, funkcja jest wykrywana przez produkt WebSphere MQ , ale nie przez powiązania języka C + +. Produkt WebSphere MQ zgłasza błąd programu, podobnie jak inne błędy MQI.

Przesyłanie komunikatów w języku C++

Ta kolekcja tematów zawiera szczegółowe informacje na temat przygotowywania, odczytywania i zapisywania komunikatów w języku C + +.

Przygotowywanie danych komunikatu w języku C++

Dane komunikatu są przygotowywane w buforze, który może być dostarczony przez system lub aplikację. Istnieją zalety jednej z metod. Podano przykłady użycia buforu.

Po wysłaniu komunikatu dane komunikatu są najpierw przygotowywane w buforze zarządzanym przez obiekt ImqCache (patrz sekcja ImqCache). Bufor jest powiązany (przez dziedziczenie) z każdym obiektem ImqMessage (patrz klasa ImqMessage C++ class): może być dostarczony przez aplikację (przy użyciu metody **useEmptyBuffer** lub **useFullBuffer**) lub automatycznie przez system. Zaletą aplikacji dostarczających bufor komunikatów jest to, że kopiowanie danych nie jest konieczne w wielu przypadkach, ponieważ aplikacja może korzystać bezpośrednio z przygotowanych obszarów danych. Wadą jest to, że podany bufor ma stałą długość.

Bufor może być ponownie wykorzystany, a liczba przesłanych bajtów może być zmieniana za każdym razem za pomocą metody **setMessageLength** przed przesłaniem.

Jeśli system jest dostarczany automatycznie przez system, liczba dostępnych bajtów jest zarządzana przez system, a dane mogą być kopiowane do buforu komunikatów za pomocą, na przykład, metody ImqCache **zapis** lub metody ImqMessage **writeItem** . Bufor komunikatów rośnie zgodnie z potrzebą. W miarę wzrostu bufora, nie dochodzi do utraty wcześniej zapisanych danych. Duży lub wieloczęściowy komunikat może być zapisany w kolejnych fragmentach.

W poniższych przykładach przedstawiono uproszczone wysyłanie komunikatów.

1. Użyj przygotowanych danych w buforze dostarczonym przez użytkownika

```
char szBuffer[ ] = "Hello world" ;

msg.useFullBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
```

2. Użyj przygotowanych danych w buforze dostarczonym przez użytkownika, w którym wielkość buforu przekracza wielkość danych

```
char szBuffer[ 24 ] = "Hello world" ;

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.setMessageLength( 12 );
```

3. Kopiowanie danych do buforu dostarczanego przez użytkownika

```
char szBuffer[ 12 ];

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

4. Kopiowanie danych do buforu dostarczonego przez system

```
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

5. Kopiowanie danych do buforu dostarczonego przez system przy użyciu obiektów (obiekty ustawiają format komunikatu i treść)

```
ImqString strText( "Hello world" );

msg.writeItem( strText );
```

Odczytywanie komunikatów w języku C++

Bufor może być dostarczony przez aplikację lub system. Dostęp do danych możliwy jest bezpośrednio z buforu lub odczytany sekwencyjnie. Dla każdego typu komunikatu istnieje równorzędna klasa. Podano przykładowy kod.

W przypadku odbierania danych aplikacja lub system może dostarczyć odpowiedni bufor komunikatów. Ten sam bufor może być używany zarówno dla wielu transmisji, jak i wielokrotnych przyjęć dla konkretnego obiektu ImqMessage . Jeśli bufor komunikatów jest dostarczany automatycznie, rośnie w celu dostosowania dowolnej długości danych. Jednak bufor komunikatów dostarczony przez aplikację może nie być wystarczająco duży, aby pomieścić otrzymane dane. Następnie może wystąpić obcięcie lub niepowodzenie, w zależności od opcji użytych do przyjęcia komunikatu.

Dane przychodzące mogą być dostępne bezpośrednio z buforu komunikatów, w którym to przypadku długość danych wskazuje łączną ilość danych przychodzących. Alternatywnie, dane przychodzące mogą być odczytywaniekolejno z buforu komunikatów. W tym przypadku wskaźnik danych zajmuje się następnym bajtem danych przychodzących, a wskaźnik danych i długość danych są aktualizowane za każdym razem, gdy dane są odczytywaane.

Elementy są elementami komunikatu, a wszystko to w obszarze użytkownika buforu komunikatów, które muszą być przetwarzane sekwencyjnie i osobno. Oprócz zwykłych danych użytkownika element może być nagłówkiem niewysłanych wiadomości lub komunikatem wyzwalacza. Elementy są zawsze powiązane z formatami komunikatów. Formaty komunikatów są ***nie*** zawsze powiązane z elementami.

Istnieje klasa obiektu dla każdej pozycji, która odpowiada rozpoznawalnym formatom komunikatów produktu WebSphere MQ . Istnieje jeden dla nagłówka niedostarczonych komunikatów i jeden dla komunikatu wyzwalacza. Brak klasy obiektu dla danych użytkownika. Oznacza to, że po wyczerpaniu rozpoznawalnych formatów przetwarzanie pozostałej części jest pozostawione do programu aplikacji. Klasy dla danych użytkownika mogą być zapisywane przez specjalizującą się klasę ImqItem .

W poniższym przykładzie przedstawiono przyjęcie komunikatu, które uwzględnia liczbę potencjalnych elementów, które mogą poprzedzać dane użytkownika w wyimaginowanej sytuacji. Dane użytkownika innego niż element są definiowane jako elementy, które pojawiają się po elementach, które można zidentyfikować. Bufor automatyczny (wartość domyślna) jest używany do przechowywania dowolnej ilości danych komunikatu.

```
ImqQueue queue ;
ImqMessage msg ;

if ( queue.get( msg ) ) {

  /* Process all items of data in the message buffer. */
  do while ( msg.dataLength( ) ) {
    ImqBoolean bFormatKnown = FALSE ;
    /* There remains unprocessed data in the message buffer. */

    /* Determine what kind of item is next. */

    if ( msg.formatIs( MQFMT_DEAD_LETTER_HEADER ) ) {
      ImqDeadLetterHeader header ;
      /* The next item is a dead-letter header.                */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;

      if ( msg.readItem( header ) ) {
        /* The dead-letter header has been extricated from the */
        /* buffer and transformed into a dead-letter object.   */
        /* The encoding and character set of the dead-letter   */
        /* object itself are MQENC_NATIVE and MQCCSI_Q_MGR.    */
        /* The encoding and character set from the dead-letter */
        /* header have been copied to the message attributes   */
        /* to reflect any remaining data in the buffer.        */

        /* Process the information in the dead-letter object.  */
        /* Note that the encoding and character set have       */
        /* already been processed.                             */
        ...
      }
      /* There might be another item after this, */
      /* or just the user data.                  */
    }
    if ( msg.formatIs( MQFMT_TRIGGER ) ) {
      ImqTrigger trigger ;
      /* The next item is a trigger message.                   */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;
      if ( msg.readItem( trigger ) ) {

        /* The trigger message has been extricated from the */
        /* buffer and transformed into a trigger object.    */
        /* Process the information in the trigger object. */
        ...
      }

      /* There is usually nothing after a trigger message. */
    }

    if ( msg.formatIs( FMT_USERCLASS ) ) {
      UserClass object ;
      /* The next item is an item of a user-defined class.     */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
```

```
      bFormatKnown = TRUE ;

      if ( msg.readItem( object ) ) {
        /* The user-defined data has been extricated from the */
        /* buffer and transformed into a user-defined object. */

        /* Process the information in the user-defined object. */
        ...
      }

      /* Continue looking for further items. */
    }
    if ( ! bFormatKnown ) {
      /* There remains data that is not associated with a specific*/
      /* item class.                                               */
      char * pszDataPointer = msg.dataPointer( );       /* Address.*/
      int iDataLength = msg.dataLength( );              /* Length. */

      /* The encoding and character set for the remaining data are */
      /* reflected in the attributes of the message object, even   */
      /* if a dead-letter header was present.                      */
      ...

    }

  }
}
```

W tym przykładzie FMT_USERCLASS jest stałą reprezentującą 8-znakową nazwę formatu powiązaną z obiektem klasy `UserClassi` jest definiowana przez aplikację.

Produkt `UserClass` pochodzi z klasy ImqItem (patrz klasa ImqItem C++ class) i implementuje z tej klasy metody wirtualnego **copyOut** i **pasteIn** .

Kolejne dwa przykłady pokazują kod z klasy ImqDeadLetterHeader (patrz ImqDeadLetterHeader C++ class). W pierwszym przykładzie przedstawiono kod niestandardowy-*zapis* w sposób niestandardowy.

```
// Insert a dead-letter header.
// Return TRUE if successful.
ImqBoolean ImqDeadLetterHeader :: copyOut ( ImqMessage & msg ) {
  ImqBoolean bSuccess ;
  if ( msg.moreBytes( sizeof( omqdlh ) ) ) {
    ImqCache cacheData( msg ); // Preserve original message content.
    // Note original message attributes in the dead-letter header.
    setEncoding( msg.encoding( ) );
    setCharacterSet( msg.characterSet( ) );
    setFormat( msg.format( ) );

    // Set the message attributes to reflect the dead-letter header.
    msg.setEncoding( MQENC_NATIVE );
    msg.setCharacterSet( MQCCSI_Q_MGR );
    msg.setFormat( MQFMT_DEAD_LETTER_HEADER );
    // Replace the existing data with the dead-letter header.
    msg.clearMessage( );
    if ( msg.write( sizeof( omqdlh ), (char *) & omqdlh ) ) {
      // Append the original message data.
      bSuccess = msg.write( cacheData.messageLength( ),
                            cacheData.bufferPointer( ) );
    } else {
      bSuccess = FALSE ;
    }
  } else {
    bSuccess = FALSE ;
  }
  // Reflect and cache error in this object.
  if ( ! bSuccess ) {
    setReasonCode( msg.reasonCode( ) );
    setCompletionCode( msg.completionCode( ) );
  }

  return bSuccess ;
}
```

W drugim przykładzie przedstawiono niestandardowy komunikat-*odczytywanie* .

```
// Read a dead-letter header.
// Return TRUE if successful.
ImqBoolean ImqDeadLetterHeader :: pasteIn ( ImqMessage & msg ) {
  ImqBoolean bSuccess = FALSE ;

  // First check that the eye-catcher is correct.
  // This is also our guarantee that the "character set" is correct.
  if ( ImqItem::structureIdIs( MQDLH_STRUC_ID, msg ) ) {
    // Next check that the "encoding" is correct, as the MQDLH
    // contains numeric data.
    if ( msg.encoding( ) == MQENC_NATIVE ) {

      // Finally check that the "format" is correct.
      if ( msg.formatIs( MQFMT_DEAD_LETTER_HEADER ) ) {
        char * pszBuffer = (char *) & omqdlh ;
        // Transfer the MQDLH from the message and move pointer on.
        if ( bSuccess = msg.read( sizeof( omdlh ), pszBuffer ) ) {
          // Update the encoding, character set and format of the
          // message to reflect the remaining data.
          msg.setEncoding( encoding( ) );
          msg.setCharacterSet( characterSet( ) );
          msg.setFormat( format( ) );
        } else {

          // Reflect the cache error in this object.
          setReasonCode( msg.reasonCode( ) );
          setCompletionCode( msg.completionCode( ) );
        }
      } else {
        setReasonCode( MQRC_INCONSISTENT_FORMAT );
        setCompletionCode( MQCC_FAILED );
      }
    } else {
      setReasonCode( MQRC_ENCODING_ERROR );
      setCompletionCode( MQCC_FAILED );
    {
  } else {
    setReasonCode( MQRC_STRUC_ID_ERROR );
    setCompletionCode( MQCC_FAILED );
  }

  return bSuccess ;
}
```

W przypadku buforu automatycznego pamięć masowa buforu jest *ulotna*. Oznacza to, że dane buforu mogą być przechowywane w innym miejscu fizycznym po każdym wywołaniu metody **get** . Z tego powodu do każdej przywoływanej danych buforu należy użyć metod **bufferPointer** lub **dataPointer** w celu uzyskania dostępu do danych komunikatu.

Użytkownik może chcieć, aby program zarezerwował stały obszar na potrzeby odbierania danych komunikatu. W tym przypadku należy wywołać metodę **useEmptyBuffer** przed użyciem metody **get** .

Użycie stałego, nieautomatycznego obszaru ogranicza liczbę komunikatów do maksymalnej wielkości, dlatego ważne jest, aby rozważyć opcję MQGMO_ACCEPT_TRUNCATED_MSG obiektu ImqGetMessageOptions . Jeśli ta opcja nie zostanie podana (wartość domyślna), kod przyczyny MQRC_TRUNCATED_MSG_FAILED może być oczekiwany. Jeśli ta opcja jest określona, kod przyczyny MQRC_TRUNCATED_MSG_ACCEPTED może być oczekiwany w zależności od projektu aplikacji.

W następnym przykładzie pokazano, w jaki sposób można wykorzystać stałą przestrzeń pamięci do odbierania komunikatów:

```
char * pszBuffer = new char[ 100 ];

msg.useEmptyBuffer( pszBuffer, 100 );
gmo.setOptions( MQGMO_ACCEPT_TRUNCATED_MSG );
queue.get( msg, gmo );

delete [ ] pszBuffer ;
```

W tym fragmencie kodu bufor może być zawsze adresowany bezpośrednio, przy użyciu metody *pszBuffer*, a nie za pomocą metody **bufferPointer** . Jednak lepszym sposobem jest użycie metody **dataPointer** w celu uzyskania dostępu ogólnego. Aplikacja (nie obiekt klasy ImqCache) musi odrzucić bufor zdefiniowany przez użytkownika (nieautomatyczny).

Uwaga: Określenie pustego wskaźnika i zerowej długości z parametrem **useEmptyBuffer** nie mianuje buforu o stałej długości równej zero, co może być oczekiwane. Ta kombinacja jest interpretowana jako żądanie do zignorowania wszystkich poprzednich buforów zdefiniowanych przez użytkownika, a zamiast tego jest przywracana do użycia buforu automatycznego.

Zapisywanie komunikatu do kolejki niedostarczonych komunikatów w języku C++

Przykładowy kod programu do zapisu komunikatu do kolejki niedostarczonych komunikatów.

Typowym przypadkiem komunikatu wieloczęściowego jest jeden zawierający nagłówek niedostarczonych komunikatów. Dane z komunikatu, którego nie można przetworzyć, są dopisane do nagłówka niedostarczonych komunikatów.

```
ImqQueueManager mgr ;        // The queue manager.
ImqQueue queueIn ;           // Incoming message queue.
ImqQueue queueDead ;         // Dead-letter message queue.
ImqMessage msg ;             // Incoming and outgoing message.
ImqDeadLetterHeader header ; // Dead-letter header information.

// Retrieve the message to be rerouted.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Set up the dead-letter header information.
header.setDestinationQueueManagerName( mgr.name( ) );
header.setDestinationQueueName( queueIn.name( ) );
header.setPutApplicationName( /* ? */ );
header.setPutApplicationType( /* ? */ );
header.setPutDate( /* TODAY */ );
header.setPutTime( /* NOW */ );
header.setDeadLetterReasonCode( FB_APPL_ERROR_1234 );

// Insert the dead-letter header information. This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the dead-letter queue.
queueDead.setConnectionReference( mgr );
queueDead.setName( mgr.deadLetterQueueName( ) );
queueDead.put( msg );
```

Zapisywanie komunikatu do mostu IMS w języku C++

Przykładowy kod programu do zapisu wiadomości do mostu IMS .

Komunikaty wysłane do mostu WebSphere MQ-IMS mogą używać specjalnego nagłówka. Nagłówek mostu IMS jest poprzedzony przedrostkiem zwykłych danych komunikatu.

```
ImqQueueManager    mgr;         // The queue manager.
ImqQueue           queueBridge; // IMS bridge message queue.
ImqMessage         msg;         // Outgoing message.
ImqIMSBridgeHeader header;      // IMS bridge header.

// Set up the message.
//
// Here we are constructing a message with format
// MQFMT_IMS_VAR_STRING, and appropriate data.
//
msg.write( 2,       /* ? */ );         // Total message length.
msg.write( 2,       /* ? */ );         // IMS flags.
msg.write( 7,       /* ? */ );         // Transaction code.
```

```
msg.write( /* ? */, /* ? */ );         // String data.
msg.setFormat( MQFMT_IMS_VAR_STRING ); // The format attribute.

// Set up the IMS bridge header information.
//
// The reply-to-format is often specified.
// Other attributes can be specified, but all have default values.
//
header.setReplyToFormat( /* ? */ );

// Insert the IMS bridge header into the message.
//
// This will:
// 1) Insert the header into the message buffer, before the existing
//    data.
// 2) Copy attributes out of the message descriptor into the header,
//    for example the IMS bridge header format attribute will now
//    be set to MQFMT_IMS_VAR_STRING.
// 3) Set up the message attributes to describe the header, in
//    particular setting the message format to MQFMT_IMS.
//
msg.writeItem( header );

// Send the message to the IMS bridge queue.
//
queueBridge.setConnectionReference( mgr );
queueBridge.setName( /* ? */ );
queueBridge.put( msg );
```

Zapisywanie komunikatu do mostu CICS w języku C++

Przykładowy kod programu do zapisu komunikatu do mostu CICS .

Komunikaty wysłane do produktu WebSphere MQ for z/OS przy użyciu mostu CICS wymagają specjalnego nagłówka. Nagłówek mostu CICS jest poprzedzony przedrostkiem regularnego danych komunikatu.

```
ImqQueueManager mgr ;         // The queue manager.
ImqQueue queueIn ;            // Incoming message queue.
ImqQueue queueBridge ;        // CICS bridge message queue.
ImqMessage msg ;              // Incoming and outgoing message.
ImqCicsBridgeHeader header ;  // CICS bridge header information.

// Retrieve the message to be forwarded.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Set up the CICS bridge header information.
// The reply-to format is often specified.
// Other attributes can be specified, but all have default values.
header.setReplyToFormat( /* ? */ );

// Insert the CICS bridge header information.  This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the CICS bridge queue.
queueBridge.setConnectionReference( mgr );
queueBridge.setName( /* ? */ );
queueBridge.put( msg );
```

Zapisywanie komunikatu z nagłówkiem pracy w języku C++

Przykładowy kod programu do zapisu komunikatu przeznaczonego dla kolejki zarządzanej za pomocą menedżera obciążenia z/OS .

Komunikaty wysyłane do produktu WebSphere MQ for z/OS, które są przeznaczone dla kolejki zarządzanej przez produkt z/OS Workload Manager, wymagają specjalnego nagłówka. Nagłówek pracy jest poprzedzony przedrostkiem zwykłych danych komunikatu.

```
ImqQueueManager mgr ;             // The queue manager.
ImqQueue queueIn ;                // Incoming message queue.
ImqQueue queueWLM ;               // WLM managed queue.
ImqMessage msg ;                  // Incoming and outgoing message.
ImqWorkHeader header ;            // Work header information

// Retrieve the message to be forwarded.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Insert the Work header information.  This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the WLM managed queue.
queueWLM.setConnectionReference( mgr );
queueWLM.setName( /* ? */ );
queueWLM.put( msg );
```

Budowanie programów w języku C++ w programie WebSphere MQ

Zostanie wyświetlony adres URL obsługiwanych kompilatorów wraz z komendami, które mają być używane do kompilowania, łączenia i uruchamiania programów w języku C + + oraz przykładów na platformach WebSphere MQ .

Kompilatory dla każdej obsługiwanej platformy i wersji produktu WebSphere MQ są wymienione na stronie wymagań systemowych produktu WebSphere MQ pod adresem Wymagania systemowe produktu IBM WebSphere MQ.

Komenda, którą należy skompilować i połączyć z programem WebSphere MQ w języku C++, zależy od instalacji i wymagań. W poniższych przykładach przedstawiono typowe komendy kompilowania i łączenia dla niektórych kompilatorów przy użyciu domyślnej instalacji produktu WebSphere MQ na wielu platformach.

Tworzenie programów C++ w systemie AIX

Zbuduj programy w języku C++ WebSphere MQ w systemie AIX za pomocą kompilatora XL C Enterprise Edition .

Klient

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

32-bitowa aplikacja wielowątkowa

```
xlC -o imqsputc_32 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqc23ia -limqb23ia -lmqic
```

32-bitowa aplikacja wielowątkowa

```
xlC_r -o imqsputc_32_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqc23ia_r -limqb23ia_r -lmqic_r
```

Aplikacja wielowątkowa z 64-bitową wersją

```
xlC -q64 -o imqsputc_64 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqc23ia -limqb23ia -lmqic
```

Aplikacja wielowątkowa z 64-bitowym

```
xlC_r -q64 -o imqsputc_64_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqc23ia_r -limqb23ia_r -lmqic_r
```

Serwer

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

32-bitowa aplikacja wielowątkowa

```
xlC -o imqsput_32 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqs23ia -limqb23ia -lmqm
```

32-bitowa aplikacja wielowątkowa

```
xlC_r -o imqsput_32_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqs23ia_r -limqb23ia_r -lmqm_r
```

Aplikacja wielowątkowa z 64-bitową wersją

```
xlC -q64 -o imqsput_64 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqs23ia -limqb23ia -lmqm
```

Aplikacja wielowątkowa z 64-bitowym

```
xlC_r -q64 -o imqsput_64_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqs23ia_r -limqb23ia_r -lmqm_r
```

Budowanie programów C++ w systemie HP-UX

Kompilowanie programów WebSphere MQ C++ w systemie HP-UX za pomocą kompilatorów aC+ + lub aCC .

W systemach HP-UX Itanium, WebSphere MQ obsługuje tylko standardowe środowisko wykonawcze. Użyj kompilatora aCC .

- libimqi23bh.sl udostępnia klasy języka C++ produktu WebSphere MQ dla standardowego środowiska wykonawczego.
- W celu zapewnienia zgodności z wcześniejszymi wersjami odsyłacz symboliczny jest udostępniany z pliku libimqi23ah.sl do libimqi23bh.sl.

IA64 (IPF)

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

Klient: IA64 (IPF)

32-bitowa aplikacja wielowątkowa

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh -lmqic
```

32-bitowa aplikacja wielowątkowa

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh_r -lmqic_r -lpthread
```

Aplikacja wielowątkowa z 64-bitową wersją

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsputc_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh -lmqic
```

Aplikacja wielowątkowa z 64-bitowym

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsputc_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh_r
-lmqic_r
-lpthread
```

Serwer: IA64 (IPF)

32-bitowa aplikacja wielowątkowa

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh -lmqm
```

32-bitowa aplikacja wielowątkowa

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh_r -lmqm_r -lpthread
```

Aplikacja wielowątkowa z 64-bitową wersją

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsput_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh -lmqm
```

Aplikacja wielowątkowa z 64-bitowym

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsput_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh_r
-lmqm_r
-lpthread
```

Budowanie programów C++ w systemie Linux

Zbuduj programy w języku C++ WebSphere MQ na serwerze Linux przy użyciu kompilatora GNU g + +.

System p

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

Klient: System p

32-bitowa aplikacja wielowątkowa

```
g++ -m32 -o imqsputc_32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl
-limqb23gl -lmqic
```

32-bitowa aplikacja wielowątkowa

```
g++ -m32 -o imqsputc_r32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r
-limqb23gl_r -lmqic_r
```

Aplikacja wielowątkowa z 64-bitową wersją

```
g++ -m64 -o imqsputc_64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Aplikacja wielowątkowa z 64-bitowym

```
g++ -m64 -o imqsputc_r64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r
```

Serwer: System p

32-bitowa aplikacja wielowątkowa

```
g++ -m32 -o imqsput_32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl
-limqb23gl -lmqm
```

32-bitowa aplikacja wielowątkowa

```
g++ -m32 -o imqsput_r32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r
-limqb23gl_r -lmqm_r
```

Aplikacja wielowątkowa z 64-bitową wersją

```
g++ -m64 -o imqsput_64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl -limqb23gl -lmqm
```

Aplikacja wielowątkowa z 64-bitowym

```
g++ -m64 -o imqsput_r64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl_r -limqb23gl_r -lmqm_r
```

System z

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

Klient: System z

32-bitowa aplikacja wielowątkowa

```
g++ -m31 -fsigned-char -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl -limqb23gl -lmqic
```

32-bitowa aplikacja wielowątkowa

```
g++ -m31 -fsigned-char -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r -limqb23gl_r -lmqic_r
-lpthread
```

Aplikacja wielowątkowa z 64-bitową wersją

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Aplikacja wielowątkowa z 64-bitowym

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64  -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Serwer: System z

32-bitowa aplikacja wielowątkowa

```
g++ -m31 -fsigned-char -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib  -Wl,-rpath=/usr/lib
-limqs23gl -limqb23gl -lmqm
```

32-bitowa aplikacja wielowątkowa

```
g++ -m31 -fsigned-char -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

Aplikacja wielowątkowa z 64-bitową wersją

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl -limqb23gl -lmqm
```

Aplikacja wielowątkowa z 64-bitowym

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

System x (32-bitowy)

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

Client: System x (32-bitowy)

32-bitowa aplikacja wielowątkowa

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,
-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqc23gl -limqb23gl -lmqic
```

32-bitowa aplikacja wielowątkowa

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqc23gl_r -limqb23gl_r
-lmqic_r -lpthread
```

Aplikacja wielowątkowa z 64-bitową wersją

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqc23gl -limqb23gl
-lmqic
```

Aplikacja wielowątkowa z 64-bitowym

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqc23gl_r -limqb23gl_r
-lmqic_r -lpthread
```

Serwer: System x (32-bitowy)

32-bitowa aplikacja wielowątkowa

```
g++ -m32 -fsigned-char -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqs23gl -limqb23gl -lmqm
```

32-bitowa aplikacja wielowątkowa

```
g++ -m32 -fsigned-char -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATHlib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqs23gl_r -limqb23gl_r
-lmqm_r -lpthread
```

Aplikacja wielowątkowa z 64-bitową wersją

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqs23gl -limqb23gl -lmqm
```

Aplikacja wielowątkowa z 64-bitowym

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqs23gl_r -limqb23gl_r
-lmqm_r -lpthread
```

Budowanie programów C++ w systemie Solaris

Zbuduj programy w języku C++ WebSphere MQ w systemie Solaris przy użyciu kompilatora Sun ONE.

SPARC

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

Klient: SPARC

Aplikacja 32-bitowa

```
CC -xarch=v8plus -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
```

```
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic -lsocket -lnsl -ldl
```

Aplikacja 64-bitowa

```
CC -xarch=v9 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Serwer: SPARC

Aplikacja 32-bitowa

```
CC -xarch=v8plus -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as
-lmqm -lsocket -lnsl -ldl
```

Aplikacja 64-bitowa

```
CC -xarch=v9 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as
-lmqm -lsocket -lnsl -ldl
```

x86-64

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

Klient: x86-64

Aplikacja 32-bitowa

```
CC -xarch=386 -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic -lsocket -lnsl -ldl
```

Aplikacja 64-bitowa

```
CC -xarch=amd64 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Serwer: x86-64

Aplikacja 32-bitowa

```
CC -xarch=386 -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as
-lmqm -lsocket -lnsl -ldl
```

Aplikacja 64-bitowa

```
CC -xarch=amd64 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as
-lmqm -lsocket -lnsl -ldl
```

Budowanie programów C++ w systemie Windows

Zbuduj programy WebSphere MQ C++ w systemie Windows, korzystając z kompilatora Microsoft Visual Studio C++.

Pliki biblioteki (.lib) i pliki DLL do użycia z aplikacjami 32-bitowymi są zainstalowane w programie *MQ_INSTALLATION_PATH*/Tools/Lib, pliki do użycia z aplikacjami 64-bitowymi są instalowane w programie *MQ_INSTALLATION_PATH*/Tools/Lib64. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .

Klient

```
cl -MD imqsput.cpp /Feimqsputc.exe imqb23vn.lib imqc23vn.lib
```

Serwer

```
 cl -MD imqsput.cpp /Feimqsput.exe imqb23vn.lib imqs23vn.lib
```

Korzystanie z klas produktu WebSphere MQ dla języka Java

Klasy produktu WebSphere MQ dla języka Java umożliwiają korzystanie z produktu WebSphere MQ w środowisku Java. Aplikacja Java może korzystać z klas produktu WebSphere MQ dla języka Java lub klas WebSphere MQ dla usługi JMS w celu uzyskania dostępu do zasobów produktu WebSphere MQ .

Klasy produktu WebSphere MQ dla języka Java umożliwiają aplikacji Java:

- Nawiąże połączenie z produktem WebSphere MQ jako klient WebSphere MQ .
- Nawiąże bezpośrednie połączenie z menedżerem kolejek produktu WebSphere MQ .

Klasy WebSphere MQ dla języka Java obudowują interfejs kolejki komunikatów (Message Queue Interface-MQI), rodzimy interfejs API produktu WebSphere MQ .

Klasy produktu WebSphere MQ dla języka Java używają podobnego modelu obiektowego do interfejsów C++ i .NET w produkcie WebSphere MQ.

Dlaczego warto używać klas produktu WebSphere MQ dla języka Java?

Jeśli w instalacji istotne są następujące punkty, należy rozważyć użycie klas WebSphere MQ dla języka Java:

- Klasy WebSphere MQ dla języka Java obudowują interfejs kolejki komunikatów (Message Queue Interface-MQI), rodzimy interfejs API produktu WebSphere MQ .
  - Użytkownik zaznajomiony z korzystaniem z interfejsu MQI w językach proceduralnych może przekazać tę wiedzę do środowiska Java.
  - Istnieje możliwość wykorzystania pełnego zakresu funkcji produktu WebSphere MQ, poza tymi dostępnymi za pośrednictwem usługi JMS.
- Klasy produktu WebSphere MQ dla języka Java używają podobnego modelu obiektowego do interfejsów C++ i .NET w produkcie WebSphere MQ. Użytkownik zaznajomiony z tymi interfejsami może przekazać tę wiedzę do środowiska Java.

Uwaga: Automatyczne ponowne połączenie klienta nie jest obsługiwane przez klasy produktu WebSphere MQ dla języka Java.

Pierwsze kroki z klasami produktu WebSphere MQ dla języka Java

Ta kolekcja tematów zawiera przegląd klas produktu WebSphere MQ dla języka Java i ich zastosowań.

Czym są klasy produktu WebSphere MQ dla języka Java?

Klasy produktu WebSphere MQ dla języka Java umożliwiają korzystanie z produktu WebSphere MQ w środowisku Java.

Klasy produktu WebSphere MQ dla języka Java umożliwiają aplikacji Java:

- Nawiąże połączenie z produktem WebSphere MQ jako klient WebSphere MQ .
- Nawiąże bezpośrednie połączenie z menedżerem kolejek produktu WebSphere MQ .

Klasy WebSphere MQ dla języka Java obudowują interfejs kolejki komunikatów (Message Queue Interface-MQI), rodzimy interfejs API produktu WebSphere MQ .

Klasy produktu WebSphere MQ dla języka Java używają podobnego modelu obiektowego do interfejsów C++ i .NET w produkcie WebSphere MQ.

Dlaczego warto używać klas produktu WebSphere MQ dla języka Java?

Aplikacja Java może korzystać z klas produktu WebSphere MQ dla języka Java lub klas WebSphere MQ dla usługi JMS w celu uzyskania dostępu do zasobów produktu WebSphere MQ . Istnieje wiele zalet używania klas produktu WebSphere MQ dla języka Java.

Jeśli w danej instalacji istotne są następujące punkty, należy rozważyć użycie klas produktu Websphere MQ dla języka Java:

- Klasy WebSphere MQ dla języka Java obudowują interfejs kolejki komunikatów (Message Queue Interface-MQI), rodzimy interfejs API produktu WebSphere MQ .
  - Użytkownik zaznajomiony z korzystaniem z interfejsu MQI w językach proceduralnych może przekazać tę wiedzę do środowiska Java.
  - Istnieje możliwość wykorzystania pełnego zakresu funkcji produktu WebSphere MQ, poza tymi dostępnymi za pośrednictwem usługi JMS.
- Klasy produktu WebSphere MQ dla języka Java używają podobnego modelu obiektowego do interfejsów C++ i .NET w produkcie WebSphere MQ. Użytkownik zaznajomiony z tymi interfejsami może przekazać tę wiedzę do środowiska Java.

Opcje połączenia dla klas WebSphere MQ dla języka Java

Klasy produktu WebSphere MQ dla języka Java mogą łączyć się w trybie klienta lub powiązań.

Opcje programowalne umożliwiają klasom produktu WebSphere MQ dla języka Java nawiązanie połączenia z produktem WebSphere MQ w jeden z następujących sposobów:

- Jako klient MQI produktu WebSphere MQ korzystający z protokołu TCP/IP (Transmission Control Protocol/Internet Protocol)
- W trybie powiązań, łącząc się bezpośrednio z produktem WebSphere MQ przy użyciu interfejsu JNI (Java Native Interface).

Klienty nie mogą być uruchamiane w systemie z/OS, ale klienty na innych platformach mogą łączyć się z menedżerem kolejek produktu WebSphere MQ for z/OS , jeśli jest zainstalowany mechanizm przyłączenia klienta.

W poniższych sekcjach opisano bardziej szczegółowo opcje połączenia trybu klienta i trybu powiązań.

Połączenie klienta

Aby nawiązać połączenie z menedżerem kolejek w trybie klienta, klasy WebSphere MQ dla aplikacji Java mogą być uruchamiane w tym samym systemie, w którym jest uruchomiony menedżer kolejek, lub w innym systemie. W każdym przypadku klasy WebSphere MQ dla języka Java łączy się z menedżerem kolejek za pośrednictwem protokołu TCP/IP.

Klasy produktu WebSphere MQ dla aplikacji Java mogą łączyć się z dowolnym obsługiwanym menedżerem kolejek za pomocą trybu klienta.

Więcej informacji na temat sposobu pisania aplikacji w celu korzystania z połączeń w trybie klienta zawiera sekcja "Klasy WebSphere MQ dla trybów połączenia Java" na stronie 685.

Połączenie powiązań

W przypadku użycia w trybie powiązań klasy WebSphere MQ dla języka Java korzystają z interfejsu JNI (Java Native Interface) w celu wywołania bezpośrednio do istniejącego interfejsu API menedżera kolejek, zamiast komunikowania się za pośrednictwem sieci. W większości środowisk łączenie w trybie powiązań zapewnia lepszą wydajność dla klas WebSphere MQ dla aplikacji Java niż łączenie w trybie klienta, przez uniknięcie kosztów komunikacji TCP/IP.

Aplikacje, które używają klas produktu WebSphere MQ dla języka Java do nawiązywania połączenia w trybie powiązań, muszą działać w tym samym systemie, co menedżer kolejek, z którym są one nawiązane.

Środowisko Java Runtime Environment, które jest używane do uruchamiania klas WebSphere MQ dla aplikacji Java, musi być skonfigurowane do ładowania klas produktu WebSphere MQ dla bibliotek Java. Aby uzyskać więcej informacji, należy zapoznać się z informacjami w sekcji Klasy WebSphere MQ dla bibliotek Java .

Więcej informacji na temat sposobu pisania aplikacji w celu korzystania z połączeń w trybie powiązań zawiera sekcja "Klasy WebSphere MQ dla trybów połączenia Java" na stronie 685.

Wymagania wstępne dla klas produktu WebSphere MQ dla języka Java

Aby używać klas WebSphere MQ dla języka Java, potrzebne są inne produkty oprogramowania.

Najnowsze informacje na temat wymagań wstępnych dla klas WebSphere MQ dla języka Java można znaleźć w pliku README produktu WebSphere MQ .

Aby utworzyć klasy produktu WebSphere MQ dla aplikacji Java, potrzebny jest pakiet Java Development Kit (JDK). Szczegółowe informacje na temat pakietów JDK obsługiwanych przez system operacyjny można znaleźć na stronie wymagań systemowych produktu WebSphere MQ pod adresem Wymagania systemowe produktu IBM WebSphere MQ.

Aby uruchomić klasy produktu WebSphere MQ dla aplikacji Java, należy użyć następujących komponentów oprogramowania:

- Menedżer kolejek produktu WebSphere MQ dla aplikacji, które łączą się z menedżerem kolejek.
- Środowisko wykonawcze programów Java (Java Runtime Environment-JRE) dla każdego systemu, w którym uruchamiane są aplikacje. Odpowiednie środowisko JRE jest dostarczane z produktem WebSphere MQ.

Jeśli do korzystania z modułów szyfrujących z certyfikatem FIPS 140-2 wymagane są połączenia SSL, wymagany jest dostawca IBM Java JSSE FIPS (IBMJSSEFIPS). Każdy pakiet IBM JDK i JRE w wersji 1.4.2 lub nowszej zawiera IBMJSSEFIPS.

W klasach WebSphere MQ dla aplikacji Java można używać adresów Internet Protocol wersja 6 (IPv6), jeśli IPv6 jest obsługiwany przez wirtualną maszynę języka Java (JVM) i implementację protokołu TCP/IP w systemie operacyjnym.

Instalowanie i konfigurowanie klas produktu WebSphere MQ dla języka Java

W tej sekcji opisano katalogi i pliki, które są tworzone podczas instalowania klas produktu WebSphere MQ dla języka Java, a także wyjaśniono sposób konfigurowania klas WebSphere MQ dla języka Java po instalacji.

Co jest zainstalowane w przypadku klas produktu WebSphere MQ dla języka Java

Najnowsza wersja klas produktu WebSphere MQ dla języka Java jest instalowana razem z produktem WebSphere MQ. Może być konieczne przesłonięcie domyślnych opcji instalacji w celu upewniania się, że jest to wykonane.

Więcej informacji na temat instalowania produktu WebSphere MQ zawierają następujące informacje:

Instalowanie serwera WebSphere MQ
Instalowanie klienta IBM WebSphere MQ

Klasy WebSphere MQ dla języka Java są zawarte w plikach archiwum Java (JAR), com.ibm.mq.jari com.ibm.mq.jmqi.jar.

Obsługa standardowych nagłówków komunikatów, takich jak programowalny format komend (Programmable Command Format-PCF), znajduje się w pliku JAR com.ibm.mq.headers.jar.

Obsługa formatu programu Programmable Command Format (PCF) jest zawarta w pliku JAR com.ibm.mq.pcf.jar.

Instalowanie i aktualizowanie klas produktu WebSphere MQ dla plików JAR Java

Jedynym obsługiwanym sposobem pobrania klas produktu WebSphere MQ na potrzeby plików JAR Java w systemie jest zainstalowanie produktu WebSphere MQ lub klienta MQI produktu WebSphere MQ SupportPaclub użycie narzędzia do zarządzania oprogramowaniem, takiego jak Apache Maven, aby uzyskać więcej informacji na temat “Narzędzia IBM WebSphere MQ classes for Java i narzędzia do zarządzania oprogramowaniem” na stronie 680.

Nie należy przenosić ani kopiować klas produktu WebSphere MQ dla plików JAR Java z innych komputerów, chyba że używane jest narzędzie do zarządzania oprogramowaniem.

- Pakietów poprawek nie można zastosować do "instalacji", w której pliki JAR zostały skopiowane z innego komputera, co znacznie utrudnia zapewnienie, że wszystkie pliki JAR są przechowywane razem ze sobą, i są na kompatybilnych poziomach.
- Skopiowanie klas WebSphere MQ dla plików JAR JMS między maszynami może również spowodować wiele kopii plików znajdujących się na tym samym komputerze, co może powodować problemy z obsługą kodu i debugowaniem problemów.

Nie należy uwzględniać klas produktu WebSphere MQ dla plików JAR Java w archiwach aplikacji.

- Aktualizacje klas produktu WebSphere MQ dla języka Java nie mogą być stosowane przy użyciu pakietu poprawek WebSphere MQ .
- Nie jest możliwe, aby dział wsparcia IBM mógł łatwo określić wersję klas WebSphere MQ dla języka Java, które są używane przez aplikację.
- Mogą wystąpić problemy, jeśli wiele aplikacji działających w tym samym środowisku Java Runtime Environment obejmuje różne wersje klas produktu WebSphere MQ dla języka Java, ponieważ wiele wersji klas produktu WebSphere MQ dla języka Java jest ładowanych do środowiska wykonawczego programów Java jednocześnie.
- Jeśli aplikacja korzysta z transportu BINDINGS w celu nawiązania połączenia z menedżerem kolejek, wszystkie główne aktualizacje menedżera kolejek wymagają zaktualizowania aplikacji tak, aby uwzględniała odpowiedni poziom klas produktu WebSphere MQ dla języka Java.

  Jeśli na przykład menedżer kolejek jest aktualizowany do wersji WebSphere MQ 7.1 , to wszystkie aplikacje, które łączą się z menedżerem kolejek przy użyciu transportu BINDINGS, również muszą zostać zaktualizowane w celu uwzględnienia klas produktu WebSphere MQ w wersji 7.1 dla języka Java.

Następująca biblioteka Java jest dystrybuowana z klasami produktu WebSphere MQ dla języka Java:

- connector.jar (wersja 1.0)

Przykładowa aplikacja o nazwie Postcard znajduje się w pliku JAR com.ibm.mq.postcard.jar.

Narzędzie Javadoc zostało użyte do wygenerowania stron HTML zawierających specyfikacje klas WebSphere MQ dla języka Java i klas WebSphere MQ dla interfejsów API JMS. Strony HTML znajdują się w podkatalogu doc w katalogu instalacyjnym klas WebSphere MQ dla katalogu instalacyjnego JMS. W systemach UNIX, Linuxi Windows podkatalog doc zawiera poszczególne strony HTML.

Po zakończeniu instalacji pliki i przykłady są instalowane w położeniach wyświetlanych w produkcie “Katalogi instalacyjne dla klas produktu WebSphere MQ dla języka Java” na stronie 673.

Po zakończeniu instalacji na dowolnej platformie innej niż Windowsnależy zaktualizować zmienne środowiskowe zgodnie z opisem w sekcji “Zmienne środowiskowe istotne dla klas WebSphere MQ dla języka Java” na stronie 673.

Katalogi instalacyjne dla klas produktu WebSphere MQ dla języka Java

Klasy WebSphere MQ dla plików Java są instalowane w różnych miejscach zgodnie z platformą.

Tabela 82 na stronie 673 pokazuje, gdzie zainstalowane są klasy WebSphere MQ dla plików Java.

Tabela 82. Klasy WebSphere MQ dla katalogów instalacyjnych Java

Platforma	**Katalog**
AIX	*MQ_INSTALLATION_PATH*/java/lib
HP-UX, Linuxi Solaris	*MQ_INSTALLATION_PATH*/java/lib
Windows	*MQ_INSTALLATION_PATH*\java\lib
MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .	

Niektóre aplikacje przykładowe, takie jak programy weryfikacji instalacji (Installation Verification Programs-IVPs), są dostarczane razem z produktem WebSphere MQ. Tabela 83 na stronie 673 pokazuje, gdzie zainstalowane są przykładowe aplikacje. Klasy produktu WebSphere MQ dla przykładów Java znajdują się w podkatalogu o nazwie `wmqjava`. Przykłady PCF znajdują się w podkatalogu o nazwie `pcf`.

Tabela 83. Katalogi przykładów

Platforma	**Katalog**
AIX	*MQ_INSTALLATION_PATH*/samp/wmqjava/
HP-UX, Linuxi Solaris	*MQ_INSTALLATION_PATH*/samp/wmqjava/
Windows	*MQ_INSTALLATION_PATH*\tools\wmqjava\
MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .	

Zmienne środowiskowe istotne dla klas WebSphere MQ dla języka Java

Aby uruchomić klasy produktu WebSphere MQ dla aplikacji Java, ścieżki klas muszą zawierać klasy produktu WebSphere MQ dla języka Java i katalogi przykładów.

W przypadku klas produktu WebSphere MQ dla aplikacji Java do uruchomienia ich ścieżka klasy musi zawierać odpowiednie klasy produktu WebSphere MQ dla katalogu Java. Aby uruchomić przykładowe aplikacje, ścieżka klasy musi także zawierać odpowiednie katalogi przykładów. Te informacje można podać w komendzie wywołania Java lub w zmiennej środowiskowej CLASSPATH.

Tabela 84 na stronie 674 przedstawia odpowiednie ustawienie zmiennej CLASSPATH, które ma być używane na każdej platformie do uruchamiania klas produktu WebSphere MQ dla aplikacji Java, w tym przykładowych aplikacji.

Tabela 84. Ustawienie CLASSPATH w celu uruchomienia klas produktu WebSphere MQ dla aplikacji Java, w tym klas produktu WebSphere MQ dla przykładowych aplikacji Java.	
Platforma	**Ustawienie CLASSPATH**
AIX	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
HP-UX, Linuxi Solaris	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
Windows	CLASSPATH=*MQ_INSTALLATION_PATH*\Java\lib\com.ibm.mq.jar; *MQ_INSTALLATION_PATH*\tools\wmqjava\samples;
MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .	

W przypadku kompilacji za pomocą opcji -Xlint może zostać wyświetlony komunikat z ostrzeżeniem, że plik com.ibm.mq.ese.jar nie jest obecny. Ostrzeżenie można zignorować. Ten plik jest dostępny tylko wtedy, gdy zainstalowano produkt IBM WebSphere MQ Advanced Message Security.

Skrypty dostarczane z klasami produktu WebSphere MQ dla języka Java korzystają z następujących zmiennych środowiskowych:

MQ_JAVA_DATA_PATH
Ta zmienna środowiskowa określa katalog dla danych wyjściowych dziennika i śledzenia.

MQ_JAVA_INSTALL_PATH
Ta zmienna środowiskowa określa katalog, w którym są zainstalowane klasy WebSphere MQ dla języka Java, jak to pokazano w sekcji Klasy produktu WebSphere MQ dla katalogów instalacyjnych Java.

MQ_JAVA_LIB_PATH
Ta zmienna środowiskowa określa katalog, w którym są przechowywane klasy produktu WebSphere MQ dla bibliotek Java, jak to pokazano w sekcji Położenie klas produktu WebSphere MQ dla bibliotek Java dla każdej platformy. Niektóre skrypty dostarczane z klasami produktu WebSphere MQ dla języka Java, takimi jak IVTRun, używają tej zmiennej środowiskowej.

W systemie Windowswszystkie zmienne środowiskowe są ustawiane automatycznie podczas instalacji. Na każdej innej platformie, musisz ustawić je samodzielnie.W systemie UNIX można użyć skryptu **setjmsenv** (jeśli używana jest 32-bitowa maszyna JVM) lub **setjmsenv64** (jeśli używana jest 64-bitowa maszyna JVM) do ustawienia zmiennych środowiskowych. W systemach AIX, HP-UX, Linuxi Solaris te skrypty znajdują się w katalogu *MQ_INSTALLATION_PATH*/java/bin .

Klasy IBM WebSphere MQ dla bibliotek Java

Położenie klas IBM WebSphere MQ dla bibliotek Java jest różne w zależności od platformy. Tę lokalizację należy określić podczas uruchamiania aplikacji.

Aby określić położenie bibliotek JNI (Java Native Interface), należy uruchomić aplikację za pomocą komendy **java** z następującym formatem:

```
java -Djava.library.path=library_path application_name
```

gdzie *ścieżka_biblioteki* jest ścieżką do klas produktu WebSphere MQ dla bibliotek Java, które obejmują biblioteki JNI. Tabela 85 na stronie 675 Wyświetla położenie klas WebSphere MQ dla bibliotek Java dla każdej platformy.

Tabela 85. Położenie klas produktu WebSphere MQ dla bibliotek Java dla każdej platformy	
Platforma	**Katalog zawierający klasy produktu WebSphere MQ dla bibliotek Java**
AIX	*MQ_INSTALLATION_PATH*/java/lib (biblioteki 32-bitowe) *MQ_INSTALLATION_PATH*/java/lib64 (biblioteki 64-bitowe)
HP-UX Linux (POWER, x86-64 i Platformy zSeries s390x) Solaris (platformyx86-64 i SPARC)	*MQ_INSTALLATION_PATH*/java/lib (biblioteki 32-bitowe) *MQ_INSTALLATION_PATH*/java/lib64 (biblioteki 64-bitowe)
Linux (platformax86)	*MQ_INSTALLATION_PATH*/java/lib
Windows	*MQ_INSTALLATION_PATH*\Java\lib (biblioteki 32-bitowe) *MQ_INSTALLATION_PATH*\Java\lib64 (biblioteki 64-bitowe)
MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .	

Uwaga:

1. W systemach AIX, HP-UX, Linux (platforma zasilania) lub Solaris należy użyć 32-bitowych bibliotek lub 64-bitowych bibliotek. Biblioteki 64-bitowej należy używać tylko wtedy, gdy aplikacja jest uruchamiana w 64-bitowej wirtualnej maszynie języka Java (JVM) na platformie 64-bitowej. W przeciwnym razie należy użyć 32-bitowych bibliotek.
2. W systemie Windowsmożna użyć zmiennej środowiskowej PATH w celu określenia położenia klas produktu WebSphere MQ dla bibliotek Java zamiast określania ich położenia w komendzie **java** .
3. Aby używać klas produktu WebSphere MQ dla języka Java w trybie powiązań w systemie IBM i, należy upewnić się, że biblioteka QMQMJAVA znajduje się na liście bibliotek.

Zadania pokrewne

Korzystanie z klas produktu WebSphere MQ dla języka Java

Obsługa środowiska OSGi w systemie IBM WebSphere MQ classes for Java

Środowisko OSGi udostępnia środowisko, które obsługuje wdrażanie aplikacji w postaci pakunków. Jeden pakunek OSGi jest dostarczany jako część produktu IBM WebSphere MQ classes for Java .

Środowisko OSGi udostępnia ogólne, bezpieczne i zarządzane środowisko produktuJava , które obsługuje wdrażanie aplikacji, które wchodzą w postaci pakunków. Urządzenia zgodne ze środowiskiem OSGi mogą pobierać i instalować pakunki, a także usuwać je, gdy nie są już wymagane. Środowisko zarządza instalacją i aktualizacją pakunków w sposób dynamiczny i skalowalny.

IBM WebSphere MQ classes for Java. zawiera następujący pakunek OSGi.

com.ibm.mq.osgi.java_< numer wersji > .jar
Pliki JAR, które umożliwiają aplikacjom korzystanie z IBM WebSphere MQ classes for Java.

gdzie < numer wersji > jest numerem wersji produktu WebSphere MQ , który został zainstalowany.

Pakunek jest instalowany w podkatalogu `java/lib/OSGi` instalacji produktu IBM WebSphere MQ lub w folderze `java\lib\OSGi` w systemie Windows.

Dziewięć innych pakunków jest również zainstalowanych w podkatalogu `java/lib/OSGi` instalacji produktu IBM WebSphere MQ lub w folderze `java\lib\OSGi` w systemie Windows. Pakunki te są częścią produktu IBM WebSphere MQ classes for JMS i nie mogą być ładowane do środowiska wykonawczego OSGi, które ma załadowany pakunek IBM WebSphere MQ classes for Java . Jeśli pakunek OSGi produktu IBM WebSphere MQ classes for Java jest ładowany do środowiska wykonawczego OSGi, które zawiera również załadowane pakunki IBM WebSphere MQ classes for JMS , błędy takie jak:

```
java.lang.ClassCastException: com.ibm.mq.MQException incompatible with com.ibm.mq.MQException
```

występują, gdy uruchamiane są aplikacje korzystające z pakunku IBM WebSphere MQ classes for Java lub pakunków IBM WebSphere MQ classes for JMS .

Pakunek OSGi dla produktu IBM WebSphere MQ classes for Java został zapisany w specyfikacji OSGi Release 4. Nie działa on w środowisku OSGi Release 3.

Należy poprawnie ustawić ścieżkę systemową lub ścieżkę do biblioteki, aby środowisko wykonawcze OSGi było w stanie znaleźć wszystkie wymagane pliki DLL lub współużytkowane biblioteki.

Jeśli używany jest pakunek OSGi dla produktu IBM WebSphere MQ classes for Java, klasy wyjścia kanału napisane w produkcie Java nie są obsługiwane z powodu nieodłącznego problemu w ładowaniu klas w środowisku programu ładującego wiele klas, takim jak środowisko OSGi. Pakunek użytkownika może być świadomy pakunku produktu IBM WebSphere MQ classes for Java , ale pakunek IBM WebSphere MQ classes for Java nie ma informacji o pakunku użytkownika. W wyniku tego program ładujący klasy używany w pakunku IBM WebSphere MQ classes for Java nie może załadować klasy wyjścia kanału, która znajduje się w pakunku użytkownika.

Więcej informacji na temat środowiska OSGi można znaleźć w serwisie WWW OSGi alliance .

Plik konfiguracyjny IBM WebSphere MQ classes for Java

Plik konfiguracyjny IBM WebSphere MQ classes for Java określa właściwości, które są używane do konfigurowania IBM WebSphere MQ classes for Java.

Format pliku konfiguracyjnego produktu IBM WebSphere MQ classes for Java jest taki sam jak standardowy plik właściwości produktu Java .

V 7.5.0.9 Z poziomu produktu IBM WebSphere MQ Version 7.5.0, pakiet poprawek 9przykładowy plik konfiguracyjny o nazwie `mqjava.config` jest dostarczany w podkatalogu `bin` katalogu instalacyjnego produktu IBM WebSphere MQ classes for Java . Ten plik dokumentuje wszystkie obsługiwane właściwości i ich wartości domyślne.

Uwaga: Przykładowy plik konfiguracyjny jest nadpisywany, gdy instalacja produktu IBM WebSphere MQ zostanie zaktualizowana do przyszłego pakietu poprawek. Z tego powodu zaleca się, aby utworzyć kopię przykładowego pliku konfiguracyjnego do użycia z aplikacjami.

Użytkownik może wybrać nazwę i położenie pliku konfiguracyjnego produktu IBM WebSphere MQ classes for Java . Po uruchomieniu aplikacji należy użyć komendy **java** z następującym formatem:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

W komendzie *config_file_url* jest jednostajnym lokalizatorem zasobów (URL), który określa nazwę i położenie pliku konfiguracyjnego IBM WebSphere MQ classes for Java . Obsługiwane są adresy URL następujących typów: `http`, `file`, `ftp`i `jar`.

W poniższym przykładzie przedstawiono komendę **java** :

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/mqjava.config MyAppClass
```

Ta komenda identyfikuje plik konfiguracyjny IBM WebSphere MQ classes for Java jako plik `D:\mydir\mqjava.config` w lokalnym systemie Windows .

Plik konfiguracyjny IBM WebSphere MQ classes for Java może być używany z dowolnym z obsługiwanych transportów między aplikacją a menedżerem kolejek lub brokerem.

Nadpisywanie właściwości określonych w pliku konfiguracyjnym IBM WebSphere MQ classes for Java

Plik konfiguracyjny IBM WebSphere MQ MQI client może również określać właściwości, które są używane do konfigurowania produktu IBM WebSphere MQ classes for Java. Jednak właściwości, które są określone w pliku konfiguracyjnym IBM WebSphere MQ MQI client , mają zastosowanie tylko wtedy, gdy aplikacja łączy się z menedżerem kolejek w trybie klienta.

Jeśli jest to wymagane, można przesłonić dowolny atrybut w pliku konfiguracyjnym IBM WebSphere MQ MQI client , określając go jako właściwość w pliku konfiguracyjnym IBM WebSphere MQ classes for Java . Aby przesłonić atrybut w pliku konfiguracyjnym IBM WebSphere MQ MQI client , należy użyć wpisu o następującym formacie w pliku konfiguracyjnym IBM WebSphere MQ classes for Java :

```
com.ibm.mq.cfg.stanza.propName=propValue
```

Zmienne w pozycji mają następujące znaczenia:

sekcja
Nazwa sekcji w pliku konfiguracyjnym IBM WebSphere MQ MQI client , który zawiera atrybut.

propName
Nazwa atrybutu określona w pliku konfiguracyjnym IBM WebSphere MQ MQI client .

propValue
Wartość właściwości, która nadpisuje wartość atrybutu określonego w pliku konfiguracyjnym IBM WebSphere MQ MQI client .

Alternatywnie można przesłonić atrybut w pliku konfiguracyjnym IBM WebSphere MQ MQI client , określając właściwość jako właściwość systemową w komendzie **java** . Aby określić właściwość jako właściwość systemową, należy użyć poprzedniego formatu.

Tylko następujące atrybuty w pliku konfiguracyjnym IBM WebSphere MQ MQI client są istotne dla produktu IBM WebSphere MQ classes for Java. Jeśli zostaną określone lub nadpisane inne atrybuty, nie będzie to miało żadnego wpływu. W szczególności należy pamiętać, że `ChannelDefinitionFile` i `ChannelDefinitionDirectory` w sekcji CHANNELS w pliku konfiguracyjnym klienta nie są używane. Szczegółowe informacje na temat korzystania z tabeli definicji kanału klienta z IBM WebSphere MQ classes for Javazawiera sekcja “Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM WebSphere MQ classes for Java” na stronie 689 .

Tabela 86. Która sekcja pliku konfiguracyjnego klienta zawiera atrybut, który atrybut

sekcja	Atrybut
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	ExitsDefaultPath
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	ExitsDefaultPath64
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	Ścieżka klasy JavaExits
Sekcja MessageBuffer pliku konfiguracyjnego klienta	MaximumSize
Sekcja MessageBuffer pliku konfiguracyjnego klienta	PurgeTime
Sekcja MessageBuffer pliku konfiguracyjnego klienta	UpdatePercentage
Sekcja TCP pliku konfiguracyjnego klienta	ClntRcvBufSize
Sekcja TCP pliku konfiguracyjnego klienta	ClntSndBufSize

Tabela 86. Która sekcja pliku konfiguracyjnego klienta zawiera atrybut, który atrybut (kontynuacja)	
sekcja	**Atrybut**
Sekcja TCP pliku konfiguracyjnego klienta	Limit_czasu_połączenia
Sekcja TCP pliku konfiguracyjnego klienta	KeepAlive

Więcej informacji na temat konfigurowania produktu IBM WebSphere MQ MQI client zawiera sekcja Konfigurowanie klienta przy użyciu pliku konfiguracyjnego.

Zadania pokrewne

Śledzenie aplikacji IBM WebSphere MQ classes for Java

Sekcja Standardowe śledzenie środowiska Java

Aby skonfigurować narzędzie śledzenia IBM WebSphere MQ classes for Java , można użyć sekcji Standard Trace Settings (Ustawienia śledzenia środowiska) produktu Java .

com.ibm.msg.client.commonservices.trace.outputName = `traceOutputName`

`traceOutputName` to katalog i nazwa pliku, do którego wysyłane są dane wyjściowe śledzenia.

Domyślna nazwa pliku śledzenia zależy od wersji produktu IBM WebSphere MQ classes for Java , która jest używana przez aplikację:

- W przypadku produktu IBM WebSphere MQ classes for Java dla systemu Version 7.5.0, Fix Pack 8 lub wcześniejszej wartości *traceOutputName* domyślnie jest to plik o nazwie `mqjms_%PID%.trc` w bieżącym katalogu roboczym.
- V 7.5.0.9 Od IBM WebSphere MQ classes for Java z Version 7.5.0, Fix Pack 9, *traceOutputName* przyjmuje wartości domyślne do pliku o nazwie `mqjava_%PID%.trc` w bieżącym katalogu roboczym.

gdzie *%PID%* jest bieżącym identyfikatorem procesu. Jeśli identyfikator procesu jest niedostępny, generowany jest numer losowy, który jest poprzedzony literą `f`. Aby dołączyć identyfikator procesu do określonej nazwy pliku, należy użyć łańcucha `%PID%` .

Jeśli zostanie podany katalog alternatywny, musi on istnieć, a użytkownik musi mieć uprawnienia do zapisu dla tego katalogu. Jeśli użytkownik nie ma uprawnień do zapisu, dane wyjściowe śledzenia są zapisywane w produkcie `System.err`.

com.ibm.msg.client.commonservices.trace.include = `includeList`

`includeList` to lista śledzonych pakietów i klas albo wartości specjalne `ALL` lub `NONE`.

Oddziel nazwy pakietów lub klas średnikami (`;`). **`includeList`** wartością domyślną jest `ALL`, a wszystkie pakiety i klasy są śledzane w produkcie IBM WebSphere MQ classes for Java.

Uwaga: Możliwe jest dołączenie pakietu, a następnie wykluczenie podpakietów tego pakietu. For example, if you include package `a.b` and exclude package `a.b.x`, the trace includes everything in `a.b.y` and `a.b.z`, but not `a.b.x` or `a.b.x.1`.

com.ibm.msg.client.commonservices.trace.exclude = `excludeList`

`excludeList` to lista pakietów i klas, które nie są śledzone, lub wartości specjalne `ALL` lub `NONE`.

Oddziel nazwy pakietów lub klas średnikami (`;`). Parametr **`excludeList`** przyjmuje wartość domyślną `NONE`, dlatego nie jest wykluczany żaden z pakietów i klas w produkcie IBM WebSphere MQ classes for Java .

Uwaga: Użytkownik może wykluczyć pakiet, ale następnie uwzględnić podpakiety tego pakietu. Jeśli na przykład pakiet `a.b` zostanie wykluczony i dołączany jest pakiet `a.b.x`, dane śledzenia będą zawierały wszystkie elementy w produkcie `a.b.x` i `a.b.x.1`, ale nie będą zawierać wartości `a.b.y` ani `a.b.z`.

Uwzględniane są wszystkie pakiety lub klasy, które są określone na tym samym poziomie, co zarówno włączone, jak i wykluczone.

com.ibm.msg.client.commonservices.trace.maxBytes = maxArrayBytes
maxArrayBytes to maksymalna liczba bajtów, które są śledzone z dowolnej tablicy bajtów.

Jeśli parametr **maxArrayBytes** jest ustawiony na dodatnią liczbę całkowitą, ogranicza liczbę bajtów w tablicy bajtów, które są zapisywane w pliku śledzenia. Powoduje obcięcie tablicy bajtów po zapisaniu w polu *maxArraybajtów* . Ustawienie **maxArrayBytes** redukuje wielkość wynikowego pliku śledzenia i zmniejsza wpływ śledzenia na wydajność aplikacji.

Wartość `0` dla tej właściwości oznacza, że żadna z treści żadnej tablicy bajtów nie jest wysyłana do pliku śledzenia.

Wartością domyślną jest `-1`, co powoduje usunięcie dowolnego limitu liczby bajtów w tablicy bajtów, które są wysyłane do pliku śledzenia.

com.ibm.msg.client.commonservices.trace.limit = maxTraceBytes
maxTraceBytes to maksymalna liczba bajtów zapisanych w pliku wyjściowym śledzenia.

Produkt **maxTraceBytes** współpracuje z produktem **traceCycles**. Jeśli liczba zapisanych bajtów śledzenia znajduje się w pobliżu limitu, plik jest zamykany, a nowy plik wyjściowy śledzenia jest uruchamiany.

Wartość `0` oznacza, że plik wyjściowy śledzenia ma zerową długość. Wartością domyślną jest `-1`, co oznacza, że ilość danych, które mają zostać zapisane w pliku wyjściowym śledzenia, jest nieograniczona.

com.ibm.msg.client.commonservices.trace.count = traceCycles
traceCycles to liczba plików wyjściowych śledzenia, przez które mają zostać przełączone dane wyjściowe.

Jeśli bieżący plik wyjściowy śledzenia osiągnie limit określony przez program **maxTraceBytes**, plik jest zamykany. Dalsze dane wyjściowe śledzenia są zapisywane w kolejnym pliku wyjściowym śledzenia w kolejności. Każdy plik wyjściowy śledzenia jest wyróżniany przyrostkiem liczbowym, który jest dodawany do nazwy pliku. Bieżący lub najnowszy plik wyjściowy śledzenia ma przyrostek `.trc.0`, następny najnowszy plik wyjściowy śledzenia kończy się na `.trc.1`itd. Starsze pliki śledzenia są zgodne z tym samym wzorcem numeracji aż do limitu.

Wartością domyślną parametru **traceCycles** jest 1. Jeśli parametr **traceCycles** ma wartość 1, wówczas gdy bieżący plik wyjściowy śledzenia osiągnie maksymalną wielkość, plik jest zamykany i usuwany. Zostanie uruchomiony nowy plik wyjściowy śledzenia o tej samej nazwie. Oznacza to, że w danym momencie istnieje tylko jeden plik wyjściowy śledzenia.

com.ibm.msg.client.commonservices.trace.parameter = traceParameters
traceParameters określa, czy parametry metody i wartości zwracane są uwzględniane w śledzeniu.

Wartością domyślną **traceParameters** jest TRUE. Jeśli parametr **traceParameters** jest ustawiony na wartość FALSE, śledzone są tylko sygnatury metod.

com.ibm.msg.client.commonservices.trace.compress = compressedTrace
Ustaw parametr **compressedTrace** na wartość TRUE , aby skompresować dane wyjściowe śledzenia.

Wartością domyślną parametru **compressedTrace** jest FALSE.

Jeśli parametr **compressedTrace** jest ustawiony na wartość TRUE, dane wyjściowe śledzenia są kompresowane. Domyślna nazwa pliku wyjściowego śledzenia ma rozszerzenie `.trz`. Jeśli dla kompresji ustawiona jest wartość FALSE, wartość domyślna, plik ma rozszerzenie `.trc` , aby wskazać, że jest nieskompresowana. Jeśli jednak nazwa pliku dla danych wyjściowych śledzenia jest określona w polu **traceOutputName** , nazwa ta jest używana, a do tego pliku nie jest stosowany żaden przyrostek.

Skompresowane dane wyjściowe śledzenia są mniejsze niż nieskompresowane. Ponieważ istnieje mniej operacji we/wy, można je zapisać szybciej niż nieskompresowane śledzenie. Śledzenie skompresowane ma mniejszy wpływ na wydajność partycji IBM WebSphere MQ classes for Java niż nieskompresowane śledzenie.

Jeśli ustawione są opcje **maxTraceBytes** i **traceCycles** , to w miejsce wielu plików tekstowych tworzone są wiele skompresowanych plików śledzenia.

Jeśli IBM WebSphere MQ classes for Java zakończy się w sposób niekontrolowany, skompresowany plik śledzenia może nie być poprawny. Z tego powodu kompresja śledzenia musi być używana tylko wtedy, gdy IBM WebSphere MQ classes for Java jest zamykany w kontrolowany sposób. Kompresja śledzenia jest używana tylko wtedy, gdy analizowane problemy nie powodują nieoczekiwanego zatrzymania maszyny JVM. Nie należy używać kompresji śledzenia podczas diagnozowania problemów, które mogą spowodować zamknięcie systemu `System.Halt()` lub nieprawidłowe, niekontrolowane zakończenie działania maszyny JVM.

com.ibm.msg.client.commonservices.trace.level = `traceLevel`

`traceLevel` określa poziom filtrowania dla śledzenia. Zdefiniowane poziomy śledzenia są następujące:

Tabela 87. To, co jest śledzone dla każdego poziomu śledzenia

Wartość	Co jest śledzone
0	Śledzenie jest wyłączone
1	Wyjątki
3	Wyjątki Ostrzeżenia
6	Wyjątki Ostrzeżenia Informacyjne punkty śledzenia
8	Wyjątki Ostrzeżenia Informacyjne punkty śledzenia Wejście i wyjście metod
9	Wyjątki Ostrzeżenia Informacyjne punkty śledzenia Wejście i wyjście metod Dane wysyłane między programem IBM WebSphere MQ classes for Java i menedżerem kolejek.

Uwaga: Zawsze należy używać wartości 9 , o ile nie jest to inaczej kierowane przez dział wsparcia produktu IBM .

Narzędzia IBM WebSphere MQ classes for Java i narzędzia do zarządzania oprogramowaniem

Narzędzia do zarządzania oprogramowaniem, takie jak Apache Maven, mogą być używane razem z produktem IBM WebSphere MQ classes for Java.

Wiele dużych organizacji programistycznych używa tych narzędzi do centralnego zarządzania repozytoriami bibliotek innych firm.

IBM WebSphere MQ classes for Java składa się z wielu plików JAR. Podczas tworzenia aplikacji w języku Java za pomocą tego interfejsu API, na komputerze, na którym aplikacja jest rozwijana, wymagana jest instalacja serwera IBM WebSphere MQ , klienta IBM WebSphere MQ lub klienta IBM WebSphere MQ SupportPac .

Aby użyć narzędzia do zarządzania oprogramowaniem i dodać pliki JAR, które tworzą IBM WebSphere MQ classes for Java w repozytorium zarządzanym centralnie, należy zaobserwować następujące punkty:

- Repozytorium lub kontener muszą być dostępne tylko dla programistów w organizacji. Dowolna dystrybucja poza organizacją jest niedozwolona.
- Repozytorium musi zawierać kompletny i spójny zestaw plików JAR z pojedynczego wydania produktu IBM WebSphere MQ lub pakietu poprawek.
- Użytkownik jest odpowiedzialny za aktualizację repozytorium za pomocą obsługi technicznej udostępnianej przez dział wsparcia IBM .

W przypadku produktu IBM WebSphere MQ Version 7.5do repozytorium muszą zostać zainstalowane następujące pliki JAR:

- com.ibm.mq.commonservices.jar
- com.ibm.mq.jar
- com.ibm.mq.jmqi.jar
- com.ibm.mq.pcf.jar
- com.ibm.mq.headers.jar
- connector.jar

Konfigurowanie po instalacji dla aplikacji produktu IBM WebSphere MQ

Po zainstalowaniu produktu IBM WebSphere MQmożna skonfigurować instalację w taki sposób, aby uruchamiała własne aplikacje.

Należy pamiętać, aby sprawdzić plik README produktu IBM WebSphere MQ w celu późniejszego lub bardziej szczegółowego pliku informacji dla używanego środowiska.

Przed podjęciem próby uruchomienia klas IBM WebSphere MQ dla aplikacji Java w trybie powiązań należy upewnić się, że skonfigurowano produkt IBM WebSphere MQ zgodnie z opisem w sekcji Konfigurowanie .

Konfigurowanie menedżera kolejek w celu akceptowania połączeń klientów z klas produktu WebSphere MQ dla języka Java

Aby skonfigurować menedżer kolejek w taki sposób, aby akceptować przychodzące żądania połączeń od klientów, należy zdefiniować i zezwolić na użycie kanału połączenia z serwerem i uruchomić program nasłuchujący.

Szczegółowe informacje na ten temat zawiera sekcja “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113.

Uruchamianie klas produktu WebSphere MQ dla aplikacji Java w ramach menedżera zabezpieczeń Java

Klasy produktu WebSphere MQ dla języka Java mogą być uruchamiane z włączoną obsługą menedżera zabezpieczeń Java. Aby pomyślnie uruchomić aplikacje z włączoną obsługą programu Security Manager, należy skonfigurować wirtualną maszynę języka Java (JVM) z odpowiednim plikiem definicji strategii.

Najprostszym sposobem na to jest zmiana pliku strategii dostarczonego ze środowiskiem JRE. W większości systemów plik ten jest przechowywany w ścieżce `lib/security/java.policy`, w porównaniu do katalogu JRE. Pliki strategii można edytować, korzystając z preferowanego edytora lub programu `policytool` dostarczonego ze środowiskiem JRE.

Należy nadać uprawnienia do pliku com.ibm.mq.jmqi.jar , tak aby mógł on:

- Tworzenie gniazd (w trybie klienta)
- Załaduj bibliotekę rodzimą (w trybie powiązań)
- Odczytywanie różnych właściwości z poziomu środowiska

Właściwość systemowa os.name musi być dostępna dla klas produktu WebSphere MQ dla języka Java podczas działania w ramach menedżera zabezpieczeń Java.

Poniżej przedstawiono przykład wpisu pliku strategii, który umożliwia pomyślne uruchomienie klas produktu WebSphere MQ dla języka Java w ramach domyślnego menedżera zabezpieczeń:

```
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.jmqi.jar" {
  //Required
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  //Required if mqclient.ini/mqs.ini configuration files are used
  permission java.io.FilePermission "/var/mqm/mqclient.ini","read";
  permission java.io.FilePermission "/var/mqm/mqs.ini","read";
  //For the client transport type.
  permission java.net.SocketPermission "*","connect";
  //For the bindings transport type.
  permission java.lang.RuntimePermission "loadLibrary.*";
  //For applications that use CCDT tables (access to the CCDT
AMQCLCHL.TAB)
  permission java.io.FilePermission
"/var/mqm/qmgrs/QMGR/@ipcc/AMQCLCHL.TAB","read";
  //For applications that use User Exits
  permission java.io.FilePermission "/var/mqm/exits/*","read";
  permission java.lang.RuntimePermission "createClassLoader";
  //Required for the z/OS platform
  permission java.util.PropertyPermission
"com.ibm.vm.bitmode","read";
};
grant codeBase
"file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.commonservices.jar" {
  permission java.util.PropertyPermission "user.dir","read";
  permission java.util.PropertyPermission "line.separator","read";
  //tracing permissions
  permission java.util.PropertyPermission "com.ibm.mq.commonservices", "read";
  permission java.util.logging.LoggingPermission "control";
  //For access to the trace properties file.
  permission java.io.FilePermission "/tmp/trace.properties", "read";
  //For access to the trace output files.
  permission java.io.FilePermission "/tmp/*", "read,write";
};
```

Uwagi:

- *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .
- Ten przykład pliku strategii umożliwia poprawne działanie klas produktu WebSphere MQ dla języka Java w ramach menedżera zabezpieczeń, ale nadal konieczne może być włączenie własnego kodu w celu poprawnego działania przed pracą aplikacji.
- Aby umożliwić klasy WebSphere MQ dla języka Java w celu uzyskania dostępu do plików archiwum Java (JAR) aplikacji, dodaj następujące uprawnienie do pierwszej instrukcji `grant` :

  ```
  permission java.io.FilePermission "/path_to_your_app/-", "read";
  ```

- Aby użyć tych instrukcji `grant` w pliku konfiguracyjnym strategii, może być konieczne zmodyfikowanie nazw ścieżek w zależności od miejsca, w którym zainstalowano klasy produktu WebSphere MQ dla języka Java, a także miejsca, w których są przechowywane aplikacje.
- Przykładowy kod dostarczany z klasami produktu WebSphere MQ dla języka Java nie został specjalnie włączony do użycia z menedżerem zabezpieczeń. Jednak testy programu IVT są uruchamiane z tym plikiem strategii i domyślnym menedżerem zabezpieczeń.

Weryfikowanie klas produktu IBM WebSphere MQ na potrzeby instalacji produktu Java

Program do weryfikacji instalacji, MQIVP, jest dostarczany z klasami produktu IBM WebSphere MQ dla produktu Java. Tego programu można użyć do przetestowania wszystkich trybów połączenia klas IBM WebSphere MQ dla produktu Java.

Program poprosi o podanie liczby wyborów i innych danych, aby określić, który tryb połączenia ma zostać sprawdzony. Aby sprawdzić instalację, wykonaj następującą procedurę:

1. Jeśli program ma być uruchamiany w trybie klienta, należy skonfigurować menedżer kolejek zgodnie z opisem w sekcji “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113. Kolejka, która ma być używana, to SYSTEM.DEFAULT.LOCAL.QUEUE.
2. Jeśli program ma być uruchamiany w trybie klienta, patrz także “Korzystanie z klas produktu WebSphere MQ dla języka Java” na stronie 669.

   Wykonaj pozostałe kroki tej procedury w systemie, w którym ma zostać uruchomiony program.
3. Upewnij się, że zaktualizowano zmienną środowiskową CLASSPATH zgodnie z instrukcjami w sekcji “Zmienne środowiskowe istotne dla klas WebSphere MQ dla języka Java” na stronie 673.
4. Przejdź do katalogu `MQ_INSTALLATION_PATH/mqm/VRM/java/samples/wmqjava`, gdzie MQ_INSTALLATION_PATH to ścieżka do instalacji produktu IBM WebSphere MQ , a `VRM` to numer wersji, wydania i modyfikacji produktu. Następnie w wierszu komend wpisz:

   ```
   java -Djava.library.path=library_path MQIVP
   ```

   gdzie *ścieżka_biblioteki* jest ścieżką do klas IBM WebSphere MQ dla bibliotek produktu Java (patrz Klasy WebSphere MQ dla bibliotek Java).

   W pytaniu zaznaczonym (1):

   - Aby użyć połączenia TCP/IP, wprowadź nazwę hosta serwera IBM WebSphere MQ .
   - Aby użyć połączenia rodzimego (tryb powiązań), należy pozostawić to pole puste (nie wpisuj nazwy).

   Program próbuje:

   - 1. Nawiąże połączenie z menedżerem kolejek
   - 2. Otwórz kolejkę SYSTEM.DEFAULT.LOCAL.QUEUE, umieść komunikat w kolejce, uzyskaj komunikat z kolejki, a następnie zamknij kolejkę.
   - 3. Odłączenie od menedżera kolejek
   - 4. Zwróć komunikat, jeśli operacje są pomyślne

Poniżej znajduje się przykład pytań i odpowiedzi, które można zobaczyć. Rzeczywiste zapytania i odpowiedzi zależą od sieci produktu IBM WebSphere MQ .

```
Please enter the IP address of the MQ server              : ipaddress(1)
Please enter the port to connect to                       : (1414)(2)
Please enter the server connection channel name            : channelname(2)
Please enter the queue manager name                       : qmname
Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager


Tests complete -
SUCCESS: This MQ Transport is functioning correctly.
Press Enter to continue ...
```

Uwaga:

1. Jeśli zostanie wybrane połączenie z serwerem, nie zostaną wyświetlone zachęty oznaczone jako (2).

Rozwiązywanie problemów z produktem IBM WebSphere MQ

Początkowo uruchom program weryfikujący instalację. Może być również konieczne użycie funkcji śledzenia.

Jeśli program nie zakończy się pomyślnie, uruchom program sprawdzający instalację i postępuj zgodnie z zaleceniami podanymi w komunikatach diagnostycznych. Ten program jest opisany w sekcji “Weryfikowanie klas produktu IBM WebSphere MQ na potrzeby instalacji produktu Java” na stronie 682.

Jeśli problemy są kontynuowane, a użytkownik musi skontaktować się z zespołem serwisowym IBM , może zostać poproszony o włączenie funkcji śledzenia. Należy to zrobić w sposób przedstawiony w poniższym przykładzie.

Aby śledzić program MQIVP :

- Utwórz plik właściwości *com.ibm.mq.commonservices* (patrz sekcja Korzystanie z usługi com.ibm.mq.commonservices .
- Wprowadź następującą komendę:

```
java -Dcom.ibm.mq.commonservices=commonservices_properties_file java
-Djava.library.path=library_path MQIVP -trace
```

gdzie:

- *plik właściwości commonservices_properties_file* to ścieżka (w tym nazwa pliku) do pliku właściwości *com.ibm.mq.commonservices* .
- *ścieżka_biblioteki* jest ścieżką do klas WebSphere MQ dla bibliotek Java (patrz Klasy WebSphere MQ dla bibliotek Java).

Więcej informacji na temat sposobu korzystania ze śledzenia zawiera sekcja Śledzenie aplikacji IBM WebSphere MQ classes for Java.

Wprowadzenie dla programistów

Ta kolekcja tematów zawiera ogólne informacje o programistach.

Więcej szczegółowych informacji na temat pisania programów zawiera sekcja “Pisanie klas produktu WebSphere MQ dla aplikacji Java” na stronie 685.

Klasy produktu WebSphere MQ dla interfejsu Java

W proceduralnym interfejsie programistycznym aplikacji WebSphere MQ używane są czasowniki, które działają na obiektach. Interfejs programistyczny Java korzysta z obiektów, które działają przy użyciu metod wywołującej.

Proceduralny interfejs programistyczny aplikacji WebSphere MQ jest zbudowany wokół komend, takich jak te:

```
       MQBACK, MQBEGIN, MQCLOSE, MQCONN, MQDISC,
       MQGET, MQINQ, MQOPEN, MQPUT, MQSET, MQSUB
```

Te czasowniki przyjmują, jako parametr, uchwyt obiektu WebSphere MQ , na którym mają działać. Ponieważ język Java jest zorientowany obiektowo, interfejs programistyczny Java zamienia tę rundkę. Program składa się z zestawu obiektów WebSphere MQ , które są używane przez wywołania metod na tych obiektach.

Podczas korzystania z interfejsu proceduralnego następuje rozłączenie z menedżerem kolejek przy użyciu wywołania MQDISC (Hconn, CompCode, Reason), gdzie *Hconn* jest uchwytem dla menedżera kolejek.

W interfejsie Java menedżer kolejek jest reprezentowany przez obiekt klasy MQQueueManager. Odłącz się od menedżera kolejek, wywołując metodę disconnect () dla tej klasy.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.disconnect();
```

Pisanie klas produktu WebSphere MQ dla aplikacji Java

Ta kolekcja tematów zawiera informacje pomocne przy pisaniu aplikacji Java w celu interakcji z systemami WebSphere MQ .

Aby użyć klas produktu WebSphere MQ dla języka Java w celu uzyskania dostępu do kolejek produktu WebSphere MQ , należy napisać aplikacje Java zawierające wywołania, które wstawiają komunikaty do kolejek i pobierają komunikaty z kolejek produktu WebSphere MQ i pobierają komunikaty z nich. Szczegółowe informacje na temat poszczególnych klas zawiera sekcja Klasy WebSphere MQ dla języka Java.

Uwaga: Automatyczne ponowne połączenie klienta nie jest obsługiwane przez klasy produktu WebSphere MQ dla języka Java.

Klasy WebSphere MQ dla trybów połączenia Java

Sposób programu dla klas WebSphere MQ dla języka Java ma pewne zależności od trybów połączenia, które mają być używane.

Jeśli używane są połączenia klienckie, istnieje wiele różnic między IBM WebSphere MQ MQI client , ale jest to koncepcyjnie podobne. Jeśli używany jest tryb powiązań, można użyć powiązań krótkiej ścieżki i można wywołać komendę MQBEGIN. Należy określić tryb, który ma być używany przez ustawienie zmiennych w klasie MQEnvironment.

Klasy produktu WebSphere MQ dla połączeń klienckich Java

Jeśli klasy produktu WebSphere MQ dla języka Java są używane jako klient, to jest on podobny do IBM WebSphere MQ MQI client, ale ma wiele różnic.

Jeśli programowanie dla produktu *Klasy WebSphere MQ dla języka Java* jest przeznaczone do użytku jako klient, należy pamiętać o następujących różnicach:

- Obsługuje on tylko protokół TCP/IP.
- Podczas uruchamiania nie odczytuje żadnych zmiennych środowiskowych produktu WebSphere MQ .
- Informacje, które będą przechowywane w definicji kanału oraz w zmiennych środowiskowych, mogą być przechowywane w klasie o nazwie Środowisko. Alternatywnie, informacje te mogą być przekazywane jako parametry podczas nawiązywania połączenia.
- Warunki błędu i wyjątku są zapisywane w dzienniku określonym w klasie `MQException` . Domyślnym miejscem docelowym błędów jest konsola Java.
- Tylko następujące atrybuty w pliku konfiguracyjnym klienta WebSphere MQ są odpowiednie dla klas produktu WebSphere MQ dla języka Java. Jeśli zostaną określone inne atrybuty, są one nieskuteczne.

sekcja	**Atrybut**
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	ExitsDefaultPath
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	ExitsDefaultPath64
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	Ścieżka klasy JavaExits
Sekcja MessageBuffer pliku konfiguracyjnego klienta	MaximumSize
Sekcja MessageBuffer pliku konfiguracyjnego klienta	PurgeTime
Sekcja MessageBuffer pliku konfiguracyjnego klienta	UpdatePercentage
Sekcja TCP pliku konfiguracyjnego klienta	ClntRcvBufSize

sekcja	Atrybut
Sekcja TCP pliku konfiguracyjnego klienta	ClntSndBufSize
Sekcja TCP pliku konfiguracyjnego klienta	Limit_czasu_połączenia
Sekcja TCP pliku konfiguracyjnego klienta	KeepAlive

- W przypadku łączenia się z menedżerem kolejek, który wymaga konwersji danych znakowych, klient Java V7 jest teraz w stanie wykonać konwersję, jeśli menedżer kolejek nie jest w stanie wykonać tej operacji. Maszyna JVM klienta musi obsługiwać konwersję między identyfikatorem CCSID klienta a tym menedżerem kolejek.
- Klasy WebSphere MQ classes for Java nie obsługują automatycznego nawiązywania ponownego połączenia przez klient.

W przypadku użycia w trybie klienta produkt *Klasy WebSphere MQ dla języka Java* nie obsługuje wywołania MQBEGIN.

Więcej informacji na temat obsługiwanych środowisk zawiera sekcja "Opcje połączenia dla klas WebSphere MQ dla języka Java" na stronie 670 .

Klasy WebSphere MQ dla trybu powiązań Java

Tryb powiązań klas WebSphere MQ dla języka Java różni się od trybu klienta na trzy główne sposoby.

W przypadku użycia w trybie powiązań klasy WebSphere MQ dla języka Java korzystają z interfejsu JNI (Java Native Interface) w celu wywołania bezpośrednio do istniejącego interfejsu API menedżera kolejek, zamiast komunikowania się za pośrednictwem sieci.

Domyślnie aplikacje korzystające z klas WebSphere MQ dla języka Java w trybie powiązań łączą się z menedżerem kolejek przy użyciu opcji *ConnectOption*, MQCNO_STANDARD_BINDINGS.

Klasy produktu WebSphere MQ dla języka Java obsługują następujące elementy *ConnectOptions*:

- MQCNO_FASTPATH_BINDING
- MQCNO_STANDARD_BINDING
- MQCNO_SHARED_BINDING
- MQCNO_ISOLATED_BINDING

Więcej informacji na temat opcji *ConnectOptions*można znaleźć w sekcji "Nawiąże połączenie z menedżerem kolejek przy użyciu wywołania MQCONNX" na stronie 214.

Tryb powiązań obsługuje wywołanie komendy MQBEGIN w celu zainicjowania globalnych jednostek pracy, które są koordynowane przez menedżer kolejek, na wszystkich platformach poza produktem WebSphere MQ for IBM i i WebSphere MQ for z/OS.

Większość parametrów udostępnianych przez klasę `MQEnvironment` nie jest odpowiednia dla trybu powiązań i są ignorowane.

Więcej informacji na temat obsługiwanych środowisk zawiera sekcja "Opcje połączenia dla klas WebSphere MQ dla języka Java" na stronie 670 .

Definiowanie klas WebSphere MQ dla połączeń Java, które mają być używane.

Typ połączenia, który ma być używany, jest określany przez ustawienie zmiennych w klasie MQEnvironment.

Używane są dwie zmienne:

MQEnvironment.properties
Typ połączenia jest określany na podstawie wartości powiązanej z nazwą klucza `CMQC.TRANSPORT_PROPERTY`. Lista poprawnych wartości:

CMQC.TRANSPORT_MQSERIES_BINDINGS
Połącz w trybie powiązań

CMQC.TRANSPORT_MQSERIES_CLIENT
Połącz w trybie klienta

CMQC.TRANSPORT_MQSERIES
Tryb połączenia jest określany na podstawie wartości właściwości *hostname* .

MQEnvironment.hostname
Ustaw wartość tej zmiennej w następujący sposób:

- W przypadku połączeń klienckich ustaw wartość tej zmiennej na nazwę hosta serwera IBM WebSphere MQ , z którym ma zostać nawiązane połączenie.
- W przypadku trybu powiązań nie należy ustawiać tej zmiennej ani ustawić jej na wartość NULL.

Operacje na menedżerach kolejek

Ta kolekcja tematów zawiera opis sposobu nawiązywania połączenia z menedżerem kolejek i ich rozłączania przy użyciu klas produktu WebSphere MQ dla języka Java.

Konfigurowanie środowiska produktu WebSphere MQ dla klas produktu WebSphere MQ dla języka Java

W przypadku aplikacji w celu nawiązania połączenia z menedżerem kolejek w trybie klienta aplikacja musi określić nazwę kanału, nazwę hosta i numer portu.

Uwaga: Informacje zawarte w tym temacie są istotne tylko wtedy, gdy aplikacja łączy się z menedżerem kolejek w trybie klienta. *Nie* jest to istotne, jeśli łączy się w trybie powiązań. Patrz: "Tryby połączenia dla klas produktu WebSphere MQ dla usługi JMS" na stronie 794

Nazwę kanału, nazwę hosta i numer portu można określić na jeden z dwóch sposobów: jako pola w klasie MQEnvironment lub jako właściwości obiektu MQQueueManager .

Jeśli pola zostaną ustawione w klasie MQEnvironment, będą one stosowane do całej aplikacji, z wyjątkiem sytuacji, w których są one nadpisywane przez tabelę mieszającą właściwości. Aby określić nazwę kanału i nazwę hosta w środowisku MQEnvironment, należy użyć następującego kodu:

```
MQEnvironment.hostname = "host.domain.com";
MQEnvironment.channel  = "java.client.channel";
```

Jest to równoznaczne z ustawieniem zmiennej środowiskowej **MQSERVER** :

```
"java.client.channel/TCP/host.domain.com".
```

Domyślnie klienty Java podejmują próbę nawiązania połączenia z programem nasłuchującym produktu WebSphere MQ na porcie 1414. Aby określić inny port, należy użyć następującego kodu:

```
MQEnvironment.port = nnnn;
```

gdzie nnnn jest wymaganym numerem portu

Jeśli właściwości zostaną przekazane do obiektu menedżera kolejek przy jego utworzeniu, będą one miały zastosowanie tylko do tego menedżera kolejek. Tworzenie wpisów w obiekcie Hashtable z kluczami **hostname**, **channel**i opcjonalnie **port**oraz z odpowiednimi wartościami. Aby użyć portu domyślnego (1414), można pominąć pozycję **port** . Utwórz obiekt MQQueueManager przy użyciu konstruktora, który akceptuje tabelę mieszającą właściwości.

Identyfikowanie połączenia z menedżerem kolejek przez ustawienie nazwy aplikacji

Aplikacja może ustawić nazwę, która identyfikuje jego połączenie z menedżerem kolejek. Ta nazwa aplikacji jest wyświetlana za pomocą komendy **DISPLAY CONN MQSC/PCF** (gdzie pole to nosi nazwę

APPLTAG) lub na ekranie WebSphere MQ Explorer **Połączenia aplikacji** (gdzie pole to jest nazywane **App name**).

Nazwy aplikacji są ograniczone do 28 znaków, a dłuższe nazwy są obcinane w celu dopasowania do nich. Jeśli nazwa aplikacji nie jest określona, zostanie podana wartość domyślna. Nazwa domyślna jest oparta na klasie wywołującej (main), ale jeśli te informacje nie są dostępne, zostanie użyty tekst `WebSphere MQ Client for Java`.

Jeśli używana jest nazwa klasy wywołującej, jest ona dopasowywana w taki sposób, aby pasować, usuwając początkowe nazwy pakietów, jeśli to konieczne. Na przykład, jeśli klasą wywołującym jest `com.example.MainApp`, używana jest pełna nazwa, ale jeśli klasą wywołującym jest `com.example.dictionaryAndThesaurus.multilingual.mainApp`, używana jest nazwa `multilingual.mainApp` , ponieważ jest to najdłuższa kombinacja nazwy klasy i najbardziej należnej nazwy pakietu, która mieści się w dostępnej długości.

Jeśli sama nazwa klasy ma więcej niż 28 znaków, zostanie obcięta do dopasowania. Na przykład `com.example.mainApplicationForSecondTestCase` staje się `mainApplicationForSecondTest`.

Uwaga: Menedżery kolejek działające na platformach z/OS nie obsługują ustawiania nazw aplikacji.

Aby ustawić nazwę aplikacji w klasie MQEnvironment, należy dodać nazwę do tabeli mieszającej MQEnvironment.properties z kluczem **`MQConstants.APPNAME_PROPERTY`**przy użyciu następującego kodu:

```
MQEnvironment.properties.put(MQConstants.APPNAME_PROPERTY, "my_application_name");
```

Aby ustawić nazwę aplikacji w tabeli mieszającej właściwości, która jest przekazywana do konstruktora MQQueueManager , należy dodać nazwę do tabeli mieszającej właściwości z kluczem **`MQConstants.APPNAME_PROPERTY`**.

Nadpisywanie właściwości określonych w pliku konfiguracyjnym klienta WebSphere MQ

Plik konfiguracyjny klienta WebSphere MQ może również określać właściwości, które są używane do konfigurowania klas WebSphere MQ dla języka Java. Jednak właściwości określone w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ mają zastosowanie tylko wtedy, gdy aplikacja łączy się z menedżerem kolejek w trybie klienta.

W razie potrzeby można przesłonić dowolny atrybut w pliku konfiguracyjnym produktu WebSphere MQ w jeden z następujących sposobów. Opcje są wyświetlane w kolejności wykonywania.

- Ustaw właściwość systemową Java dla właściwości konfiguracji.
- Ustaw właściwość w odwzorowaniu MQEnvironment.properties .
- W wersjach Java5 i nowszych ustaw systemową zmienną środowiskową.

Tylko następujące atrybuty w pliku konfiguracyjnym klienta WebSphere MQ są istotne dla klas produktu WebSphere MQ dla języka Java. Jeśli zostaną określone lub nadpisane inne atrybuty, nie będzie to miało żadnego wpływu.

sekcja	**Atrybut**
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	ExitsDefaultPath
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	ExitsDefaultPath64
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	Ścieżka klasy JavaExits
Sekcja MessageBuffer pliku konfiguracyjnego klienta	MaximumSize

sekcja	Atrybut
Sekcja MessageBuffer pliku konfiguracyjnego klienta	PurgeTime
Sekcja MessageBuffer pliku konfiguracyjnego klienta	UpdatePercentage
Sekcja TCP pliku konfiguracyjnego klienta	ClntRcvBufSize
Sekcja TCP pliku konfiguracyjnego klienta	ClntSndBufSize
Sekcja TCP pliku konfiguracyjnego klienta	Limit_czasu_połączenia
Sekcja TCP pliku konfiguracyjnego klienta	KeepAlive

Nawiąże połączenie z menedżerem kolejek w klasach produktu WebSphere MQ dla języka Java

Połącz się z menedżerem kolejek, tworząc nową instancję klasy MQQueueManager . Odłącz się od menedżera kolejek, wywołując metodę rozłączania ().

Teraz można nawiązać połączenie z menedżerem kolejek, tworząc nową instancję klasy MQQueueManager :

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Aby rozłączyć się z menedżerem kolejek, wywołaj metodę disconnect () w menedżerze kolejek:

```
queueManager.disconnect();
```

Jeśli zostanie wywołana metoda rozłączania, wszystkie otwarte kolejki i procesy, do których użytkownik uzyskuje dostęp za pośrednictwem tego menedżera kolejek, są zamykane. Jednak dobrym rozwiązaniem programowym jest zamknięcie tych zasobów w sposób jawny po zakończeniu korzystania z nich. Aby to zrobić, należy użyć metody close () dla odpowiednich obiektów.

Metody commit () i backout () w menedżerze kolejek są równoważne wywołaniach MQCMIT i MQBACK, które są używane w interfejsie proceduralnym.

Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM WebSphere MQ classes for Java

Aplikacja kliencka IBM WebSphere MQ classes for Java może korzystać z definicji kanału połączenia klienta zapisanych w tabeli definicji kanału klienta (CCDT).

Alternatywą dla tworzenia definicji kanału połączenia klienta przez ustawienie określonych pól i właściwości środowiska w klasie `MQEnvironment` lub przekazywania ich do partycji `MQQueueManager` w tabeli mieszającej właściwości, aplikacja kliencka IBM WebSphere MQ classes for Java może używać definicji kanału połączenia klienta zapisanych w tabeli definicji kanału klienta. Definicje te są tworzone za pomocą komendIBM WebSphere MQ Script (MQSC) lub IBM WebSphere MQ Programmable Command Format (PCF) lub za pomocą IBM WebSphere MQ Explorer.

Gdy aplikacja tworzy obiekt `MQQueueManager` , klient IBM WebSphere MQ classes for Java przeszukuje tabelę definicji kanału klienta dla odpowiedniej definicji kanału połączenia klienckiego i używa definicji kanału w celu uruchomienia kanału MQI. Więcej informacji na temat tabel definicji kanału klienta i sposobu ich tworzenia zawiera sekcja Tabela definicji kanału klienta.

Aby użyć tabeli definicji kanału klienta, aplikacja musi najpierw utworzyć obiekt URL. Obiekt URL hermetyzuje jednolity wskaźnik zasobów (URL), który identyfikuje nazwę i położenie pliku zawierającego tabelę definicji kanału klienta i określa sposób dostępu do pliku.

Na przykład, jeśli plik `ccdt1.tab` zawiera tabelę definicji kanału klienta i jest przechowywany w tym samym systemie, w którym działa aplikacja, aplikacja może utworzyć obiekt URL w następujący sposób:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
```

Załóżmy na przykład, że plik `ccdt2.tab` zawiera tabelę definicji kanału klienta i jest przechowywany w systemie innym niż ten, w którym działa aplikacja. Jeśli dostęp do pliku można uzyskać za pomocą protokołu FTP, aplikacja może utworzyć obiekt URL w następujący sposób:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
```

Po utworzeniu obiektu adresu URL aplikacja może utworzyć obiekt `MQQueueManager` , korzystając z jednego z konstruktorów, który pobiera obiekt URL jako parametr. Oto przykład:

```
MQQueueManager mars = new MQQueueManager("MARS", chanTab2);
```

Ta instrukcja powoduje, że klient IBM WebSphere MQ classes for Java uzyskuje dostęp do tabeli definicji kanału klienta identyfikowanej przez obiekt URL chanTab2, wyszuka odpowiednią definicję kanału połączenia klienta, a następnie używa definicji kanału w celu uruchomienia kanału MQI dla menedżera kolejek o nazwie MARS.

Należy zwrócić uwagę na następujące punkty, które mają zastosowanie, jeśli aplikacja korzysta z tabeli definicji kanału klienta:

- Gdy aplikacja tworzy obiekt `MQQueueManager` przy użyciu konstruktora, który pobiera obiekt URL jako parametr, żadna nazwa kanału nie musi być ustawiona w klasie `MQEnvironment` jako właściwość w polu lub jako właściwość środowiska. Jeśli ustawiona jest nazwa kanału, klient IBM WebSphere MQ classes for Java zgłasza `MQException`. Właściwość pola lub środowiska określająca nazwę kanału jest uznawana za ustawioną, jeśli jej wartością jest dowolna wartość inna niż null, pusty łańcuch lub łańcuch zawierający wszystkie puste znaki.
- Parametr **queueManagerName** w konstruktorze `MQQueueManager` może mieć jedną z następujących wartości:
  - Nazwa menedżera kolejek
  - Gwiazdka (*), po której następuje nazwa grupy menedżerów kolejek
  - Gwiazdka (*)
  - Wartość NULL, pusty łańcuch lub łańcuch zawierający wszystkie puste znaki.

  Są to te same wartości, których można użyć dla parametru **QMgrName** w wywołaniu MQCONN wywoływanym przez aplikację kliencką korzystaniową z interfejsu kolejki komunikatów (Message Queue Interface-MQI). Więcej informacji na temat znaczenia tych wartości znajduje się w sekcji “Interfejs kolejki komunikatów-przegląd” na stronie 199.

  Jeśli aplikacja korzysta z zestawiania połączeń, należy zapoznać się z “Sterowanie domyślną pulą połączeń w klasach produktu WebSphere MQ dla języka Java” na stronie 711.
- Gdy klient IBM WebSphere MQ classes for Java znajdzie odpowiednią definicję kanału połączenia klienta w tabeli definicji kanału klienta, używa ona tylko informacji wyodrębnionych z tej definicji kanału w celu uruchomienia kanału MQI. Wszystkie pola powiązane z kanałem lub właściwości środowiska, które aplikacja mogła ustawić w klasie `MQEnvironment` , są ignorowane.

  W przypadku korzystania z protokołu SSL (Secure Sockets Layer) należy w szczególności zwrócić uwagę na następujące kwestie:
  - Kanał MQI używa protokołu SSL tylko wtedy, gdy definicja kanału wyodrębniona z tabeli definicji kanału klienta określa nazwę klasy CipherSpec obsługiwaną przez klient IBM WebSphere MQ classes for Java .
  - Tabela definicji kanału klienta zawiera również informacje na temat położenia serwerów LDAP (Lightweight Directory Access Protocol), które przechowują listy odwołań certyfikatów (CRL). Klient

IBM WebSphere MQ classes for Java korzysta tylko z tych informacji w celu uzyskania dostępu do serwerów LDAP, które przechowują listy CRL.

– Tabela definicji kanału klienta może również zawierać położenie odpowiadającego protokołu OCSP (Online Certificate Status Protocol). IBM WebSphere MQ classes for Java nie mogą korzystać z informacji OCSP w pliku tabeli definicji kanału klienta. Można jednak skonfigurować protokół OCSP w sposób opisany w sekcji Korzystanie z protokołu Online Certificate Protocol.

Więcej informacji na temat korzystania z protokołu SSL przy użyciu tabeli definicji kanału klienta zawiera sekcja Określanie, czy kanał MQI używa protokołu SSL.

Jeśli używane są wyjścia kanału, należy pamiętać również o następujących punktach:

– Kanał MQI używa wyjść kanału i powiązanych danych użytkownika określonych przez definicję kanału wyodrębnioną z tabeli definicji kanału klienta w preferencjach do wyjść kanału i danych określonych przy użyciu innych metod.

– Definicja kanału wyodrębniona z tabeli definicji kanału klienta może określać wyjścia kanału zapisane w produkcie Java, C lub C + +. Więcej informacji na temat sposobu pisania wyjścia kanału w programie Javazawiera sekcja “Tworzenie wyjścia kanału w klasach produktu WebSphere MQ dla języka Java” na stronie 704. Więcej informacji na temat pisania wyjścia kanału w innych językach zawiera sekcja “Używanie wyjść kanału nie napisanych w języku Java z klasami produktu WebSphere MQ dla języka Java” na stronie 708.

Określanie zakresu portów dla połączeń klienckich systemu IBM WebSphere MQ classes for Java

Można określić port lub zakres portów, które mogą być powiązane z aplikacją na jeden z dwóch sposobów.

Gdy aplikacja IBM WebSphere MQ classes for Java próbuje nawiązać połączenie z menedżerem kolejek produktu IBM WebSphere MQ w trybie klienta, firewall może zezwalać tylko na połączenia, które pochodzą z określonych portów lub zakresu portów. W takiej sytuacji można określić port lub zakres portów, z którymi aplikacja może się wiązać. Port (y) można określić w następujący sposób:

- Pole localAddress(Adres lokalny) można ustawić w klasie MQEnvironment. Oto przykład:

```
MQEnvironment.localAddressSetting = "192.0.2.0(2000,3000)";
```

- Istnieje możliwość ustawienia właściwości środowiska CMQC.LOCAL_ADDRESS_PROPERTY. Oto przykład:

```
(MQEnvironment.properties).put(CMQC.LOCAL_ADDRESS_PROPERTY,
                               "192.0.2.0(2000,3000)");
```

- Jeśli można skonstruować obiekt MQQueueManager , można przekazać właściwości hashtable zawierające właściwość LOCAL_ADDRESS_PROPERTY o wartości "192.0.2.0(2000,3000)".

W każdym z tych przykładów, gdy aplikacja nawiąże później połączenie z menedżerem kolejek, aplikacja wiąże się z lokalnym adresem IP i numerem portu z zakresu od 192.0.2.0(2000) do 192.0.2.0(3000).

W systemie z więcej niż jednym interfejsem sieciowym można również użyć pola localAddressSetting (Adres lokalny) lub właściwości środowiska CMQC.LOCAL_ADDRESS_PROPERTY, aby określić, który interfejs sieciowy musi być używany dla połączenia.

W przypadku ograniczenia zakresu portów mogą wystąpić błędy połączenia. Jeśli wystąpi błąd, zgłaszany jest wyjątek MQException zawierający kod przyczyny IBM WebSphere MQ MQRC_Q_MGR_NOT_AVAILABLE i następujący komunikat:

```
Socket connection attempt refused due to LOCAL_ADDRESS_PROPERTY restrictions
```

Błąd może wystąpić, jeśli używane są wszystkie porty w podanym zakresie lub jeśli podany adres IP, nazwa hosta lub numer portu nie są poprawne (na przykład jest to ujemny numer portu).

Uzyskiwanie dostępu do kolejek, tematów i procesów w klasach produktu WebSphere MQ dla języka Java

Aby uzyskać dostęp do kolejek, tematów i procesów, należy użyć metod klasy MQQueueManager . Tabela MQOD (struktura deskryptora obiektu) jest zwinięta do parametrów tych metod.

Kolejki

Aby otworzyć kolejkę, można użyć metody accessQueue klasy MQQueueManager . Na przykład w menedżerze kolejek o nazwie queueManagernależy użyć następującego kodu:

```
MQQueue queue = queueManager.accessQueue("qName",CMQC.MQOO_OUTPUT);
```

Metoda accessQueue zwraca nowy obiekt klasy MQQueue.

Po zakończeniu korzystania z kolejki należy użyć metody close (), aby ją zamknąć, tak jak w następującym przykładzie:

```
queue.close();
```

Kolejkę można również utworzyć za pomocą konstruktora MQQueue. Parametry są dokładnie takie same, jak w przypadku metody accessQueue , z dodaniem parametru menedżera kolejek. Na przykład:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            CMQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserID");
```

W przypadku tworzenia kolejek można określić wiele opcji. Szczegółowe informacje na ten temat można znaleźć w sekcji Class.com.ibm.mq.MQQueue. Konstruowanie obiektu kolejki w ten sposób umożliwia zapisywanie własnych podklas kolejki MQQueue.

Tematy

W podobny sposób można otworzyć temat przy użyciu metody accessTopic klasy MQQueueManager . Na przykład w menedżerze kolejek o nazwie queueManagernależy użyć następującego kodu, aby utworzyć subskrybent i publikator:

```
MQTopic subscriber =
      queueManager.accessTopic("TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

```
MQTopic publisher =
      queueManager.accessTopic("TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_PUBLICATION, CMQC.MQOO_OUTPUT);
```

Po zakończeniu korzystania z tematu należy użyć metody close (), aby ją zamknąć.

Istnieje również możliwość utworzenia tematu przy użyciu konstruktora MQTopic. Parametry są dokładnie takie same, jak w przypadku metody accessTopic , z dodaniem parametru menedżera kolejek. Na przykład:

```
MQTopic subscriber = new
      MQTopic(queueManager,"TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

Podczas tworzenia tematów można określić wiele opcji. Szczegółowe informacje na ten temat zawiera sekcja Klasa com.ibm.mq.MQTopic . Konstruowanie obiektu tematu w ten sposób umożliwia pisanie własnych podklas tematu MQTopic.

Temat musi zostać otwarty albo w celu opublikowania, albo w celu subskrypcji. Klasa MQQueueManager ma osiem metod accessTopic , a klasa Topic ma osiem konstruktorów. W każdym przypadku cztery z nich

mają parametr **destination** , a cztery mają parametr **subscriptionName** (w tym dwa, które mają oba te parametry). Mogą one być używane tylko do otwierania tematu dla subskrypcji. Dwie pozostałe metody mają parametr **openAs** , a temat może zostać otwarty dla publikacji lub subskrypcji w zależności od wartości parametru **openAs** .

Aby utworzyć temat jako trwały subskrybent, należy użyć metody accessTopic klasy MQQueueManager lub konstruktora MQTopic, który akceptuje nazwę subskrypcji, a w obu przypadkach należy ustawić wartość CMQC.MQSO_DURABLE .

Procesy

Aby uzyskać dostęp do procesu, należy użyć metody accessProcess obiektu MQQueueManager. Na przykład w menedżerze kolejek o nazwie queueManagernależy użyć następującego kodu w celu utworzenia obiektu MQProcess:

```
MQProcess process =
      queueManager.accessProcess("PROCESSNAME",
      CMQC.MQOO_FAIL_IF_QUIESCING);
```

Aby uzyskać dostęp do procesu, należy użyć metody accessProcess obiektu MQQueueManager.

Metoda accessProcess zwraca nowy obiekt MQProcess klasy.

Po zakończeniu korzystania z obiektu procesu należy użyć metody close (), aby ją zamknąć, tak jak w następującym przykładzie:

```
process.close();
```

Proces można również utworzyć za pomocą konstruktora MQProcess. Parametry są dokładnie takie same, jak w przypadku metody accessProcess , z dodaniem parametru menedżera kolejek. Na przykład:

```
 MQProcess process =
      new MQProcess(queueManager,"PROCESSNAME",
      CMQC.MQOO_FAIL_IF_QUIESCING);
```

Konstruowanie obiektu procesu w ten sposób umożliwia pisanie własnych podklas procesu MQProcess.

Obsługa komunikatów w klasach produktu WebSphere MQ dla języka Java

Komunikaty są reprezentowane przez klasę MQMessage. Komunikaty są umieszczane i wysyłane przy użyciu metod klasy MQDestination, które mają podklasy MQQueue i MQTopic.

Umieszczanie komunikatów w kolejkach lub tematach przy użyciu metody put () klasy MQDestination. Komunikaty z kolejek lub tematów są wysyłane za pomocą metody get () klasy MQDestination. W przeciwieństwie do interfejsu proceduralnego, w którym są wstawiane tabele MQPUT i MQGET i otrzymujemy tablice bajtów, język programowania Java umieszcza i pobiera instancje klasy MQMessage. Klasa MQMessage hermetykuje bufor danych, który zawiera rzeczywiste dane komunikatu, wraz ze wszystkimi parametrami deskryptora MQMD (deskryptor komunikatu) oraz właściwościami komunikatu opisujących ten komunikat.

Aby zbudować nowy komunikat, należy utworzyć nową instancję klasy MQMessage i użyć metod writeXXX w celu umieszczenia danych w buforze komunikatów.

Po utworzeniu nowej instancji komunikatu wszystkie parametry MQMD są automatycznie ustawiane na wartości domyślne, zgodnie z definicją w sekcji Wartości początkowe i deklaracje języków dla deskryptora MQMD . Metoda put () obiektu MQDestination pobiera również instancję klasy opcji MQPutMessagejako parametr. Ta klasa reprezentuje strukturę MQPMO. W poniższym przykładzie tworzony jest komunikat i umieszcza go w kolejce:

```
// Build a new message containing my age followed by my name
MQMessage myMessage = new MQMessage();
myMessage.writeInt(25);

String name = "Charlie Jordan";
```

```
myMessage.writeInt(name.length());
myMessage.writeBytes(name);

// Use the default put message options...
MQPutMessageOptions pmo = new MQPutMessageOptions();

// put the message!
queue.put(myMessage,pmo);
```

Metoda get () obiektu MQDestination zwraca nową instancję komunikatu MQMessage, która reprezentuje komunikat, który został właśnie odebrany z kolejki. Pobiera ona również instancję klasy opcji MQGetMessagejako parametr. Ta klasa reprezentuje strukturę MQGMO.

Nie ma potrzeby określania maksymalnej wielkości komunikatu, ponieważ metoda get () automatycznie dostosowuje wielkość swojego wewnętrznego buforu, aby pasował do komunikatu przychodzącego. Aby uzyskać dostęp do danych w zwróconej wiadomości, należy użyć metod readXXX klasy MQMessage.

W poniższym przykładzie przedstawiono sposób pobrania komunikatu z kolejki:

```
// Get a message from the queue
MQMessage theMessage    = new MQMessage();
MQGetMessageOptions gmo = new MQGetMessageOptions();
queue.get(theMessage,gmo);  // has default values

// Extract the message data
int age = theMessage.readInt();
int strLen = theMessage.readInt();
byte[] strData = new byte[strLen];
theMessage.readFully(strData,0,strLen);
String name = new String(strData,0);
```

Istnieje możliwość zmiany formatu liczb, którego używają metody odczytu i zapisu, ustawiając zmienną elementu *encoding* .

Zestaw znaków używany do odczytu i zapisu łańcuchów można zmienić, ustawiając zmienną składową *characterSet* .

Więcej informacji na ten temat zawiera sekcja “Klasa MQMessage” na stronie 1099 .

Uwaga: Metoda writeUTF() obiektu MQMessage automatycznie koduje długość łańcucha, a także bajty Unicode, które zawiera. Gdy komunikat zostanie odczytany przez inny program Java (za pomocą komendy readUTF()), jest to najprostszy sposób wysyłania informacji o łańcuchach.

Zwiększanie wydajności komunikatów nietrwałych w klasach produktu WebSphere MQ dla języka Java

Aby zwiększyć wydajność podczas przeglądania komunikatów lub korzystania z nietrwałych komunikatów z aplikacji klienckiej, można użyć funkcji *read ahead*(odczyt z wyprzedzeniem). Aplikacje klienckie używające wywołania MQGET lub asynchronicznego będą korzystać z usprawnień wydajności podczas przeglądania komunikatów lub korzystania z nietrwałych komunikatów.

Ogólne informacje na temat narzędzia do odczytu z wyprzedzeniem zawiera sekcja pokrewny temat.

W klasach produktu WebSphere MQ dla języka Java jest używana klasa CMQC.MQSO_READ_AHEAD i CMQC.MQSO_NO_READ_AHEAD obiektu MQQueue lub obiektu MQTopic w celu określenia, czy konsumenci komunikatów i przeglądarki kolejek mogą używać odczytu z wyprzedzeniem dla tego obiektu.

Asynchroniczne umieszczanie komunikatów przy użyciu klas produktu WebSphere MQ dla języka Java

Aby umieścić komunikat asynchronicznie, należy ustawić wartość MQPMO_ASYNC_RESPONSE.

Komunikaty umieszczane są w kolejkach lub tematach przy użyciu metody put () klasy MQDestination. Aby umieścić komunikat asynchronicznie, to znaczy zezwalając na zakończenie operacji bez oczekiwania na odpowiedź z menedżera kolejek, można ustawić wartość MQPMO_ASYNC_RESPONSE w polu opcji opcji MQPutMessage. Aby określić powodzenie lub niepowodzenie operacji put asynchronicznych, należy użyć wywołania statusu MQQueueManager.getAsync.

Publikowanie/subskrypcja w klasach produktu WebSphere MQ dla języka Java

W klasach produktu WebSphere MQ dla języka Java temat jest reprezentowany przez klasę MQTopic, a użytkownik opublikuje go przy użyciu metod MQTopic.put().

Ogólne informacje na temat publikowania/subskrypcji produktu WebSphere MQ zawiera sekcja Wprowadzenie do przesyłania komunikatów w trybie publikowania/subskrypcji produktu WebSphere MQ .

Obsługa nagłówków komunikatów produktu WebSphere MQ z klasami produktu WebSphere MQ dla języka Java

Dostępne są klasy Java reprezentujące różne typy nagłówków komunikatów. Dostępne są również dwie klasy pomocnicze.

Obiekty nagłówka są opisane przez interfejs MQHeader, który udostępnia metody ogólnego przeznaczenia służące do uzyskiwania dostępu do pól nagłówka oraz do odczytu i zapisu treści komunikatu. Każdy typ nagłówka ma własną klasę implementującą interfejs MQHeader, a także dodaje metody pobierające i ustawiające dla poszczególnych pól. Na przykład typ nagłówka MQRFH2 jest reprezentowany przez klasę MQRFH2 ; typ nagłówka MQDLH przez klasę MQDLH, itd. Klasy nagłówka automatycznie wykonują wszelkie niezbędne konwersje danych, a także mogą odczytywać lub zapisywać dane w dowolnym określonym kodowaniu liczbowym lub zestawie znaków (CCSID).

Dwie klasy pomocnicze: MQHeaderIterator i MQHeaderList, wspomagają odczytywanie i dekodowanie (analizowanie) treści nagłówka w komunikatach:

- Klasa MQHeaderIterator działa w taki sposób, jak klasa java.util.Iterator. Tak długo, jak w komunikacie znajduje się więcej nagłówków, metoda next () zwraca wartość true, a metoda nextHeader() lub next () zwraca następny obiekt nagłówka.
- Obiekt MQHeaderList działa podobnie jak element java.util.List. Podobnie jak w przypadku elementu MQHeaderIterator, analizuje on treść nagłówka, ale umożliwia również wyszukiwanie konkretnych nagłówków, dodawanie nowych nagłówków, usuwanie istniejących nagłówków, aktualizowanie pól nagłówka, a następnie zapisywanie treści nagłówka z powrotem w komunikacie. Alternatywnie można utworzyć pustą listę MQHeaderList, a następnie zapełnić ją instancjami nagłówków i zapisać je do komunikatu raz lub wielokrotnie.

Klasy MQHeaderIterator i MQHeaderList używają informacji znajdujących się w rejestrze MQHeaderRegistry , aby wiedzieć, które klasy nagłówka WebSphere MQ są powiązane z określonymi typami komunikatów i formatami. Rejestr MQHeaderRegistry jest skonfigurowany ze znajomością wszystkich bieżących formatów i typów nagłówków produktu WebSphere MQ oraz ich klas implementacji, a także można zarejestrować własne typy nagłówków.

Obsługa jest udostępniana dla następujących powszechnie używanych nagłówków produktu Websphere MQ

- MQRFH-nagłówek reguł i formatowania
- MQRFH2 -podobnie jak MQRFH, używane do przekazywania komunikatów do i z brokera komunikatów należącego do produktu WebSphere Message Broker. Służy również do przechowywania właściwości komunikatu.
- MQCIH-most CICS
- MQDLH-Nagłówek niewysłanych wiadomości
- MQIIH-nagłówek informacji IMS
- MQRMH-nagłówek komunikatu odwołania
- MQSAPH-nagłówek SAP
- MQWIH-nagłówek informacji o pracy
- MQXQH-nagłówek kolejki transmisji

- MQDH-nagłówek dystrybucji
- MQEPH-nagłówek PCF hermetyzowanego

Można również zdefiniować klasy reprezentujące własne nagłówki.

Aby użyć elementu MQHeaderIterator w celu uzyskania nagłówka RFH2 , należy ustawić właściwość MQGMO_PROPERTIES_FORCE_MQRFH2 w opcjach GetMessagelub ustawić właściwość kolejki PROPCTL na FORCE.

Drukowanie wszystkich nagłówków w komunikacie przy użyciu klas produktu WebSphere MQ dla języka Java

W tym przykładzie instancja obiektu MQHeaderIterator analizuje nagłówki w komunikacie MQMessage, który został odebrany z kolejki. Obiekty MQHeader zwracane przez metodę nextHeader() wyświetlają ich strukturę i zawartość, gdy wywoływana jest metoda toString .

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeader;
import com.ibm.mq.headers.MQHeaderIterator;
...
MQMessage message = ... // Message received from a queue.
MQHeaderIterator it = new MQHeaderIterator (message);

while (it.hasNext ())
{
    MQHeader header = it.nextHeader ();

    System.out.println ("Header type " + header.type () + ": " + header);
}
```

Pomijanie nagłówków w komunikacie przy użyciu klas produktu WebSphere MQ dla języka Java

W tym przykładzie metoda skipHeaders() klasy MQHeaderIterator umieszcza kursor odczytu komunikatu bezpośrednio po ostatnim nagłówku.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderIterator;
...
MQMessage message = ... // Message received from a queue.
MQHeaderIterator it = new MQHeaderIterator (message);

it.skipHeaders ();
```

Znajdowanie kodu przyczyny w komunikacie niedostarczonych komunikatów przy użyciu klas produktu WebSphere MQ dla języka Java

W tym przykładzie metoda read zapełnia obiekt MQDLH odczytem z komunikatu. Po operacji odczytu kursor odczytu komunikatu jest ustawiany natychmiast po treści nagłówka MQDLH.

Komunikaty w kolejce niedostarczonych komunikatów menedżera kolejek są poprzedzane nagłówkiem z niedostarczonym literem (MQDLH). Aby zdecydować, w jaki sposób obsługiwać te komunikaty-na przykład w celu określenia, czy mają być one ponawiane, czy je odrzucić-w aplikacji obsługi niedostarczonych komunikatów należy sprawdzić kod przyczyny znajdujący się w tabeli MQDLH.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQDLH dlh = new MQDLH ();

dlh.read (message);

System.out.println ("Reason: " + dlh.getReason ());
```

Wszystkie klasy nagłówka udostępniają również wygodny konstruktor, który inicjuje się bezpośrednio z komunikatu w jednym kroku. Tak więc kod w tym przykładzie można uprościć w następujący sposób:

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQDLH dlh = new MQDLH (message);

System.out.println ("Reason: " + dlh.getReason ());
```

Odczytywanie i usuwanie tabeli MQDLH z komunikatu niedostarczonych komunikatów przy użyciu klas produktu WebSphere MQ dla języka Java

W tym przykładzie komenda MQDLH jest używana do usuwania nagłówka z komunikatu niewysłanych wiadomości.

Aplikacja obsługi niedostarczonych komunikatów będzie zazwyczaj ponownie wysyłać komunikaty, które zostały odrzucone, jeśli ich kod przyczyny wskazuje na błąd przejściowy. Przed ponownym przesłaniem komunikatu musi on usunąć nagłówek MQDLH.

W tym przykładzie wykonywane są następujące kroki (patrz komentarze w kodzie przykładowym):

1. Element MQHeaderList odczytuje całą wiadomość, a każdy nagłówek napotkany w komunikacie staje się elementem na liście.
2. Komunikaty o niewysłanych literach zawierają kod MQDLH jako pierwszy nagłówek, więc można go znaleźć w pierwszym elemencie listy nagłówków. Obiekt MQDLH został już zapełniony z komunikatu, gdy jest zbudowana lista MQHeaderList , dlatego nie ma potrzeby wywoływania metody odczytu.
3. Kod przyczyny jest wyodrębniany przy użyciu metody getReason() udostępnianej przez klasę MQDLH.
4. Kod przyczyny został sprawdzony i wskazuje, że należy wprowadzić ponownie komunikat. Zmaterializowana tabela MQDLH jest usuwana przy użyciu metody remove () MQHeaderList .
5. Obiekt MQHeaderList zapisuje swoją pozostałą treść w nowym obiekcie komunikatu. Nowy komunikat zawiera teraz wszystko, co znajduje się w oryginalnym komunikacie, z wyjątkiem MQDLH i może zostać zapisany w kolejce. Argument **true** dla konstruktora i metody zapisu wskazuje, że treść komunikatu ma być wstrzymana w obrębie elementu MQHeaderListi jest zapisywana ponownie.
6. Pole formatu w deskryptorze komunikatu nowego komunikatu zawiera teraz wartość, która była wcześniej określona w polu formatu MQDLH. Dane komunikatu są zgodne z kodowaniem liczbowym i zestawem CCSID ustawionym w deskryptorze komunikatu.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQHeaderList list = new MQHeaderList (message, true); // Step 1.
MQDLH dlh = (MQDLH) list.get (0); // Step 2.
int reason = dlh.getReason (); // Step 3.
...
list.remove (dlh); // Step 4.

MQMessage newMessage = new MQMessage ();

list.write (newMessage, true); // Step 5.
newMessage.format = list.getFormat (); // Step 6.
```

Drukowanie treści komunikatu przy użyciu klas produktu WebSphere MQ dla języka Java

W tym przykładzie użyto komendy MQHeaderList w celu wydrukowania treści komunikatu, w tym jego nagłówków.

Dane wyjściowe zawierają widok wszystkich treści nagłówka, a także treści komunikatu. Klasa MQHeaderList dekoduje wszystkie nagłówki w jednym kroku, podczas gdy iterator MQHeaderIterator wykonuje je po jednym w czasie pod kontrolą aplikacji. Tej techniki można użyć do udostępnienia prostego narzędzia do debugowania podczas zapisywania aplikacji produktu Websphere MQ .

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderList;
```

```
...
MQMessage message = ... // Message received from a queue.

System.out.println (new MQHeaderList (message, true));
```

Ten przykład powoduje również wydrukowanie pól deskryptora komunikatu przy użyciu klasy MQMD. Metoda copyFrom() klasy com.ibm.mq.headers.MQMD zapełnia obiekt nagłówka z pól deskryptora komunikatu w komunikacie MQMessage, a nie przez odczytywanie treści komunikatu.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQMD;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ...
MQMD md = new MQMD ();
...
md.copyFrom (message);
System.out.println (md + "\n" + new MQHeaderList (message, true));
```

Znajdowanie konkretnego typu nagłówka w komunikacie przy użyciu klas produktu WebSphere MQ dla języka Java

W tym przykładzie użyto metody indexOf(String) elementu MQHeaderList w celu znalezienia nagłówka MQRFH2 w komunikacie, jeśli taki nagłówek jest obecny.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderList;
import com.ibm.mq.headers.MQRFH2;
...
MQMessage message = ...
MQHeaderList list = new MQHeaderList (message);
int index = list.indexOf ("MQRFH2");

if (index >= 0)
{
    MQRFH2 rfh = (MQRFH2) list.get (index);
    ...
}
```

Analizowanie nagłówka MQRFH2 przy użyciu klas produktu WebSphere MQ dla języka Java

W tym przykładzie pokazano, jak można uzyskać dostęp do znanej wartości pola w nazwanym folderze, używając klasy MQRFH2 .

Klasa MQRFH2 udostępnia wiele sposobów uzyskiwania dostępu nie tylko do pól w stałej części struktury, ale także do treści folderu zakodowanego w formacie XML, które są przenoszone w polu NameValueData. W tym przykładzie pokazano, w jaki sposób można uzyskać dostęp do znanej wartości pola w nazwanym folderze-w tym przypadku pole Rto w folderze jms, które reprezentuje nazwę kolejki odpowiedzi w komunikacie JMS produktu MQ .

```
MQRFH2 rfh = ...

String value = rfh.getStringFieldValue ("jms", "Rto");
```

Aby wykryć treść obiektu MQRFH2 (w przeciwieństwie do bezpośredniego żądania konkretnych pól), można użyć metody getFolders w celu zwrócenia listy MQRFH2.Element, który reprezentuje strukturę folderu, który może zawierać pola i inne foldery. Ustawienie pola lub folderu na wartość NULL powoduje usunięcie go z obiektu MQRFH2. W przypadku manipulowania zawartością folderu danych NameValuew ten sposób pole StrucLength jest automatycznie aktualizowane.

Odczytywanie i zapisywanie strumieni bajtów innych niż obiekty MQMessage przy użyciu klas produktu WebSphere MQ dla języka Java

W tych przykładach używane są klasy nagłówka do analizowania treści nagłówka produktu WebSphere MQ i manipulowania nimi, gdy źródło danych nie jest obiektem MQMessage.

Klasy nagłówka można używać do analizowania treści nagłówka produktu WebSphere MQ i manipulowania nimi nawet wtedy, gdy źródło danych jest inne niż obiekt MQMessage. Interfejs MQHeader implementowany przez każdą klasę nagłówka udostępnia metody `int read (java.io.DataInput message, int encoding, int characterSet)` i `int write (java.io.DataOutput message, int encoding, int characterSet)`. Klasa com.ibm.mq.MQMessage implementuje interfejsy java.io.DataInput i java.io.DataOutput . Oznacza to, że można użyć dwóch metod MQHeader do odczytu i zapisu treści MQMessage, nadpisując kodowanie i identyfikator CCSID określone w deskryptorze komunikatu. Jest to przydatne w przypadku komunikatów, które zawierają łańcuch nagłówków w różnych kodowaniach.

Istnieje również możliwość uzyskania obiektów DataInput i DataOutput z innych strumieni danych, na przykład strumieni plików lub gniazd, lub tablic bajtowych przenoszonych w komunikatach JMS. Klasy java.io.DataInputStream implementują interfejs DataInput , a klasy java.io.DataOutputStream implementują DataOutput. W tym przykładzie treść nagłówka WebSphere MQ jest odczytywaana z tablicy bajtów:

```
import java.io.*;
import com.ibm.mq.headers.*;
...
byte [] bytes = ...
DataInput in = new DataInputStream (new ByteArrayInputStream (bytes));
MQHeaderIterator it = new MQHeaderIterator (in, CMQC.MQENC_NATIVE,
  CMQC.MQCCSI_DEFAULT);
```

Wiersz rozpoczynający MQHeaderIterator może zostać zastąpiony

```
MQDLH dlh = new MQDLH (in, CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
// or any other header type
```

W tym przykładzie zapis do tablicy bajtów przy użyciu strumienia DataOutputjest następujący:

```
MQHeader header = ... // Could be any header type
ByteArrayOutputStream out = new ByteArrayOutputStream ();

header.write (new DataOutputStream (out), CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
byte [] bytes = out.toByteArray ();
```

Podczas pracy ze strumieniami w ten sposób należy zachować ostrożność, aby użyć poprawnych wartości dla kodowania i argumentów characterSet . Podczas odczytywania nagłówków należy określić kodowanie i identyfikator CCSID, z którym oryginalnie została zapisana treść bajtowa. Podczas zapisywania nagłówków należy określić kodowanie i identyfikator CCSID, który ma zostać wyproduowany. Konwersja danych jest wykonywana automatycznie przez klasy nagłówka.

Tworzenie klas dla nowych typów nagłówków przy użyciu klas produktu WebSphere MQ dla języka Java

Klasy Java można tworzyć dla typów nagłówków, które nie są dostarczane z klasami produktu WebSphere MQ dla języka Java.

Aby dodać klasę Java reprezentującą nowy typ nagłówka, który może być używany w taki sam sposób, jak dowolna klasa nagłówka dostarczana z klasami produktu WebSphere MQ dla języka Java, należy utworzyć klasę implementuającą interfejs MQHeader. Najprostszym podejściem jest rozszerzenie klasy com.ibm.mq.headers.impl.Header . W tym przykładzie jest tworzony w pełni funkcjonalna klasa reprezentująca strukturę nagłówka MQTM. Nie trzeba dodawać pojedynczych metod pobierających i ustawiających dla każdego pola, ale jest to użyteczna wygodna dla użytkowników klasy nagłówka. Ogólne metody getValue i setValue , które pobierają łańcuch dla nazwy pola, będą działać dla wszystkich pól zdefiniowanych w typie nagłówka. Odziedziczone metody odczytu, zapisu i wielkości umożliwią odczytywanie i zapisywanie instancji nowego typu nagłówka oraz poprawnie obliczą wielkość nagłówka w oparciu o jego definicję pola. Definicja typu jest tworzona tylko raz, jednak tworzone są wiele instancji tej klasy nagłówka. Aby nowa definicja nagłówka była dostępna do dekodowania przy użyciu klas MQHeaderIterator lub MQHeaderList , należy zarejestrować ją za pomocą MQHeaderRegistry. Należy jednak pamiętać, że klasa nagłówka MQTM jest już w rzeczywistości udostępniona w tym pakiecie i zarejestrowana w rejestrze domyślnym.

```
import com.ibm.mq.headers.impl.Header;
import com.ibm.mq.headers.impl.HeaderField;
import com.ibm.mq.headers.CMQC;

public class MQTM extends Header {
    final static HeaderType TYPE = new HeaderType ("MQTM");
    final static HeaderField StrucId = TYPE.addMQChar ("StrucId", CMQC.MQTM_STRUC_ID);
    final static HeaderField Version = TYPE.addMQLong ("Version", CMQC.MQTM_VERSION_1);
    final static HeaderField QName = TYPE.addMQChar ("QName", CMQC.MQ_Q_NAME_LENGTH);
    final static HeaderField ProcessName = TYPE.addMQChar ("ProcessName",
      CMQC.MQ_PROCESS_NAME_LENGTH);
    final static HeaderField TriggerData = TYPE.addMQChar ("TriggerData",
      CMQC.MQ_TRIGGER_DATA_LENGTH);
    final static HeaderField ApplType = TYPE.addMQLong ("ApplType");
    final static HeaderField ApplId = TYPE.addMQChar ("ApplId", 256);
    final static HeaderField EnvData = TYPE.addMQChar ("EnvData", 128);
    final static HeaderField UserData = TYPE.addMQChar ("UserData", 128);

    protected MQTM (HeaderType type){
        super (type);
    }
    public String getStrucId () {
        return getStringValue (StrucId);
    }
    public int getVersion () {
        return getIntValue (Version);
    }
    public String getQName () {
        return getStringValue (QName);
    }
    public void setQName (String value) {
        setStringValue (QName, value);
    }
    // ...Add convenience getters and setters for remaining fields in the same way.
}
```

Obsługa komunikatów PCF z klasami produktu WebSphere MQ dla języka Java

Klasy Java są udostępniane do tworzenia i analizowania komunikatów ustrukturyzowanych PCF, a także do ułatwiania wysyłania żądań PCF i zbierania odpowiedzi PCF.

Klasy PCFMessage & MQCFGR reprezentują tablice struktur parametrów PCF. Udostępniają one podręczne metody dodawania i pobierania parametrów PCF.

Struktury parametrów PCF są reprezentowane przez klasy MQCFH, MQCFIN, MQCFIN64, MQCFST, MQCFBS, MQCFIL, MQCFIL64 MQCFSL i MQCFGR. Te podstawowe interfejsy operacyjne są współużytkowane:

- Metody do odczytu i zapisu treści wiadomości: read (), write () i size ()
- Metody służące do manipulowania parametrami: getValue (), setValue (), getParameter () i inne
- Metoda enumeratora.nextParameter (), która analizuje treść PCF w komunikacie MQMessage

Parametr filtru PCF jest używany w komendach inquire w celu udostępnienia funkcji filtrowania. W enkapsulacji w następujących klasach:

- MQCFIF-filtr liczby całkowitej
- MQCFSF-filtr łańcuchów
- MQCFBF-filtr bajtów

Do zarządzania połączeniem z menedżerem kolejek, kolejką serwera komend i powiązaną kolejką odpowiedzi udostępniono dwa klasy agenta, PCFAgent i PCFMessageAgent . Agent PCFMessageAgent rozszerza agent PCFAgent i zwykle powinien być używany do jego preferencji. Klasa PCFMessageAgent przekształca odebrane komunikaty MQMessages i przekazuje je z powrotem do programu wywołującego jako tablicę PCFMessage. Agent PCFAgent zwraca tablicę komunikatów MQMessages, którą należy przeanalizować przed użyciem.

Obsługa właściwości komunikatu w klasach produktu WebSphere MQ dla języka Java

Wywołania funkcji do obsługi uchwytów komunikatów nie mają odpowiednika w klasach produktu WebSphere MQ dla języka Java. Aby ustawić, zwrócić lub usunąć właściwości uchwytu komunikatu, należy użyć metod klasy MQMessage.

Ogólne informacje na temat właściwości komunikatu zawiera sekcja “Nazwy właściwości” na stronie 19.

W produkcie WebSphere MQ classes for Java access to messages jest za pośrednictwem klasy MQMessage. Uchwyty komunikatów nie są więc udostępniane w środowisku Java i nie ma odpowiednika dla funkcji WebSphere MQ wywołują wywołania MQCRTMH, MQDLTMH, MQMHBUF i MQBUFMH

Aby ustawić właściwości uchwytu komunikatu w interfejsie proceduralnym, należy użyć wywołania MQSETMP. W klasach produktu WebSphere MQ dla języka Java należy użyć odpowiedniej metody klasy MQMessage:

- Właściwość setBoolean
- Właściwość setByte
- Właściwość setBytes
- Właściwość setShort
- Właściwość setInt
- setInt2Property
- setInt4Property
- setInt8Property
- Właściwość setLong
- Właściwość setFloat
- Właściwość setDouble
- Właściwość setString
- Właściwość setObject

Są one czasami nazywane wspólnie metodami *set*property* .

Aby zwrócić wartość właściwości uchwytu komunikatu w interfejsie proceduralnym, należy użyć wywołania MQINQMP. W klasach produktu WebSphere MQ dla języka Java należy użyć odpowiedniej metody klasy MQMessage:

- Właściwość getBoolean
- Właściwość getByte
- Właściwość getBytes
- Właściwość getShort
- Właściwość getInt
- getInt2Property
- getInt4Property
- getInt8Property
- Właściwość getLong
- Właściwość getFloat
- Właściwość getDouble
- Właściwość getString
- Właściwość getObject

Są one czasami nazywane wspólnie metodami *get*property* .

Aby usunąć wartość właściwości uchwytu komunikatu w interfejsie proceduralnym, należy użyć wywołania MQDLTMP. W klasach produktu WebSphere MQ dla języka Java należy użyć metody deleteProperty klasy MQMessage.

Obsługa błędów w klasach produktu WebSphere MQ dla języka Java

Obsługa błędów związanych z klasami produktu WebSphere MQ dla języka Java za pomocą bloków Java `try` i `catch` .

Metody w interfejsie Java nie zwracają kodu zakończenia i kodu przyczyny. Zamiast tego zgłaszają one wyjątek, gdy kod zakończenia i kod przyczyny wynikające z wywołania WebSphere MQ nie są jednocześnie zerami. Upraszcza to logikę programu, tak aby nie było konieczne sprawdzanie kodów powrotu po każdym wywołaniu funkcji WebSphere MQ. Możesz zdecydować, w jakich punktach w programie chcesz poradzić sobie z możliwością awarii. W tych punktach można surround kodu za pomocą bloków `try` i `catch` , jak w następującym przykładzie:

```
try {
    myQueue.put(messageA,putMessageOptionsA);
    myQueue.put(messageB,putMessageOptionsB);
}
catch (MQException ex) {
    // This block of code is only executed if one of
    // the two put methods gave rise to a non-zero
    // completion code or reason code.
    System.out.println("An error occurred during the put operation:" +
                           "CC = " + ex.completionCode +
                           "RC = " + ex.reasonCode);
    System.out.println("Cause exception:" + ex.getCause() );
}
```

Kody przyczyny wywołania produktu WebSphere MQ zgłoszone z powrotem w wyjątkach języka Java dla systemu z/OS są udokumentowane w sekcji Kody przyczyny dla systemu z/OS i Kody przyczyn dla wszystkich innych platform.

Wyjątki zgłaszane podczas działania klas produktu WebSphere MQ dla aplikacji Java są również zapisywane w dzienniku. Jednak aplikacja może wywołać metodę MQException.logExclude(), aby zapobiec protokołowaniu wyjątków powiązanych z określonym kodem przyczyny. Można to zrobić w sytuacjach, w których oczekiwane jest zgłoszenie wielu wyjątków powiązanych z określonym kodem przyczyny i nie chcesz, aby dziennik był wypełniany tymi wyjątkami. Jeśli na przykład aplikacja próbuje pobrać komunikat z kolejki za każdym razem, gdy iteruje on pętlę, a w przypadku większości tych prób nie oczekuje się, aby w kolejce nie było odpowiedniego komunikatu, można zapobiec zarejestowaniu wyjątków powiązanych z kodem przyczyny MQRC_NO_MSG_AVAILABLE. Jeśli aplikacja wcześniej uniemożliwiła rejestrowanie wyjątków powiązanych z określonym kodem przyczyny, może zezwolić na ponowne zalogowanie się tych wyjątków przez wywołanie metody MQException.logInclude().

Czasami kod przyczyny nie przekazuje wszystkich szczegółów związanych z błędem. W przypadku każdego zgłoszonego wyjątku aplikacja powinna sprawdzić powiązany wyjątek. Sam powiązany wyjątek może mieć inny powiązany wyjątek, a więc połączone wyjątki tworzą łańcuch prowadzący do pierwotnego problemu bazowego. Powiązany wyjątek jest implementowany przy użyciu mechanizmu połączonych wyjątków klasy java.lang.Throwable , a aplikacja uzyskuje powiązany wyjątek, wywołując metodę Throwable.getCause(). Z wyjątku, który jest instancją wyjątku MQException, MQException.getCause() pobiera podstawową instancję klasy com.ibm.mq.jmqi.JmqiException, a metoda getCause z tego wyjątku pobiera bazowy wyjątek java.lang.Exception , który spowodował błąd.

Domyślnie klasa MQException automatycznie strumieniuje wyjątki do pliku System.err, który zwykle jest kierowany do konsoli. Aby zatrzymać wyjątki pojawiające się na konsoli, należy umieścić w aplikacji wiersz w celu ustawienia pliku MQException.log= null.

Uzyskiwanie i ustawianie wartości atrybutów w klasach produktu WebSphere MQ dla języka Java

Metody getXXX() i setXXX() są dostępne dla wielu wspólnych atrybutów. Dostęp do innych osób można uzyskać za pomocą metod ogólnych inquire () i set ().

W przypadku wielu wspólnych atrybutów klasy MQManagedObject, MQDestination, MQQueue, MQTopic, MQProcess i MQQueueManager zawierają metody getXXX() i setXXX(). Metody te pozwalają na uzyskanie i ustawienie ich wartości atrybutów. Należy pamiętać, że w przypadku obiektu MQDestination, MQQueue i MQTopic metody działają tylko wtedy, gdy podczas otwierania obiektu określone zostaną odpowiednie opcje zapytania i ustawienia.

W przypadku mniej wspólnych atrybutów klasy MQQueueManager, MQDestination, MQQueue, MQTopic i MQProcess dziedziczą wszystkie klasy z klasy o nazwie MQManagedObject. Ta klasa definiuje interfejsy inquire () i set ().

Po utworzeniu nowego obiektu menedżera kolejek za pomocą operatora *nowy* jest on automatycznie otwierany na potrzeby zapytania. Jeśli do uzyskania dostępu do obiektu procesu używana jest metoda accessProcess(), obiekt ten jest automatycznie otwierany w celu uzyskania informacji. Jeśli do uzyskania dostępu do obiektu kolejki używana jest metoda accessQueue(), obiekt ten *nie* jest automatycznie otwierany na potrzeby operacji sprawdzania lub ustawiania operacji. Jest to spowodowane tym, że dodanie tych opcji automatycznie może spowodować problemy z niektórymi typami kolejek zdalnych. Aby użyć metod inquire, set, getXXXi setXXX w kolejce, należy określić odpowiednie opcje zapytania i ustawienia w parametrze openOptions metody accessQueue(). To samo dotyczy obiektów docelowych i obiektów tematów.

Metody zapytania i ustawiania przyjmują trzy parametry:

- tablica selektorów
- Tablica intAttrs
- Tablica charAttrs

Nie są potrzebne parametry SelectorCount, IntAttrCount i CharAttrLength, które znajdują się w tabeli MQINQ, ponieważ długość tablicy w języku Java jest zawsze znana. W poniższym przykładzie przedstawiono sposób wykonania zapytania w kolejce:

```
// inquire on a queue
final static int MQIA_DEF_PRIORITY = 6;
final static int MQCA_Q_DESC = 2013;
final static int MQ_Q_DESC_LENGTH = 64;

int[] selectors  = new int[2];
int[] intAttrs   = new int[1];
byte[] charAttrs = new byte[MQ_Q_DESC_LENGTH]

selectors[0] = MQIA_DEF_PRIORITY;
selectors[1] = MQCA_Q_DESC;

queue.inquire(selectors,intAttrs,charAttrs);

System.out.println("Default Priority = " + intAttrs[0]);
System.out.println("Description : " + new String(charAttrs,0));
```

Programy wielowątkowe w języku Java

Środowisko wykonawcze programów Java jest z natury wielowątkowe. Klasy WebSphere MQ dla języka Java umożliwiają współużytkowanie obiektu menedżera kolejek przez wiele wątków, ale zapewnia, że wszystkie uprawnienia dostępu do docelowego menedżera kolejek są synchronizowane.

Programy wielowątkowe są trudne do uniknięcia w języku Java. Rozważmy prosty program, który łączy się z menedżerem kolejek i otwiera kolejkę przy starcie. Program wyświetla na ekranie pojedynczy przycisk. Gdy użytkownik kliknie ten przycisk, program pobierze komunikat z kolejki.

Środowisko wykonawcze programów Java jest z natury wielowątkowe. W związku z tym inicjowanie aplikacji występuje w jednym wątku, a kod wykonywany w odpowiedzi na naciśnięcie przycisku jest wykonywany w osobnym wątku (wątek interfejsu użytkownika).

W przypadku klienta MQI produktu WebSphere MQ opartego na języku C może to spowodować problem, ponieważ w wielu wątkach istnieją ograniczenia dotyczące współużytkowania uchwytów. Klasy WebSphere MQ classes for Java odprężają to ograniczenie, umożliwiając współużytkowanie obiektu menedżera kolejek (oraz powiązanej z nim kolejki, tematu i obiektów procesu) przez wiele wątków.

Implementacja klas produktu WebSphere MQ dla języka Java zapewnia, że dla określonego połączenia (instancja obiektuMQQueueManager) wszystkie uprawnienia dostępu do docelowego menedżera kolejek produktu WebSphere MQ są zsynchronizowane. Wątek, który chce wydać wywołanie do menedżera kolejek, jest blokowany do momentu zakończenia wszystkich innych wywołań w toku dla tego połączenia. Jeśli wymagany jest jednoczesny dostęp do tego samego menedżera kolejek z wielu wątków w programie, należy utworzyć nowy obiekt MQQueueManager dla każdego wątku, który wymaga współbieżnego dostępu. (Jest to równoznaczne z wywołaniem oddzielnego wywołania MQCONN dla każdego wątku).

Uwaga: Instancje klasy `com.ibm.mq.MQGetMessageOptions` nie mogą być współużytkowane między wątkami, które jednocześnie żądają komunikatów. Instancje tej klasy są aktualizowane przy użyciu danych podczas odpowiadania na odpowiednie żądanie MQGET, co może spowodować nieoczekiwane konsekwencje, gdy wiele wątków działa jednocześnie w tej samej instancji obiektu.

Używanie wyjść kanału w klasach produktu WebSphere MQ dla języka Java

Przegląd sposobów korzystania z wyjść kanału w aplikacji przy użyciu klas WebSphere MQ dla języka Java.

W poniższych tematach opisano sposób pisania wyjścia kanału w języku Java, sposób jego przypisywania oraz sposób przekazywania danych do niego. Następnie opisują, w jaki sposób używać wyjść kanału napisanych w języku C oraz jak używać sekwencji wyjść kanału.

Aby załadować klasę wyjścia kanału, aplikacja musi mieć poprawne uprawnienia zabezpieczeń.

Tworzenie wyjścia kanału w klasach produktu WebSphere MQ dla języka Java

Istnieje możliwość udostępnienia własnych wyjść kanału przez zdefiniowanie klasy Java, która implementuje odpowiedni interfejs.

Aby zaimplementować wyjście, należy zdefiniować nową klasę Java, która implementuje odpowiedni interfejs. W pakiecie com.ibm.mq.exits są zdefiniowane trzy interfejsy wyjścia:

- WMQSendExit
- WMQReceiveExit
- WMQSecurityExit

Uwaga: Wyjścia kanału są obsługiwane tylko dla połączeń klientów. Nie są one obsługiwane w przypadku połączeń powiązań. Nie można użyć wyjścia kanału Java poza klasami produktu WebSphere MQ dla języka Java, na przykład w przypadku korzystania z aplikacji klienckiej napisanej w języku C.

Wszystkie szyfrowanie SSL zdefiniowane dla połączenia jest wykonywane *po* wywołaniu wyjścia wysyłania i zabezpieczeń. Podobnie deszyfrowanie jest wykonywane *przed* wywołaniem odbierania i kończy działanie zabezpieczeń.

Poniższy przykład definiuje klasę, która implementuje wszystkie trzy interfejsy:

```
public class MyMQExits implements
WMQSendExit, WMQReceiveExit, WMQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method comes from the send exit interface
  public ByteBuffer channelSendExit(
MQCXP channelExitParms,
                                    MQCD channelDefinition,
                                    ByteBuffer agentBuffer)
  {
    // Fill in the body of the send exit here
  }
    // This method comes from the receive exit interface
  public ByteBuffer channelReceiveExit(
MQCXP channelExitParms,
                                       MQCD channelDefinition,
                                       ByteBuffer agentBuffer)
  {
    // Fill in the body of the receive exit here
  }
    // This method comes from the security exit interface
  public ByteBuffer channelSecurityExit(
MQCXP channelExitParms,
                                        MQCD channelDefinition,
```

```
                                          ByteBuffer agentBuffer)
  {
    // Fill in the body of the security exit here
  }
}
```

Każde wyjście jest przekazywane do obiektu MQCXP i do obiektu MQCD. Obiekty te reprezentują struktury MQCXP i MQCD zdefiniowane w interfejsie proceduralnym.

Każda klasa wyjścia, którą należy napisać, musi mieć konstruktor. Może to być konstruktor domyślny lub inny, który pobiera argument łańcuchowy. Jeśli pobiera on łańcuch, dane użytkownika zostaną przekazane do klasy wyjścia podczas jej tworzenia. Jeśli klasa wyjścia zawiera zarówno konstruktor domyślny, jak i konstruktor z jednym argumentem, to pojedynczy konstruktor argumentów ma priorytet.

W przypadku wyjścia wysyłania i zabezpieczeń kod wyjścia musi zwracać dane, które mają zostać wysłane do serwera. W przypadku wyjścia odbierania kod wyjścia musi zwracać zmodyfikowane dane, które mają być interpretowane przez produkt WebSphere MQ .

Najprostszym możliwym organem wyjściowym jest:

```
{ return agentBuffer; }
```

Nie zamykać menedżera kolejek z poziomu wyjścia kanału.

Korzystanie z istniejących klas wyjścia kanału

W wersjach produktu WebSphere MQ wcześniejszych niż 7.0można zaimplementować te wyjścia przy użyciu interfejsów MQSendExit, MQReceiveExiti MQSecurityExit, tak jak w poniższym przykładzie. Ta metoda pozostaje poprawna, ale nowa metoda jest preferowana w celu zwiększenia funkcjonalności i wydajności.

```
public class MyMQExits implements MQSendExit, MQReceiveExit, MQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method comes from the send exit
  public byte[] sendExit(MQChannelExit channelExitParms,
                         MQChannelDefinition channelDefParms,
                         byte agentBuffer[])
  {
    // Fill in the body of the send exit here
  }
    // This method comes from the receive exit
  public byte[] receiveExit(MQChannelExit channelExitParms,
                            MQChannelDefinition channelDefParms,
                            byte agentBuffer[])
  {
    // Fill in the body of the receive exit here
  }
    // This method comes from the security exit
  public byte[] securityExit(MQChannelExit channelExitParms,
                             MQChannelDefinition channelDefParms,
                             byte agentBuffer[])
  {
    // Fill in the body of the security exit here
  }
}
```

Przypisywanie wyjścia kanału w programie IBM WebSphere MQ classes for Java

Wyjście kanału można przypisać za pomocą programu IBM WebSphere MQ classes for Java.

W produkcie IBM WebSphere MQ classes for Javanie ma bezpośredniego odpowiednika dla kanału IBM WebSphere MQ . Wyjścia kanału są przypisywane do menedżera MQQueueManager. Na przykład, po zdefiniowaniu klasy, która implementuje interfejs WMQSecurityExit , aplikacja może korzystać z wyjścia zabezpieczeń na jeden z czterech sposobów:

- Poprzez przypisanie instancji klasy do pola MQEnvironment.channelSecurityExit przed utworzeniem obiektu MQQueueManager .

- Ustawiając wartość w polu MQEnvironment.channelSecurityExit na łańcuch reprezentujący klasę wyjścia zabezpieczeń przed utworzeniem obiektu MQQueueManager .
- Tworząc parę klucz/wartość we właściwościach hashtable przekazanej do MQQueueManager z kluczem CMQC.SECURITY_EXIT_PROPERTY
- Korzystanie z tabeli definicji kanału klienta (CCDT)

Każde wyjście przypisane przez ustawienie pola MQEnvironment.channelSecurityExit na łańcuch, utworzenie pary klucz/wartość w tabeli mieszającej właściwości lub przy użyciu tabeli definicji kanału klienta, musi zostać zapisane z konstruktorem domyślnym. Wyjście przypisane jako instancja klasy nie wymaga konstruktora domyślnego, w zależności od aplikacji.

Aplikacja może używać wyjścia wysyłania lub odbierania w podobny sposób. Na przykład poniższy fragment kodu przedstawia sposób użycia zabezpieczeń, wysyłania i odbierania wyjść implementowanych w klasie MyMQExits, która została wcześniej zdefiniowana przy użyciu środowiska MQEnvironment:

```
MyMQExits myexits = new MyMQExits();
MQEnvironment.channelSecurityExit = myexits;
MQEnvironment.channelSendExit = myexits;
MQEnvironment.channelReceiveExit = myexits;
:
MQQueueManager jupiter = new MQQueueManager("JUPITER");
```

Jeśli do przypisania wyjścia kanału używana jest więcej niż jedna metoda, kolejność wykonywania operacji jest następująca:

1. Jeśli adres URL tabeli definicji kanału klienta jest przekazywany do menedżera kolejek MQQueueManager, treść definicji kanału klienta określa wyjścia kanału, które mają być używane, a wszystkie definicje wyjścia w środowisku MQEnvironment lub właściwości hashtable właściwości są ignorowane.
2. Jeśli nie jest przekazywany adres URL CCDT, scalane są definicje wyjścia ze środowiska MQEnvironment i tabeli mieszającej.
   - Jeśli ten sam typ wyjścia jest zdefiniowany zarówno w środowisku MQEnvironment, jak i w tabeli mieszającej, używana jest definicja w tabeli mieszającej.
   - Jeśli określono równoważne stare i nowe typy wyjścia (na przykład pole sendExit , które może być używane tylko dla typu wyjścia używanego w wersjach produktu IBM WebSphere MQ wcześniejszych niż wersja 7.0i channelSendExit, które mogą być używane dla dowolnego wyjścia nadawczego), zamiast starego wyjścia jest używane nowe wyjście (channelSendExit).

Jeśli zadeklarowano wyjście kanału jako łańcuch, należy włączyć program IBM WebSphere MQ w celu zlokalizowania programu obsługi wyjścia kanału. Można to zrobić na różne sposoby, w zależności od środowiska, w którym aplikacja jest uruchomiona, oraz w jaki sposób programy obsługi wyjścia kanału są spakowane.

- W przypadku aplikacji działającej na serwerze aplikacji należy zapisać pliki znajdujące się w katalogu podanym w sekcji Tabela 88 na stronie 707 lub spakowane w plikach JAR, do których odwołuje się produkt **exitClasspath**.
- W przypadku aplikacji, która nie jest uruchomiona na serwerze aplikacji, mają zastosowanie następujące reguły:
  - Jeśli klasy wyjścia kanału są spakowane w oddzielnych plikach JAR, te pliki JAR muszą zostać dołączone do produktu **exitClasspath**.
  - Jeśli klasy wyjścia kanału nie są spakowane w plikach JAR, pliki klas mogą być przechowywane w katalogu podanym w sekcji Tabela 88 na stronie 707 lub w dowolnym katalogu w ścieżce klasy systemowej maszyny JVM lub w katalogu **exitClasspath**.

Właściwość **exitClasspath** może być określona na cztery sposoby. W kolejności priorytetów są to następujące sposoby:

1. Właściwość systemowa com.ibm.mq.exitClasspath (zdefiniowana w wierszu komend przy użyciu opcji `-D`).
2. Sekcja exitPath w pliku `mqclient.ini`

3. Pozycja tabeli mieszającej z kluczem CMQC.EXIT_CLASSPATH_PROPERTY
4. Zmienna MQEnvironment **exitClasspath**

Wiele ścieżek należy rozdzielić za pomocą znaku java.io.File.pathSeparator .

Tabela 88. Katalog dla programów obsługi wyjścia kanału

Platforma	**Katalog**
AIX, HP-UX, Linuxi System Solaris	`/var/mqm/exits` (32-bitowe programy obsługi wyjścia kanału) `/var/mqm/exits64` (64-bitowe programy obsługi wyjścia kanału)
Windows	*katalog_danych_instalacji*\exits
Uwaga: *katalog_danych_instalacji* to katalog, który został wybrany dla plików danych programu IBM WebSphere MQ podczas instalacji. Katalog domyślny to `C:\Program Files\IBM\WebSphere MQ`.	

Przekazywanie danych do wyjść kanału w klasach produktu WebSphere MQ dla języka Java

Dane można przekazywać do wyjść kanału i zwracać dane z wyjść kanału do aplikacji.

Parametr agentBuffer

W przypadku wyjścia wysyłania parametr *agentBuffer* zawiera dane, które mają zostać wysłane. W przypadku wyjścia odbierania lub wyjścia zabezpieczeń parametr *agentBuffer* zawiera dane, które właśnie zostały odebrane. Parametr length nie jest wymagany, ponieważ wyrażenie agentBuffer.limit () wskazuje długość tablicy.

W przypadku wyjścia wysyłania i zabezpieczeń kod wyjścia musi zwracać dane, które mają zostać wysłane do serwera. W przypadku wyjścia odbierania kod wyjścia musi zwracać zmodyfikowane dane, które mają być interpretowane przez produkt WebSphere MQ .

Najprostszym możliwym organem wyjściowym jest:

```
{ return agentBuffer; }
```

Wyjścia kanału są wywoływane z buforem, w którym znajduje się tablica zapasowa. Aby uzyskać najlepszą wydajność, wyjście powinno zwrócić bufor z tablicą bazową.

Dane użytkownika

Jeśli aplikacja łączy się z menedżerem kolejek, ustawiając opcję channelSecurityExit, channelSendExit lub channelReceiveExit, 32 bajty danych użytkownika mogą być przekazywane do odpowiedniej klasy wyjścia kanału, gdy jest wywoływana, za pomocą pól channelSecurityExitUserData, channelSendExitUserData lub channelReceiveExitUserData. Te dane użytkownika są dostępne dla klasy wyjścia kanału, ale są odświeżane za każdym razem, gdy wywoływane jest wyjście. W związku z tym wszelkie zmiany wprowadzone w danych użytkownika w wyjściu kanału zostaną utracone. Aby trwałe zmiany danych w wyjściu kanału były wprowadzane, należy użyć obszaru exitUserproduktu MQCXP. Dane w tym polu są przechowywane między wywołaniami wyjścia.

Jeśli aplikacja ustawi wartości securityExit, sendExitlub receiveExit, dane użytkownika nie mogą być przekazywane do tych klas wyjścia kanału.

Jeśli aplikacja korzysta z tabeli definicji kanału klienta (CCDT) w celu nawiązania połączenia z menedżerem kolejek, wszystkie dane użytkownika określone w definicji kanału połączenia klienta są przekazywane do klas wyjścia kanału podczas ich wywołania. Więcej informacji na temat korzystania z tabeli definicji kanału klienta zawiera sekcja “Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM WebSphere MQ classes for Java” na stronie 689.

Używanie wyjść kanału nie napisanych w języku Java z klasami produktu WebSphere MQ dla języka Java

W jaki sposób używać programów obsługi wyjścia kanału napisanych w języku C z aplikacji Java.

W produkcie WebSphere MQ , wersja 7.0, można określić nazwę programu obsługi wyjścia kanału napisanego w języku C jako łańcuch przekazany do pól channelSecurityExit, channelSendExit lub channelReceivew obiekcie MQEnvironment lub w tabeli Hashtable. Nie można jednak użyć wyjścia kanału napisanego w języku Java w aplikacji napisanej w innym języku.

Podaj nazwę programu obsługi wyjścia w formacie `library(function)` i upewnij się, że położenie programu obsługi wyjścia jest zawarte w zmiennej środowiskowej ścieżki.

Więcej informacji na temat pisania wyjścia kanału w języku C zawiera sekcja “Programy obsługi wyjścia kanału dla kanałów przesyłania komunikatów” na stronie 406.

Korzystanie z zewnętrznych klas wyjścia

W wersjach produktu WebSphere MQ wcześniejszych niż wersja 7.0udostępniono trzy klasy, które umożliwiały korzystanie z wyjść kanału napisanych w językach innych niż Java:

- MQExternalSecurityWyjście, które implementuje interfejs MQSecurityExit
- Wyjście MQExternalSend, które implementuje interfejs MQSendExit
- Wyjście MQExternalReceive, które implementuje interfejs MQReceiveExit

Korzystanie z tych klas pozostaje poprawne, ale preferowana jest nowa metoda.

Aby użyć wyjścia zabezpieczeń, które nie jest napisane w języku Java, aplikacja najpierw musiała utworzyć obiekt wyjścia MQExternalSecurity. Podana aplikacja, jako parametry w konstruktorze wyjścia MQExternalSecurity, nazwa biblioteki zawierającej wyjście zabezpieczeń, nazwa punktu wejścia dla wyjścia zabezpieczeń oraz dane użytkownika, które mają być przekazane do wyjścia zabezpieczeń podczas jego wywołania. Programy obsługi wyjścia kanału, które nie są zapisywane w języku Java, zostały zapisane w katalogu podanym w sekcji Tabela 88 na stronie 707.

Korzystanie z sekwencji wyjść wysyłania lub odbierania kanału w klasach WebSphere MQ dla języka Java

Klasy produktu WebSphere MQ dla aplikacji Java mogą używać sekwencji wyjść wysyłania lub odbierania kanału, które są uruchamiane w ramach dziedziczenia.

Aby użyć sekwencji wysyłania wyjść, aplikacja może utworzyć listę lub łańcuch zawierający wyjścia wysyłania. Jeśli używana jest lista, każdy element listy może mieć jedną z następujących wartości:

- Instancja klasy zdefiniowanej przez użytkownika, która implementuje interfejs WMQSendExit
- Instancja klasy zdefiniowanej przez użytkownika, która implementuje interfejs MQSendExit (w przypadku wyjścia wysyłania napisanego w języku Java)
- Instancja klasy wyjścia MQExternalSend(w przypadku wyjścia wysyłania nie napisanego w języku Java)
- Instancja klasy łańcucha MQSendExit
- Instancja klasy String

Lista nie może zawierać innej listy.

Aplikacja może korzystać z sekwencji wyjść odbierania w podobny sposób.

Jeśli używany jest łańcuch, musi składać się z jednej lub większej liczby definicji wyjścia oddzielonych przecinkami, z których każda może być nazwą klasy Java lub programu C w formacie `library(function)`.

Następnie aplikacja przypisuje obiekt List lub String do pola MQEnvironment.channelSendExit przed utworzeniem obiektu MQQueueManager .

Kontekst informacji przekazywanych do wyjść jest wyłącznie w obrębie domeny wyjść. Na przykład, jeśli wyjście języka Java i wyjście C są połączone łańcuchowo, obecność wyjścia Java nie ma wpływu na wyjście C.

Korzystanie z klas łańcucha wyjścia

W wersjach produktu WebSphere MQ wcześniejszych niż wersja 7.0udostępniono dwie klasy umożliwiające sekwencje wyjść:

- MQSendExitŁańcuch, który implementuje interfejs MQSendExit
- MQReceiveExitŁańcuch, który implementuje interfejs MQReceiveExit

Korzystanie z tych klas pozostaje poprawne, ale preferowana jest nowa metoda. Użycie klas WebSphere MQ dla interfejsów Java oznacza, że aplikacja nadal ma zależność od `com.ibm.mq.jar` . Jeśli nowy zestaw interfejsów w pakiecie `com.ibm.mq.exits` jest używany, nie ma zależności od produktu `com.ibm.mq.jar`.

Aby użyć sekwencji wysyłania wyjść, aplikacja utworzyła listę obiektów, w której każdy obiekt był jednym z następujących obiektów:

- Instancja klasy zdefiniowanej przez użytkownika, która implementuje interfejs MQSendExit (w przypadku wyjścia wysyłania napisanego w języku Java)
- Instancja klasy wyjścia MQExternalSend(w przypadku wyjścia wysyłania nie napisanego w języku Java)
- Instancja klasy łańcucha MQSendExit

Aplikacja utworzyła obiekt łańcucha MQSendExit, przekazując tę listę obiektów jako parametr w konstruktorze. Aplikacja przypisałaby następnie obiekt MQSendExitdo pola MQEnvironment.sendExit przed utworzeniem obiektu MQQueueManager .

Kompresja kanału w klasach produktu WebSphere MQ dla języka Java

Kompresowanie danych, które przepływa na kanał, może poprawić wydajność kanału i zmniejszyć ruch w sieci. IBM WebSphere MQ classes for Java Użyj funkcji kompresji wbudowanej w program IBM WebSphere MQ.

Za pomocą funkcji dostarczonej z produktem IBM WebSphere MQmożna kompresować dane, które przepływają w kanałach komunikatów i kanałach MQI, a także-w obu typach kanałów-można kompresować dane nagłówka i dane komunikatów niezależnie od siebie. Domyślnie żadne dane nie są kompresowane w kanale. Pełny opis kompresji kanałów, w tym sposób jego implementacji w produkcie IBM WebSphere MQ, zawiera sekcja Kompresja danych (COMPMSG) i Kompresja nagłówka (COMPHDR).

Aplikacja IBM WebSphere MQ classes for Java określa techniki, które mogą być używane do kompresowania nagłówka lub danych komunikatów w połączeniu klienta przez utworzenie obiektu java.util.Collection . Każda technika kompresji jest obiektem typu Integer w kolekcji, a kolejność, w jakiej aplikacja dodaje techniki kompresji do kolekcji, jest to kolejność, w jakiej techniki kompresji są negocjowane z menedżerem kolejek po uruchomieniu połączenia klienckiego. Aplikacja może następnie przypisać kolekcję do pola hdrCompList, dla danych nagłówka lub pola msgCompList, dla danych komunikatu, w klasie MQEnvironment. Gdy aplikacja jest gotowa, może ona uruchomić połączenie klienta, tworząc obiekt MQQueueManager .

Opisane poniżej fragmenty kodu ilustrują opisane podejście. Pierwszy fragment kodu pokazuje, jak zaimplementować kompresję danych nagłówka:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(CMQXC.MQCOMPRESS_SYSTEM));
:
MQEnvironment.hdrCompList = headerComp;
:
MQQueueManager qMgr = new MQQueueManager(QM);
```

Drugi fragment kodu przedstawia sposób implementowania kompresji danych komunikatu:

```
Collection msgComp = new Vector();
msgComp.add(new Integer(CMQXC.MQCOMPRESS_RLE));
msgComp.add(new Integer(CMQXC.MQCOMPRESS_ZLIBHIGH));
:
MQEnvironment.msgCompList = msgComp;
:
MQQueueManager qMgr = new MQQueueManager(QM);
```

W drugim przykładzie techniki kompresji są negocjowane w kolejności RLE, a następnie ZLIBHIGH, gdy rozpoczyna się połączenie z klientem. Wybrana technika kompresji nie może zostać zmieniona w czasie życia obiektu MQQueueManager .

Techniki kompresji dla danych nagłówka i komunikatu, które są obsługiwane zarówno przez klienta, jak i menedżera kolejek w połączeniu klienta, są przekazywane do wyjścia kanału jako kolekcje w polach listy hdrCompi msgCompw obiekcie MQChannelDefinition . Rzeczywiste techniki, które są obecnie używane do kompresowania nagłówka i danych komunikatów w połączeniu klienta, są przekazywane do wyjścia kanału w polach kompresji CurHdri kompresji CurMsgobiektu MQChannelExit .

Jeśli kompresja jest używana w połączeniu z klientem, dane są kompresowane przed przetworami i wyodrębnionymi wyjściami nadawczego kanału po przetworzeniu wszystkich wyjść odbierania kanału. Dane przekazywane do wyjścia wysyłania i odbierania są w związku z tym w stanie skompresowanym.

Więcej informacji na temat określania technik kompresji i technik kompresji można znaleźć w sekcji Klasa com.ibm.mq.MQEnvironment i Interfejs com.ibm.mq.MQC .

Współużytkowanie połączenia TCP/IP w produkcie IBM WebSphere MQ classes for Java

W celu współużytkowania pojedynczego połączenia TCP/IP można utworzyć wiele instancji kanału MQI.

W produkcie IBM WebSphere MQ classes for Javaużywana jest zmienna MQEnvironment.sharingConversations , która umożliwia sterowanie liczbą konwersacji, które mogą współużytkować pojedyncze połączenie TCP/IP.

Atrybut SHARECNV jest najlepszym sposobem podejścia do współużytkowania połączeń. W związku z tym, gdy wartość SHARECNV większa niż 0 jest używana razem z IBM WebSphere MQ classes for Java , nie ma gwarancji, że nowe żądanie połączenia zawsze będzie współużytkować już nawiązane połączenie.

Zestawianie połączeń w klasach produktu WebSphere MQ dla języka Java

Klasy WebSphere MQ dla języka Java umożliwiają łączenie zapasowych połączeń w celu ich ponownego wykorzystania.

Klasy WebSphere MQ classes for Java zapewniają dodatkowe wsparcie dla aplikacji, które zajmują się wieloma połączeniami z menedżerami kolejek produktu WebSphere MQ . Gdy połączenie nie jest już potrzebne, zamiast niszczyć go, można je połączyć i później ponownie wykorzystać. Może to zapewnić znaczące zwiększenie wydajności aplikacji i oprogramowania pośredniego, które łączą się szeregowo z dowolnymi menedżerami kolejek.

Produkt WebSphere MQ udostępnia domyślną pulę połączeń. Aplikacje mogą aktywować lub dezaktywować tę pulę połączeń poprzez zarejestrowanie i wyrejestrowywanie tokenów za pomocą klasy MQEnvironment. Jeśli pula jest aktywna, gdy klasy produktu WebSphere MQ dla języka Java konstruują obiekt MQQueueManager , przeszukuje tę pulę domyślną i ponownie użyje odpowiedniego połączenia. Gdy wystąpi wywołanie MQQueueManager.disconnect (), bazowe połączenie jest zwracane do puli.

Alternatywnie aplikacje mogą skonstruować pulę połączeń menedżera MQSimpleConnectiondla konkretnego zastosowania. Następnie aplikacja może określić tę pulę podczas budowy obiektu MQQueueManager lub przekazać tę pulę do środowiska MQEnvironment w celu użycia jako domyślnej puli połączeń.

Aby zapobiec korzystaniu z zbyt dużej ilości zasobów, można ograniczyć łączną liczbę połączeń, które może obsłużyć obiekt MQSimpleConnectionManager, a także ograniczyć wielkość puli połączeń. Ustawienie limitów jest przydatne, jeśli występują konflikty żądań połączeń w obrębie maszyny JVM.

Domyślnie metoda getMaxConnections () zwraca wartość zero, co oznacza, że nie ma limitu liczby połączeń, które mogą być obsługiwane przez obiekt MQSimpleConnectionManager. Limit można ustawić za pomocą metody setMaxConnections (). Jeśli zostanie ustawiony limit, a limit zostanie osiągnięty, żądanie dalszego połączenia może spowodować zgłoszenie wyjątku MQException z kodem przyczyny MQRC_MAX_CONNS_LIMIT_REACHED.

Sterowanie domyślną pulą połączeń w klasach produktu WebSphere MQ dla języka Java

W tym przykładzie przedstawiono sposób korzystania z domyślnej puli połączeń.

Należy rozważyć zastosowanie następującej przykładowej aplikacji MQApp1:

```
import com.ibm.mq.*;
public class MQApp1
{
      public static void main(String[] args) throws MQException
      {
         for (int i=0; i<args.length; i++) {
            MQQueueManager qmgr=new MQQueueManager(args[i]);
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }
      }
}
```

Program MQApp1 pobiera z wiersza komend listę lokalnych menedżerów kolejek, a następnie łączy się z każdym z kolei i wykonuje pewne operacje. Jednak w przypadku, gdy wiersz komend zawiera wiele razy ten sam menedżer kolejek, można połączyć się z tym samym menedżerem kolejek tylko raz, a także wielokrotnie ponownie wykorzystywać to połączenie.

Klasy WebSphere MQ classes for Java udostępniają domyślną pulę połączeń, która może być używana do wykonania tej czynności. Aby włączyć tę pulę, należy użyć jednej z metod MQEnvironment.addConnectionPoolToken(). Aby wyłączyć pulę, należy użyć metody MQEnvironment.removeConnectionPoolToken().

Następująca przykładowa aplikacja MQApp2jest funkcjonalnie identyczna z aplikacją MQApp1, ale łączy się tylko raz z każdym menedżerem kolejek.

```
import com.ibm.mq.*;
public class MQApp2
{
      public static void main(String[] args) throws MQException
      {
         MQPoolToken token=MQEnvironment.addConnectionPoolToken();

         for (int i=0; i<args.length; i++) {
            MQQueueManager qmgr=new MQQueueManager(args[i]);
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }

         MQEnvironment.removeConnectionPoolToken(token);

      }
}
```

Pierwszy pogrubiony wiersz aktywuje domyślną pulę połączeń przez zarejestrowanie obiektu MQPoolToken w środowisku MQEnvironment.

Konstruktor MQQueueManager wyszukuje w tej puli odpowiednie połączenie i tworzy połączenie z menedżerem kolejek tylko wtedy, gdy nie może znaleźć istniejącego połączenia. Wywołanie qmgr.disconnect() zwraca połączenie z pulą w celu późniejszego ponownego wykorzystania. Te wywołania funkcji API są takie same, jak przykładowa aplikacja MQApp1.

Druga podświetlona linia dezaktywuje domyślną pulę połączeń, która niszczy wszystkie połączenia menedżera kolejek zapisane w puli. Jest to ważne, ponieważ w przeciwnym razie aplikacja zostanie zakończona z pewną liczbą aktywnych połączeń menedżera kolejek w puli. Ta sytuacja może powodować błędy, które pojawiają się w dziennikach menedżera kolejek.

Jeśli aplikacja korzysta z tabeli definicji kanału klienta (CCDT) w celu nawiązania połączenia z menedżerem kolejek, konstruktor MQQueueManager najpierw wyszukuje odpowiednią definicję kanału połączenia klienta. Jeśli zostanie znaleziony, konstruktor przeszukuje domyślną pulę połączeń dla połączenia, które może być użyte dla kanału. Jeśli konstruktor nie może znaleźć odpowiedniego połączenia w puli, przeszukuje tabelę definicji kanału klienta dla następnej odpowiedniej definicji kanału połączenia klienta i kontynuuje działanie zgodnie z opisem poprzednio. Jeśli konstruktor zakończy wyszukiwanie w tabeli definicji kanału klienta i nie znajdzie żadnego odpowiedniego połączenia w puli, konstruktor rozpocznie drugie wyszukiwanie w tabeli. Podczas tego wyszukiwania konstruktor próbuje utworzyć nowe połączenie dla każdej odpowiedniej definicji kanału połączenia klienckiego, a następnie użyje pierwszego połączenia, które ma zostać utworzone.

Domyślna pula połączeń przechowuje maksymalnie dziesięć nieużywanych połączeń i utrzymuje nieużywane połączenia aktywne przez maksymalnie pięć minut. Aplikacja może to zmienić (szczegółowe informacje na ten temat zawiera sekcja “Dostarczanie innej puli połączeń w klasach produktu WebSphere MQ dla języka Java” na stronie 713).

Zamiast używać środowiska MQEnvironment do dostarczania znacznika MQPoolToken, aplikacja może utworzyć własne elementy:

```
MQPoolToken token=new MQPoolToken();
MQEnvironment.addConnectionPoolToken(token);
```

Niektóre aplikacje lub dostawcy oprogramowania pośredniego udostępniają podklasy elementu MQPoolToken w celu przekazania informacji do niestandardowej puli połączeń. Można je utworzyć i przekazać do metody addConnectionPoolToken() w ten sposób, aby dodatkowe informacje mogły zostać przekazane do puli połączeń.

Domyślna pula połączeń i wiele komponentów w klasach produktu WebSphere MQ dla języka Java

W tym przykładzie pokazano, jak dodać lub usunąć element MQPoolTokens ze statycznego zestawu zarejestrowanych obiektów MQPoolToken .

Środowisko MQEnvironment zawiera statyczny zestaw zarejestrowanych obiektów MQPoolToken . Aby dodać lub usunąć element MQPoolTokens z tego zestawu, należy użyć następujących metod:

- MQEnvironment.addConnectionPoolToken()
- MQEnvironment.removeConnectionPoolToken()

Aplikacja może składać się z wielu komponentów, które istnieją niezależnie i wykonują pracę przy użyciu menedżera kolejek. W takiej aplikacji każdy komponent powinien dodać element MQPoolToken do zestawu MQEnvironment ustawionego na jego czas życia.

Na przykład przykładowa aplikacja MQApp3 tworzy dziesięć wątków i uruchamia każdą z nich. Każdy wątek rejestruje swój własny obiekt MQPoolToken, czeka przez pewien czas, a następnie łączy się z menedżerem kolejek. Po rozłączonym wątku usuwa on własny element MQPoolToken.

Domyślna pula połączeń pozostaje aktywna, gdy w zestawie MQPoolTokensznajduje się co najmniej jeden znacznik, więc pozostanie on aktywny przez czas trwania tej aplikacji. Aplikacja nie musi przechowywać obiektu głównego w ogólnej kontroli wątków.

```
import com.ibm.mq.*;
public class MQApp3
{
      public static void main(String[] args)
      {
         for (int i=0; i<10; i++) {
            MQApp3_Thread thread=new MQApp3_Thread(i*60000);
            thread.start();
```

```
        }
    }
}

class MQApp3_Thread extends Thread
{
    long time;

    public MQApp3_Thread(long time)
    {
        this.time=time;
    }

    public synchronized void run()
    {
        MQPoolToken token=MQEnvironment.addConnectionPoolToken();
        try {
            wait(time);
            MQQueueManager qmgr=new MQQueueManager("my.qmgr.1");
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
        }
        catch (MQException mqe) {System.err.println("Error occurred!");}
        catch (InterruptedException ie) {}

        MQEnvironment.removeConnectionPoolToken(token);
    }
}
```

Dostarczanie innej puli połączeń w klasach produktu WebSphere MQ dla języka Java

W tym przykładzie przedstawiono sposób użycia klasy **com.ibm.mq.MQSimpleConnectionManager** w celu dostarczenia innej puli połączeń.

Ta klasa udostępnia podstawowe narzędzia do zestawiania połączeń, a aplikacje mogą korzystać z tej klasy w celu dostosowania zachowania puli.

Po utworzeniu instancji menedżer MQSimpleConnectionmoże zostać określony w konstruktorze MQQueueManager . Menedżer MQSimpleConnectionzarządza następnie połączeniem, który stanowi podstawę dla skonstruowanego menedżera MQQueueManager. Jeśli menedżer MQSimpleConnectionzawiera odpowiednie połączenie z puli, to połączenie jest ponownie wykorzystywane i zwracane do menedżera MQSimpleConnectionpo wywołaniu metody MQQueueManager.disconnect ().

Poniższy fragment kodu demonstruje to zachowanie:

```
MQSimpleConnectionManager myConnMan=new MQSimpleConnectionManager();
myConnMan.setActive(MQSimpleConnectionManager.MODE_ACTIVE);
MQQueueManager qmgr=new MQQueueManager("my.qmgr.1", myConnMan);
:
: (do something with qmgr)
:
qmgr.disconnect();

MQQueueManager qmgr2=new MQQueueManager("my.qmgr.1", myConnMan);
:
: (do something with qmgr2)
:
qmgr2.disconnect();
myConnMan.setActive(MQSimpleConnectionManager.MODE_INACTIVE);
```

Połączenie, które zostało sfałszowane podczas pierwszego konstruktora MQQueueManager , jest zapisywane w pliku myConnMan po wywołaniu metody qmgr.disconnect(). Połączenie jest następnie ponownie wykorzystywane podczas drugiego wywołania konstruktora MQQueueManager .

Drugi wiersz umożliwia włączenie menedżera MQSimpleConnection. Ostatni wiersz wyłącza menedżer MQSimpleConnection, niszcząc wszystkie połączenia znajdujące się w puli. Menedżer MQSimpleConnectionjest domyślnie w trybie MODE_AUTO, który jest opisany w dalszej części tej sekcji.

Menedżer MQSimpleConnectionumożliwia przydzielaniu połączeń na podstawie ostatnio używanej bazy danych i niszczy połączenia w najmniejszym-ostatnio używanym środowisku. Domyślnie

połączenie jest niszczone, jeśli nie zostało użyte przez pięć minut, lub jeśli w puli znajduje się więcej niż dziesięć nieużywanych połączeń. Wartości te można zmienić, wywołując metodę MQSimpleConnectionManager.setTimeout().

Można również skonfigurować menedżer MQSimpleConnection, który będzie używany jako domyślna pula połączeń, jeśli menedżer połączeń nie jest dostępny w konstruktorze MQQueueManager .

Poniższa aplikacja demonstruje, że:

```
import com.ibm.mq.*;
public class MQApp4
{
      public static void main(String []args)
      {
         MQSimpleConnectionManager myConnMan=new MQSimpleConnectionManager();
         myConnMan.setActive(MQSimpleConnectionManager.MODE_AUTO);
         myConnMan.setTimeout(3600000);
         myConnMan.setMaxConnections(75);
         myConnMan.setMaxUnusedConnections(50);
         MQEnvironment.setDefaultConnectionManager(myConnMan);
         MQApp3.main(args);
      }
}
```

Pogrubione wiersze tworzą i konfigurują obiekt menedżera MQSimpleConnection. Konfiguracja wykonuje następujące czynności:

- Kończy połączenia, które nie są używane przez godzinę
- Ogranicza liczbę połączeń zarządzanych przez myConnMan do 75
- Ogranicza liczbę nieużywanych połączeń w puli do 50.
- Ustawia wartość MODE_AUTO, która jest wartością domyślną. Oznacza to, że pula jest aktywna tylko wtedy, gdy jest domyślnym menedżerem połączeń i istnieje co najmniej jeden znacznik w zestawie MQPoolTokens , który jest organizowany przez produkt MQEnvironment.

Nowy menedżer MQSimpleConnectionjest następnie ustawiany jako domyślny menedżer połączeń.

W ostatnim wierszu aplikacja wywołuje komendę MQApp3.main(). Spowoduje to uruchomienie pewnej liczby wątków, w których każdy wątek używa produktu WebSphere MQ niezależnie. Te wątki używają programu myConnMan, gdy są one wymuszane połączenia.

Dostarczanie własnego programu ConnectionManager dla klas produktu WebSphere MQ dla języka Java

Klasy WebSphere MQ classes for Java udostępniają częściową implementację architektury Java EE Connector Architecture, która umożliwia użycie implementacji interfejsu javax.resource.spi.ConnectionManager .

Aplikacje i dostawcy oprogramowania pośredniego mogą udostępniać alternatywne implementacje pul połączeń. Klasy WebSphere MQ classes for Java udostępniają częściową implementację architektury Java EE Connector Architecture. Implementacje interfejsu **javax.resource.spi.ConnectionManager** mogą być używane jako domyślny menedżer połączeń lub być określane w konstruktorze MQQueueManager .

Klasy WebSphere MQ dla języka Java są zgodne z umową zarządzania połączeniem architektury konektora Java EE Connector Architecture. Zapoznaj się z tą sekcją w połączeniu z umową zarządzania połączeniem architektury Java EE Connector Architecture (patrz serwis WWW Sun Java pod adresem https://java.sun.com).

Interfejs ConnectionManager definiuje tylko jedną metodę:

```
package javax.resource.spi;
public interface ConnectionManager {
      Object allocateConnection(ManagedConnectionFactory mcf,
                                ConnectionRequestInfo cxRequestInfo);
}
```

Konstruktor MQQueueManager wywołuje funkcję allocateConnection w odpowiednim produkcie ConnectionManager. Przekazuje odpowiednie implementacje fabryki ManagedConnectionoraz informacje o parametrach ConnectionRequestw celu opisania wymaganego połączenia.

Program ConnectionManager wyszukuje w puli dla obiektu javax.resource.spi.ManagedConnection , który został utworzony z identycznymi obiektami ManagedConnectionFactory i ConnectionRequest. Jeśli program ConnectionManager znajdzie dowolne odpowiednie obiekty ManagedConnection , tworzy obiekt java.util.Set , który zawiera kandydata ManagedConnections. Następnie program ConnectionManager wywołuje następujące wywołania:

```
ManagedConnection mc=mcf.matchManagedConnections(connectionSet, subject,
cxRequestInfo);
```

Implementacja fabryki ManagedConnection WebSphere MQ ignoruje parametr podmiotu. Ta metoda wybiera i zwraca odpowiedni obiekt ManagedConnection z zestawu lub zwraca wartość null, jeśli nie znajduje odpowiedniego obiektu ManagedConnection. Jeśli w puli nie ma odpowiedniego obiektu ManagedConnection , program ConnectionManager może utworzyć jeden za pomocą następujących elementów:

```
ManagedConnection mc=mcf.createManagedConnection(subject, cxRequestInfo);
```

Ponownie parametr podmiotu jest ignorowany. Ta metoda łączy się z menedżerem kolejek produktu WebSphere MQ i zwraca implementację interfejsu javax.resource.spi.ManagedConnection , która reprezentuje nowo wycenione połączenie. Gdy program ConnectionManager uzyska obiekt ManagedConnection (z puli lub ze świeżo utworzonego połączenia), tworzy on uchwyt połączenia przy użyciu:

```
Object handle=mc.getConnection(subject, cxRequestInfo);
```

Ten uchwyt połączenia może zostać zwrócony z funkcji allocateConnection().

ConnectionManager musi zarejestrować zainteresowanie w obiekcie ManagedConnection poprzez:

```
mc.addConnectionEventListener()
```

Obiekt nasłuchiwania ConnectionEventjest powiadamiany, jeśli w połączeniu wystąpi poważny błąd lub gdy wywołano komendę MQQueueManager.disconnect (). Po wywołaniu metody MQQueueManager.disconnect () obiekt nasłuchiwania ConnectionEventmoże wykonać jedną z następujących czynności:

- Zresetuj wartość ManagedConnection przy użyciu wywołania mc.cleanup(), a następnie zwróć wartość ManagedConnection do puli.
- Niszczenie obiektu ManagedConnection za pomocą wywołania mc.destroy()

Jeśli ConnectionManager jest domyślnym ConnectionManager, może również zarejestrować zainteresowanie stanem zestawu zarządzanego MQEnvironment MQPoolTokens. Aby to zrobić, najpierw należy utworzyć obiekt MQPoolServices , a następnie zarejestrować obiekt MQPoolServicesEventListener przy użyciu obiektu MQPoolServices :

```
MQPoolServices mqps=new MQPoolServices();
mqps.addMQPoolServicesEventListener(listener);
```

Program nasłuchujący jest powiadamiany, gdy element MQPoolToken zostanie dodany lub usunięty z zestawu lub gdy zmieni się wartość domyślna ConnectionManager . Obiekt MQPoolServices udostępnia również sposób na zapytanie o bieżącą wielkość zestawu MQPoolTokens.

Koordynacja JTA/JDBC przy użyciu klas produktu WebSphere MQ dla języka Java

Klasy WebSphere MQ classes for Java obsługują metodę MQQueueManager.begin (), która umożliwia WebSphere MQ działanie jako koordynator dla bazy danych, która udostępnia sterownik JDBC typu 2 lub JDBC typu 4 zgodny ze standardem.

Ta obsługa nie jest dostępna na wszystkich platformach. Aby sprawdzić, które platformy obsługują koordynację JDBC , należy zapoznać się z sekcji https://www.ibm.com/software/integration/wmq/requirements/.

Aby korzystać z obsługi XA-JTA, należy skorzystać ze specjalnej biblioteki przełącznika JTA. Metoda korzystania z tej biblioteki różni się w zależności od tego, czy używany jest system Windows , czy też jeden z innych platform.

Konfigurowanie koordynacji JTA/JDBC w produkcie Windows

Biblioteka XA jest dostarczana jako biblioteka DLL o nazwie formatu `jdbcxxx.dll`.

V 7.5.0.7 Dostarczona produkt `jdbcora12.dll` zapewnia kompatybilność z bazą danych Oracle 12Cw przypadku instalacji serwera IBM WebSphere MQ Windows .

W systemach Windows biblioteka XA jest dostarczana jako kompletna biblioteka DLL. Nazwa tej biblioteki DLL to `jdbcxxx.dll` , gdzie `xxx` wskazuje bazę danych, dla której skompilowano bibliotekę przełącznika. Ta biblioteka znajduje się w katalogu `java\lib\jdbc` lub `java\lib64\jdbc` klas IBM WebSphere MQ dla instalacji produktu Java . Należy zadeklarować bibliotekę XA, która jest również opisana jako plik ładowania przełącznika, do menedżera kolejek. Należy używać komponentu IBM WebSphere MQ Explorer. Określ szczegóły pliku ładowania przełącznika na panelu właściwości menedżera kolejek pod kontrolą menedżera zasobów XA. Należy podać tylko nazwę biblioteki. Na przykład:

W przypadku bazy danych Db2 ustaw pole SwitchFile na wartość: dbcdb2

W przypadku bazy danych Oracle ustaw pole SwitchFile na wartość: `jdbcora`

Konfigurowanie koordynacji JTA/JDBC na platformach innych niż Windows

Pliki obiektów są dostarczane. Powiąż odpowiedni plik z podanym plikiem makefile i zadeklaruj go w menedżerze kolejek przy użyciu pliku konfiguracyjnego.

W przypadku każdego systemu zarządzania bazami danych program WebSphere MQ udostępnia dwa pliki obiektów. Aby utworzyć 32-bitową bibliotekę przełącznika, należy utworzyć dowiązanie do jednego pliku obiektu, a następnie utworzyć dowiązanie do innego pliku wynikowego, aby utworzyć 64-bitową bibliotekę przełącznika. W przypadku bazy danych DB2nazwa każdego pliku obiektu to jdbcdb2.o , a w przypadku bazy danych Oraclenazwą każdego pliku obiektu jest jdbcora.o.

Każdy plik obiektu musi być dowiązany za pomocą odpowiedniego pliku makefile dostarczanego razem z produktem WebSphere MQ. Biblioteka przełączników wymaga innych bibliotek, które mogą być przechowywane w różnych położeniach w różnych systemach. Biblioteka przełącznika nie może jednak użyć zmiennej środowiskowej ścieżki biblioteki do zlokalizowania tych bibliotek, ponieważ biblioteka przełączników jest ładowana przez menedżer kolejek, który działa w środowisku setuid. Podany plik makefile zapewnia więc, że biblioteka przełączników zawiera pełne nazwy ścieżek dla tych bibliotek.

Aby utworzyć bibliotekę przełącznika, należy wprowadzić komendę **make** z następującym formatem. Aby utworzyć 32-bitową bibliotekę przełącznika, należy wprowadzić komendę w katalogu /java/lib/jdbc instalacji produktu WebSphere MQ . Aby utworzyć 64-bitową bibliotekę przełącznika, wprowadź komendę w katalogu /java/lib64/jdbc .

```
make DBMS
```

gdzie *DBMS* jest systemem zarządzania bazami danych, dla którego tworzona jest biblioteka przełącznika. Poprawne wartości to db2 dla DB2 i `oracle` dla Oracle.

Poniżej przedstawiono przykład komendy **make** :

```
make db2
```

Należy zwrócić uwagę na następujące kwestie:

- Aby uruchomić 32-bitowe aplikacje, należy utworzyć zarówno 32-bitową, jak i 64-bitową bibliotekę przełączników dla każdego używanego systemu zarządzania bazami danych. Aby uruchomić aplikacje 64-bitowe, należy utworzyć tylko 64-bitową bibliotekę przełączników. W przypadku bazy danych DB2nazwa każdej biblioteki przełącznika to jdbcdb2 , a w przypadku bazy danych Oraclenazwą każdej biblioteki przełącznika jest jdbcora. Pliki makefile zapewniają, że 32-bitowe i 64-bitowe biblioteki przełączników są przechowywane w różnych katalogach produktu WebSphere MQ . 32-bitowa biblioteka przełączników jest przechowywana w katalogu /java/lib/jdbc, a 64-bitowa biblioteka przełącznika jest przechowywana w katalogu /java/lib64/jdbc .
- Ponieważ produkt Oracle można zainstalować w dowolnym miejscu w systemie, pliki makefile używają zmiennej środowiskowej ORACLE_HOME w celu znalezienia miejsca, w którym zainstalowano bazę danych Oracle .

Po utworzeniu bibliotek przełącznika dla bazy danych DB2, Oraclelub obu tych bibliotek należy je zadeklarować w menedżerze kolejek. Jeśli plik konfiguracyjny menedżera kolejek (qm.ini) zawiera już sekcje XAResourceManager dla baz danych DB2 lub Oracle , należy zastąpić wpis SwitchFile w każdej sekcji za pomocą jednej z następujących czynności:

Dla bazy danych DB2

```
SwitchFile=jdbcdb2
```

Dla bazy danych Oracle

```
SwitchFile=jdbcora
```

Nie należy podawać pełnej nazwy ścieżki dla 32-bitowej lub 64-bitowej biblioteki przełącznika. Należy podać tylko nazwę biblioteki.

Jeśli plik konfiguracyjny menedżera kolejek nie zawiera już sekcji XAResourceManager dla baz danych DB2 lub Oracle lub jeśli wymagane jest dodanie dodatkowych sekcji XAResourceManager , należy zapoznać się z sekcją Administrowanie , aby uzyskać informacje na temat sposobu tworzenia sekcji XAResourceManager . Jednak każda pozycja SwitchFile w nowej sekcji XAResourceManager musi być dokładnie taka jak opisana wcześniej w przypadku bazy danych DB2 lub Oracle . Należy również dołączyć wpis `ThreadOfControl=PROCESS`.

Po zaktualizowaniu pliku konfiguracyjnego menedżera kolejek i upewniu się, że zostały ustawione wszystkie odpowiednie zmienne środowiskowe bazy danych, można zrestartować menedżer kolejek.

Korzystanie z koordynacji JTA/JDBC

Należy zakodować wywołania funkcji API, tak jak w podanym przykładzie.

Podstawowa sekwencja wywołań API dla aplikacji użytkownika jest następująca:

```
  qMgr = new MQQueueManager("QM1")
  Connection con = qMgr.getJDBCConnection( xads );
  qMgr.begin()

 < Perform MQ and DB operations to be grouped in a unit of work >

  qMgr.commit() or qMgr.backout();
  con.close()
  qMgr.disconnect()
```

`xads` w wywołaniu metody getJDBCConnection jest implementacją interfejsu XADataSource specyficzną dla bazy danych, która definiuje szczegóły bazy danych, z którą ma zostać nawiązane połączenie. Aby określić, w jaki sposób utworzyć odpowiedni obiekt XADataSource , który ma zostać przekazany do funkcji getJDBCConnection, należy zapoznać się z dokumentacją bazy danych.

Należy również zaktualizować ścieżkę klasy z odpowiednimi plikami jar specyficznymi dla bazy danych w celu wykonania pracy JDBC .

Jeśli konieczne jest nawiązanie połączenia z wieloma bazami danych, należy kilkakrotnie wywołać funkcję getJDBCConnection , aby wykonać transakcję na kilku różnych połączeniach.

Istnieją dwie formy metody getJDBCConnection, które są odzwierciedleniem dwóch form XADataSource.getXAConnection:

```
public java.sql.Connection getJDBCConnection(javax.sql.XADataSource xads)
  throws MQException, SQLException, Exception

public java.sql.Connection getJDBCConnection(XADataSource dataSource,
                                             String userid, String password)
  throws MQException, SQLException, Exception
```

Metody te deklarują wyjątek w klauzulach throws w celu uniknięcia problemów z weryfikatorem maszyny JVM dla klientów, którzy nie korzystają z funkcji JTA. Rzeczywisty zgłoszony wyjątek to javax.transaction.xa.XAException , który wymaga dodania pliku jta.jar do ścieżki klasy dla programów, które wcześniej nie wymagali tego pliku.

Aby korzystać z obsługi JTA/JDBC , w aplikacji należy umieścić następującą instrukcję:

```
MQEnvironment.properties.put(CMQC.THREAD_AFFINITY_PROPERTY, new Boolean(true));
```

Znane problemy i ograniczenia związane z koordynacją JTA/JDBC

Istnieją pewne problemy i ograniczenia dotyczące obsługi JTA/JDBC , niektóre w zależności od używanego systemu zarządzania bazami danych.

Ponieważ obsługa ta wywołuje sterowniki JDBC , implementacja tych sterowników JDBC może mieć istotny wpływ na działanie systemu. W szczególności testowane sterowniki JDBC zachowują się inaczej, gdy baza danych jest wyłączona, gdy aplikacja jest uruchomiona. **Zawsze** należy unikać nagłego zamknięcia bazy danych, gdy istnieją aplikacje, które mają do niego otwarte połączenia.

Wiele sekcji XAResourceManager

Użycie więcej niż jednej sekcji XAResourceManager w pliku konfiguracyjnym menedżera kolejek, qm.ini, nie jest obsługiwane. Każda sekcja XAResourceManager inna niż ta pierwsza jest ignorowana.

DB2

Czasami DB2 zwraca błąd SQL0805N . Ten problem można rozwiązać za pomocą następującej komendy CLP:

```
DB2 bind @db2cli.lst blocking all grant public
```

Więcej informacji na ten temat można znaleźć w dokumentacji DB2 .

Sekcja XAResourceManager musi być skonfigurowana tak, aby używana była opcja ThreadOfControl=PROCESS. W przypadku produktu DB2 w wersji 8.1 i nowszej nie jest to zgodne z domyślnym wątkiem ustawienia elementu sterującego dla bazy danych DB2, dlatego wartość toc=p musi być określona w łańcuchu Open String XA. Przykładowa sekcja XAResourceManager dla bazy danych DB2 z koordynacją JTA/JDBC jest następująca:

```
XAResourceManager:
     Name=jdbcdb2
     SwitchFile=jdbcdb2
     XAOpenString=uid=userid,db=dbalias,pwd=password,toc=p
     ThreadOfControl=PROCESS
```

Nie uniemożliwia to aplikacjom Java, które używają koordynacji JTA/JDBC , same wielowątkowe.

Oracle

Wywołanie metody JDBC Connection.close() po wywołaniu metody MQQueueManager.disconnect () powoduje wygenerowanie wyjątku SQLException. Należy wywołać metodę Connection.close()

przed wywołaniem metody MQQueueManager.disconnect () lub pominąć wywołanie metody Connection.close().

Obsługa protokołu SSL (Secure Sockets Layer) w klasach produktu WebSphere MQ dla języka Java

Klasy WebSphere MQ dla aplikacji klienckich Java obsługują szyfrowanie Secure Sockets Layer (SSL). Do korzystania z szyfrowania SSL wymagany jest dostawca JSSE.

Klasy WebSphere MQ dla aplikacji klienckich Java korzystających z protokołu TRANSPORT (CLIENT) obsługują szyfrowanie SSL (Secure Sockets Layer). Protokół SSL zapewnia szyfrowanie komunikacji, uwierzytelnianie i integralność komunikatów. Jest on zwykle używany do zabezpieczania komunikacji między dowolnymi dwoma równorzędnymi płatnikami w sieci Internet lub w intranecie.

Klasy WebSphere MQ classes for Java używają rozszerzenia JSSE (Java Secure Socket Extension) do obsługi szyfrowania SSL, dlatego wymaga dostawcy JSSE. Maszyny JVM JSE v1.4 mają wbudowaną dostawcę JSSE. Szczegółowe informacje na temat zarządzania certyfikatami i ich przechowywania mogą być różne od dostawcy. Aby uzyskać informacje na ten temat, zapoznaj się z dokumentacją dostawcy JSSE.

W tej sekcji założono, że dostawca JSSE jest poprawnie zainstalowany i skonfigurowany oraz że odpowiednie certyfikaty zostały zainstalowane i udostępnione dla dostawcy JSSE.

Jeśli klasy WebSphere MQ dla aplikacji klienckiej Java używają tabeli definicji kanału klienta (CCDT) do łączenia się z menedżerem kolejek, należy zapoznać się z “Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM WebSphere MQ classes for Java” na stronie 689.

Włączanie protokołu SSL w produkcie IBM WebSphere MQ classes for Java

Aby włączyć obsługę protokołu SSL, należy określić pakiet CipherSuite. Istnieją dwa sposoby określania pakietu CipherSuite.

Protokół SSL jest obsługiwany tylko dla połączeń klienckich. Aby włączyć protokół SSL, należy określić parametr CipherSuite , który ma być używany podczas komunikacji z menedżerem kolejek, a zestaw CipherSuite musi być zgodny z atrybutem CipherSpec ustawionym na kanale docelowym. Ponadto nazwa CipherSuite musi być obsługiwana przez dostawcę JSSE. Jednak pakiety CipherSuites różnią się od specyfikacji CipherSpecs i mają różne nazwy. Produkt “CipherSpecs i CipherSuites w klasach WebSphere MQ dla języka Java” na stronie 724 zawiera tabelę odwzorowania specyfikacji CipherSpecs obsługiwaną przez produkt IBM WebSphere MQ na równoważną wartość CipherSuites (w postaci znanej dla JSSE).

Aby włączyć protokół SSL, należy określić pakiet CipherSuite przy użyciu statycznej zmiennej składowej sslCipherSuite produktu MQEnvironment. Poniższy przykład przyłącza się do kanału SVRCONN o nazwie SECURE.SVRCONN.CHANNEL, który został skonfigurowany tak, aby wymagał połączenia SSL z atrybutem CipherSpec o wartości RC4_MD5_EXPORT:

```
MQEnvironment.hostname       = "your_hostname";
MQEnvironment.channel        = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.sslCipherSuite = "SSL_RSA_EXPORT_WITH_RC4_40_MD5";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Mimo że w kanale znajduje się specyfikacja CipherSpec o wartości RC4_MD5_EXPORT, aplikacja Java musi określić pakiet CipherSuite o wartości SSL_RSA_EXPORT_WITH_RC4_40_MD5. Listę odwzorowań między CipherSpecs i CipherSuitesmożna znaleźć w sekcji “CipherSpecs i CipherSuites w klasach WebSphere MQ dla języka Java” na stronie 724 .

Aplikacja może również określić pakiet CipherSuite , ustawiając właściwość środowiska CMQC.SSL_CIPHER_SUITE_PROPERTY.

Alternatywnie można użyć tabeli definicji kanału klienta (CCDT). Więcej informacji na ten temat zawiera sekcja “Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM WebSphere MQ classes for Java” na stronie 689.

Jeśli wymagane jest połączenie klienta z serwerem CipherSuite obsługiwanym przez dostawcę IBM Java JSSE FIPS (IBMJSSEFIPS), to aplikacja może ustawić pole Wymagane sslFipsw klasie

MQEnvironment na wartość `true`. Alternatywnie aplikacja może ustawić właściwość środowiska CMQC.SSL_FIPS_REQUIRED_PROPERTY. Wartością domyślną jest `false`, co oznacza, że połączenie klienckie może używać dowolnego zestawu CipherSuite obsługiwanego przez produkt IBM WebSphere MQ.

Jeśli aplikacja używa więcej niż jednego połączenia klienckiego, wartość pola sslFipswymaganego, która jest używana, gdy aplikacja tworzy pierwsze połączenie klienckie, określa wartość używaną podczas tworzenia kolejnego połączenia klienckiego przez aplikację. Z tego powodu, gdy aplikacja utworzy kolejne połączenie klienckie, wartość pola sslFipsRequired jest ignorowana. Jeśli w polu sslFipswymagane jest użycie innej wartości, należy zrestartować aplikację.

Aby pomyślnie nawiązać połączenie przy użyciu protokołu SSL, magazyn zaufanych certyfikatów JSSE musi być skonfigurowany z certyfikatami głównego ośrodka certyfikacji, z których certyfikat prezentowany przez menedżer kolejek może zostać uwierzytelniony. Podobnie, jeśli parametr SSLClientAuth w kanale SVRCONN został ustawiony na wartość MQSSL_CLIENT_AUTH_REQUIRED, magazyn kluczy JSSE musi zawierać certyfikat identyfikujący, który jest zaufany przez menedżer kolejek.

Odsyłacze pokrewne

Standardy FIPS (Federal Information Processing Standards) dla systemów UNIX, Linux i Windows

Korzystanie z nazwy wyróżniającej menedżera kolejek w produkcie IBM WebSphere MQ classes for Java

Menedżer kolejek identyfikuje się za pomocą certyfikatu SSL, który zawiera nazwę wyróżniającą (DN). Aplikacja kliencka IBM WebSphere MQ classes for Java może używać tej nazwy wyróżniającej, aby upewnić się, że komunikuje się z poprawnym menedżerem kolejek.

Wzorzec nazwy wyróżniającej jest określany przy użyciu zmiennej nazwy sslPeernazwy środowiska MQEnvironment. Na przykład:

```
MQEnvironment.sslPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

Umożliwia nawiązanie połączenia tylko wtedy, gdy menedżer kolejek wyświetli certyfikat o nazwie Common Name rozpoczynający się od QMGR., i co najmniej dwie nazwy jednostek organizacyjnych, z których pierwszym musi być IBM i drugi WebSphere.

Jeśli ustawiona jest nazwa sslPeerNazwa, połączenia powiodą się tylko wtedy, gdy zostanie ustawiona na poprawny wzorzec, a menedżer kolejek wyświetli pasujący certyfikat.

Aplikacja może również określić nazwę wyróżniającą menedżera kolejek, ustawiając właściwość środowiska CMQC.SSL_PEER_NAME_PROPERTY. Więcej informacji na temat nazw wyróżniających zawiera sekcja Nazwy wyróżniające.

Korzystanie z list odwołań certyfikatów w produkcie IBM WebSphere MQ classes for Java

Należy określić listy odwołań certyfikatów, które mają być używane przez klasę java.security.cert.CertStore . IBM WebSphere MQ classes for Java następnie sprawdza certyfikaty dla określonej listy CRL.

Lista odwołań certyfikatów (CRL) jest zestawem certyfikatów, które zostały odwołane przez wystawiający ośrodek certyfikacji lub przez organizację lokalną. Listy CRL są zwykle udostępniane na serwerach LDAP. W przypadku systemu Java 2 v1.4serwer listy CRL może zostać określony w czasie połączenia, a certyfikat przedstawiony przez menedżer kolejek jest sprawdzany przed listą CRL przed zezwolniem na nawiązanie połączenia. Więcej informacji na temat list odwołań certyfikatów i IBM WebSphere MQzawiera sekcja Praca z listami odwołań certyfikatów i listami odwołań uprawnień oraz Uzyskiwanie dostępu do list CRL i ARL z klasami produktu WebSphere MQ dla klas Java i klas WebSphere MQ dla usługi JMS.

Uwaga: Aby pomyślnie użyć narzędzia CertStore z listą CRL udostępnianą na serwerze LDAP, należy upewnić się, że pakiet Java Software Development Kit (SDK) jest kompatybilny z CRL. Niektóre pakiety SDK wymagają, aby lista CRL była zgodna z dokumentem RFC 2587, który definiuje schemat dla LDAP v2. Większość serwerów LDAP v3 używa zamiast niego RFC 2256.

Listy CRL, które mają być używane, są określone za pomocą klasy java.security.cert.CertStore . Szczegółowe informacje na temat uzyskiwania instancji programu CertStoremożna znaleźć w dokumentacji tej klasy. Aby utworzyć obiekt CertStore na podstawie serwera LDAP, należy najpierw utworzyć instancję LDAPCertStore, która zostanie zainicjowana z użyciem ustawień serwera i portu. Na przykład:

```
import java.security.cert.*;
CertStoreParameters csp = new LDAPCertStoreParameters("crl_server", 389);
```

Po utworzeniu instancji parametrów CertStorenależy użyć konstruktora statycznego w składnicy CertStore w celu utworzenia składnicy CertStore typu LDAP:

```
CertStore cs = CertStore.getInstance("LDAP", csp);
```

Obsługiwane są także inne typy CertStore (na przykład Kolekcja). Często istnieje kilka serwerów CRL ustawionych z identycznymi informacjami listy CRL, aby zapewnić nadmiarowość. Jeśli dla każdego z tych serwerów CRL istnieje obiekt CertStore , należy umieścić je wszystkie w odpowiedniej kolekcji. W poniższym przykładzie przedstawiono obiekty CertStore umieszczone na liście ArrayList:

```
import java.util.ArrayList;
Collection crls = new ArrayList();
crls.add(cs);
```

Ta kolekcja może zostać ustawiona w zmiennej statycznej MQEnvironment sslCertSklepy przed nawiązywaniem połączenia w celu włączenia sprawdzania CRL:

```
MQEnvironment.sslCertStores = crls;
```

Poprawność certyfikatu prezentowanego przez menedżera kolejek przy ustawionym połączeniu jest sprawdzana w następujący sposób:

1. Pierwszy obiekt CertStore w kolekcji identyfikowanej przez sslCertStores służy do identyfikacji serwera CRL.
2. Podjęto próbę skontaktowania się z serwerem CRL.
3. Jeśli próba zakończy się pomyślnie, serwer jest wyszukiwany pod kątem zgodności z certyfikatem.
   a. Jeśli certyfikat zostanie unieważniony, proces wyszukiwania zakończy się, a żądanie nawiązania połączenia nie powiedzie się i zostanie odebrany kod przyczyny MQRC_SSL_CERTIFICATE_ODWOŁANE.
   b. Jeśli certyfikat nie zostanie znaleziony, proces wyszukiwania zostanie nadany, a połączenie może być kontynuowane.
4. Jeśli próba skontaktowania się z serwerem nie powiedzie się, następny obiekt CertStore jest używany do identyfikacji serwera CRL, a proces jest powtarzany z kroku 2.

   Jeśli była to ostatnia CertStore w kolekcji lub jeśli kolekcja nie zawiera obiektów CertStore , proces wyszukiwania nie powiódł się, a żądanie nawiązania połączenia nie powiodło się. Kod przyczyny: MQRC_SSL_CERT_STORE_ERROR.

Obiekt Collection określa kolejność, w jakiej używane są CertStores .

Kolekcja CertStores może być również ustawiona za pomocą CMQC.SSL_CERT_STORE_PROPERTY. Dla wygody ta właściwość umożliwia również określenie pojedynczego obiektu CertStore , który nie jest elementem kolekcji.

Jeśli parametr sslCertStores jest ustawiony na wartość null, sprawdzanie listy CRL nie jest wykonywane. Ta właściwość jest ignorowana, jeśli zestaw sslCipherSuite nie jest ustawiony.

Renegocjowanie klucza tajnego w klasach produktu WebSphere MQ dla języka Java

Klasy WebSphere MQ dla aplikacji klienckiej Java mogą sterować, gdy ponownie negocjowany jest klucz tajny używany do szyfrowania połączenia klienckiego, pod względem łącznej liczby wysłanych i odebranych bajtów.

Aplikacja może to zrobić w jeden z następujących sposobów: jeśli aplikacja używa więcej niż jednego z tych sposobów, stosowane są zwykłe reguły dotyczące kolejności wykonywania zadań.

- W tym celu należy ustawić pole sslResetCount w klasie MQEnvironment.
- Ustawiając właściwość środowiska MQC.SSL_RESET_COUNT_PROPERTY w obiekcie Hashtable. Następnie aplikacja przypisuje tabelę mieszającą do pola `properties` w klasie MQEnvironment lub przekazuje tabelę mieszającą do obiektu MQQueueManager na jego konstruktorze.

Wartość pola licznika sslResetlub właściwości środowiska MQC.SSL_RESET_COUNT_PROPERTY reprezentuje łączną liczbę bajtów wysłanych i odebranych przez klasy produktu WebSphere MQ dla kodu klienta Java, zanim klucz tajny zostanie ponownie wynegocjowany. Liczba wysłanych bajtów jest liczbą przed zaszyfrowaniem, a liczba odebranych bajtów jest liczbą po deszyfrowaniu. Liczba bajtów obejmuje również informacje sterujące wysłane i odebrane przez klasy produktu WebSphere MQ dla klienta Java.

Jeśli wartość licznika resetowania wynosi zero, co jest wartością domyślną, klucz tajny nigdy nie będzie ponownie negocjowany. Licznik resetowania jest ignorowany, jeśli nie zostanie podany parametr CipherSuite .

Dostarczanie dostosowanej fabryki SSLSocketFactory w produkcie IBM WebSphere MQ classes for Java

Jeśli używana jest niestandardowa fabryka gniazd JSSE, należy ustawić właściwość MQEnvironment.sslSocketFactory na dostosowany obiekt fabryczny. Szczegóły różnią się między różnymi implementacjami JSSE.

Różne implementacje JSSE mogą udostępniać różne funkcje. Na przykład wyspecjalizowana implementacja JSSE może zezwalać na konfigurację konkretnego modelu sprzętu szyfrującego. Ponadto niektóre dostawcy JSSE umożliwiają dostosowywanie plików kluczy i magazynów zaufanych certyfikatów według programu lub umożliwiają zmianę opcji wyboru certyfikatu tożsamości z magazynu kluczy. W środowisku JSSE wszystkie te dostosowania są streszczane w klasie fabryki, javax.net.ssl.SSLSocketFactory.

Szczegółowe informacje na temat tworzenia dostosowanej implementacji SSLSocketFactory można znaleźć w dokumentacji JSSE. Szczegóły różnią się od dostawcy do dostawcy, ale typowa sekwencja kroków może być następująca:

1. Tworzenie obiektu SSLContext przy użyciu metody statycznej przy użyciu kontekstu SSLContext.
2. Zainicjuj ten kontekst SSLContext przy użyciu odpowiednich implementacji KeyManager i TrustManager (utworzonych na podstawie ich własnych klas fabrycznych)
3. Utwórz obiekt SSLSocketFactory z kontekstu SSLContext.

Jeśli istnieje obiekt SSLSocketFactory , ustaw właściwość MQEnvironment.sslSocketFactory na dostosowany obiekt fabryczny. Na przykład:

```
javax.net.ssl.SSLSocketFactory sf = sslContext.getSocketFactory();
MQEnvironment.sslSocketFactory = sf;
```

IBM WebSphere MQ classes for Java użyj tej komendy SSLSocketFactory , aby połączyć się z menedżerem kolejek produktu IBM WebSphere MQ . Tę właściwość można również ustawić przy użyciu właściwości CMQC.SSL_SOCKET_FACTORY_PROPERTY. Jeśli właściwość sslSocketFactory jest ustawiona na wartość NULL, używana jest domyślna fabryka SSLSocketFactory maszyny JVM. Ta właściwość jest ignorowana, jeśli zestaw sslCipherSuite nie jest ustawiony.

Jeśli używana jest niestandardowa właściwość SSLSocketFactories, należy wziąć pod uwagę wpływ współużytkowania połączeń TCP/IP. Jeśli możliwe jest współużytkowanie połączeń, nie jest wymagane podanie nowego gniazda dla podanej fabryki SSLSocketFactory , nawet jeśli w kontekście kolejnego żądania połączenia wygenerowane gniazdo będzie inne niż w jakiś sposób. Na przykład, jeśli w kolejnym

połączeniu ma zostać przedstawiony inny certyfikat klienta, współużytkowanie połączeń nie może być dozwolone.

Wprowadzanie zmian w magazynie kluczy lub magazynie zaufanych certyfikatów JSSE w klasach produktu WebSphere MQ dla języka Java

Jeśli zmienisz magazyn kluczy lub magazyn zaufanych certyfikatów JSSE, musisz wykonać określone działania, aby zmiany zostały uwzględnione.

Jeśli zawartość magazynu kluczy lub magazynu zaufanych certyfikatów JSSE zostanie zmieniona lub zostanie zmieniona lokalizacja pliku kluczy lub pliku zaufanych certyfikatów, klasy WebSphere MQ dla aplikacji Java, które są uruchomione w danym momencie, nie pobierają automatycznie zmian. Aby zmiany zostały uwzględnione, muszą zostać wykonane następujące działania:

- Aplikacje muszą zamknąć wszystkie połączenia i zniszczyć wszelkie nieużywane połączenia w pulach połączeń.
- Jeśli dostawca JSSE buforuje informacje z magazynu kluczy i magazynu zaufanych certyfikatów, te informacje muszą zostać odświeżone.

Po wykonaniu tych czynności aplikacje mogą następnie ponownie utworzyć połączenia.

W zależności od sposobu zaprojektowania aplikacji oraz funkcji udostępnianej przez dostawcę JSSE możliwe jest wykonanie tych działań bez zatrzymywania i restartowania aplikacji. Jednak zatrzymywanie i restartowanie aplikacji może być najprostszym rozwiązaniem.

Obsługa błędów podczas korzystania z protokołu SSL z klasami produktu WebSphere MQ dla języka Java

WebSphere MQ classes for Java-podczas nawiązywania połączenia z menedżerem kolejek przy użyciu protokołu SSL-może być wystawiona wiele kodów przyczyny.

Wyjaśnienia te przedstawiono na poniższej liście:

MQRC_SSL_NOT_ALLOWED
Ustawiono właściwość pakietu sslCipherSuite, ale użyto połączenia powiązań. Tylko połączenie klienta obsługuje protokół SSL.

MQRC_JSSE_ERROR
Dostawca JSSE zgłosił błąd, który nie może być obsługiwany przez produkt WebSphere MQ. Może to być spowodowane problemem konfiguracji z rozszerzeniem JSSE lub nie powiodło się sprawdzenie poprawności certyfikatu przedstawionego przez menedżer kolejek. Wyjątek utworzony przez JSSE można pobrać za pomocą metody getCause() w wyjątku MQException.

MQRC_SSL_INITIALIZATION_ERROR,
Wywołano komendę MQCONN lub MQCONNX z określonymi opcjami konfiguracji SSL, ale wystąpił błąd podczas inicjowania środowiska SSL.

MQRC_SSL_PEER_NAME_MISMATCH
Wzorzec nazwy wyróżniającej określony we właściwości sslPeerName nie jest zgodny z nazwą wyróżniającą podaną przez menedżer kolejek.

MQRC_SSL_PEER_NAME_ERROR-BŁĄD
Wzorzec nazwy wyróżniającej określony we właściwości sslPeerName nie był poprawny.

MQRC_UNSUPPORTED_CIPHER_SUITE
Pakiet CipherSuite nazwany w pakiecie sslCipherSuite nie został rozpoznany przez dostawcę JSSE. Pełna lista pakietów CipherSuites obsługiwana przez dostawcę JSSE może zostać uzyskana przez program przy użyciu metody SSLSocketFactory.getSupportedCipherSuites(). Listę CipherSuites , które mogą być używane do komunikowania się z produktem WebSphere MQ , można znaleźć w sekcji “CipherSpecs i CipherSuites w klasach WebSphere MQ dla języka Java” na stronie 724.

MQRC_SSL_CERTIFICATE_ODWOŁANE
Certyfikat prezentowany przez menedżer kolejek został znaleziony w CRL określonej za pomocą właściwości sslCertStores. Zaktualizuj menedżer kolejek, aby używać zaufanych certyfikatów.

MQRC_SSL_CERT_STORE_ERROR
Żaden z dostarczonych obiektów CertStores nie może być przeszukiwany pod kątem certyfikatu prezentowanego przez menedżera kolejek. Metoda MQException.getCause() zwraca błąd, który wystąpił podczas wyszukiwania pierwszej próby wykonania komendy CertStore . Jeśli wyjątek przyczynowy to NoSuchElementException, ClassCastException lub NullPointerException, należy sprawdzić, czy kolekcja określona we właściwości sslCertStores zawiera co najmniej jeden poprawny obiekt CertStore .

CipherSpecs i CipherSuites w klasach WebSphere MQ dla języka Java

To, czy aplikacja IBM WebSphere MQ classes for Java może nawiązać połączenie z menedżerem kolejek, zależy od specyfikacji CipherSpec określonej na końcu serwera kanału MQI, a CipherSuite określonej na końcu klienta.

W przypadku każdej kombinacji CipherSpec i CipherSuite, czy aplikacja IBM WebSphere MQ classes for Java może łączyć się z menedżerem kolejek, zależy od wartości pola sslFipsRequired w klasie MQEnvironment lub wartości właściwości CMQC.SSL_FIPS_REQUIRED_PROPERTY.

Na końcu serwera kanału MQI nazwa CipherSpec może zostać określona jako wartość parametru SSLCIPH w komendzie DEFINE CHANNEL CHLTYPE (SVRCONN). Na końcu klienta kanału MQI aplikacja IBM WebSphere MQ classes for Java może ustawić pole sslCipherSuite w klasie MQEnvironment lub ustawić właściwość środowiska CMQC.SSL_CIPHER_SUITE_PROPERTY.

Konfigurowanie aplikacji do korzystania z odwzorowań IBM Java lub Oracle Java CipherSuite

From IBM WebSphere MQ Version 7.5.0, pakiet poprawek 5, you can configure whether your application uses the default IBM Java CipherSuite to WebSphere MQ CipherSpec mappings, or the Oracle CipherSuite to WebSphere MQ CipherSpec mappings. W tym celu można użyć protokołu TLS CipherSuites , niezależnie od tego, czy aplikacja korzysta ze środowiska JRE firmy IBM , czy środowiska Oracle JRE. Właściwość systemowa Java `com.ibm.mq.cfg.useIBMCipherMappings` określa, które odwzorowania są używane. Właściwość może mieć jedną z następujących wartości:

Prawda
Use the IBM Java CipherSuite to WebSphere MQ CipherSpec mappings.

Ta wartość jest wartością domyślną.

Fałsz
Use the Oracle CipherSuite to WebSphere MQ CipherSpec mappings.

W poniższej tabeli znajduje się lista specyfikacji CipherSpecs obsługiwanych przez produkt IBM WebSphere MQ oraz ich odpowiedniki CipherSuites. Ta tabela wskazuje również, czy aplikacja IBM WebSphere MQ classes for Java może nawiązać połączenie z menedżerem kolejek, jeśli na końcu kanału MQI określono wartość CipherSpec , a na końcu klienta jest określony odpowiednik CipherSuite .

Tabela 89. CipherSpecs obsługiwane przez produkt WebSphere MQ i ich odpowiedniki CipherSuites

CipherSpec	Odpowiednik CipherSuite	Połączenie można nawiązać, jeśli wartość SFIPS [1] jest ustawiona na YES?
NULL_MD5	SSL_RSA_WITH_NULL_MD5	Nie
NULL_SHA	SSL_RSA_WITH_NULL_SHA	Nie

Tabela 89. CipherSpecs obsługiwane przez produkt WebSphere MQ i ich odpowiedniki CipherSuites (kontynuacja)

CipherSpec	Odpowiednik CipherSuite	Połączenie można nawiązać, jeśli wartość SFIPS [1] jest ustawiona na YES?
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5 (środowiskoIBM JRE) Nie ma odpowiednika dla środowiska Oracle JRE.	Nie
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	Nie
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA (środowiskoIBM JRE) Nie ma odpowiednika dla środowiska Oracle JRE.	Nie
RC2_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (środowiskoIBM JRE) SSL_RSA_EXPORT_WITH_RC4_40_MD5 (Oracle JRE)	Nie
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA (IBM JRE) Nie ma odpowiednika dla środowiska Oracle JRE.	Nie
RC4_56_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA (środowiskoIBM JRE) Nie ma odpowiednika dla środowiska Oracle JRE.	Nie
DES_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA (środowiskoIBM JRE) Nie ma odpowiednika dla środowiska Oracle JRE.	Nie
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA (środowiskoIBM JRE) Nie ma odpowiednika dla środowiska Oracle JRE.	Nie
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256 (środowiskoIBM JRE) TLS_RSA_WITH_NULL_SHA256 (Oracle JRE)	Nie [7]
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA (środowiskoIBM JRE) TLS_RSA_WITH_AES_128_CBC_SHA (Oracle JRE)	Tak [5] [7]

Tabela 89. CipherSpecs obsługiwane przez produkt WebSphere MQ i ich odpowiedniki CipherSuites (kontynuacja)

CipherSpec	Odpowiednik CipherSuite	Połączenie można nawiązać, jeśli wartość SFIPS [1] jest ustawiona na YES?
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256 (środowiskoIBM JRE) TLS_RSA_WITH_AES_128_CBC_SHA256 (Oracle JRE)	Tak [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA (środowiskoIBM JRE) TLS_RSA_WITH_AES_256_CBC_SHA (Oracle JRE)	Tak [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256 (środowiskoIBM JRE) TLS_RSA_WITH_AES_256_CBC_SHA256 (Oracle JRE)	Tak [5] [7]
AES_SHA_US[2]		
TLS_RSA_WITH_DES_CBC_SHA[8]	SSL_RSA_WITH_DES_CBC_SHA	Nie[3]
TLS_RSA_WITH_3DES_EDE_CBC_SHA[8] [9]	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Tak
FIPS_WITH_DES_CBC_SHA	SSL_RSA_FIPS_WITH_DES_CBC_SHA (IBM JRE) Nie ma odpowiednika dla środowiska Oracle JRE.	Nie[4]
FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (środowiskoIBM JRE) Nie ma odpowiednika dla środowiska Oracle JRE.	Nie[6]

Uwagi:

1. W aplikacji IBM WebSphere MQ classes for Java należy wskazać, że tylko algorytmy z certyfikatem FIPS mają być używane przez ustawienie pola sslFipsRequired w klasie MQEnvironment na wartość `true` i wskazanie, że algorytmy niecertyfikowane przez FIPS mogą być również używane przez ustawienie pola sslFipsRequired (Wymagane) na wartość `false`. Alternatywnie można ustawić właściwość środowiska CMQC.SSL_FIPS_REQUIRED_PROPERTY.
2. Ta właściwość CipherSpec nie ma odpowiednika CipherSuite.
3. Ta specyfikacja CipherSpec miała certyfikat FIPS 140-2 przed 19th maja 2007.
4. Ta specyfikacja CipherSpec miała certyfikat FIPS 140-2 przed 19th maja 2007. Nazwa FIPS_WITH_DES_CBC_SHA jest historyczna i odzwierciedla fakt, że ta specyfikacja CipherSpec była poprzednio (ale nie jest już) zgodna ze standardem FIPS. Ta specyfikacja szyfrowania jest nieaktualna i jej użycie nie jest zalecane.
5. Te CipherSpecs (TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) nie mogą być używane do zabezpieczenia połączenia z programu WebSphere MQ Explorer do menedżera kolejek,

chyba że odpowiednie nieograniczone pliki strategii są stosowane do środowiska JRE używanego przez eksplorator.

Więcej informacji na temat plików strategii zawiera sekcja Informacje o zabezpieczeniach .

6. Nazwa FIPS_WITH_3DES_EDE_CBC_SHA jest historyczna i odzwierciedla fakt, że ta specyfikacja CipherSpec była poprzednio (ale nie jest już) zgodna ze standardem FIPS. Ta specyfikacja szyfrowania jest nieaktualna i jej użycie nie jest zalecane.
7. Te CipherSpecs (TLS_RSA_WITH_NULL_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) wymagają środowisk IBM 6.0 SR13 FP2 , 7.0 SR4 FP2 lub nowszego.
8. Te CipherSpecs (TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_DES_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA256) mogą używać zarówno protokołu SSLv3 , jak i TLS. Domyślnie, gdy tryb FIPS nie jest włączony, używany jest protokół SSLv3 . Aby użyć protokołu TLS, ustaw właściwość systemową Java **com.ibm.mq.cfg.preferTLS** na wartość `true`.
9. Ta specyfikacja CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA jest nieaktualna. Jednak może być ona nadal używana do przesyłania do 32 GB danych, zanim połączenie zostanie zakończone z błędem AMQ9288. Aby uniknąć tego błędu, należy unikać używania potrójnego algorytmu szyfrowania DES lub włączyć resetowanie klucza tajnego podczas korzystania z tej specyfikacji CipherSpec.

Informacje pokrewne

Określanie, że w czasie wykonywania w kliencie MQI są używane tylko specyfikacje CipherSpecs z certyfikatem FIPS

Standardy FIPS (Federal Information Processing Standards) dla systemów UNIX, Linux i Windows

Blog MQdev: MQ Java, TLS Ciphers, Non-IBM JREs & APARs IT06775, IV66840, IT09423, IT10837

Blog MQdev: relacja między produktem MQ CipherSpecs a Java Cipher Suites

Uruchamianie klas produktu WebSphere MQ dla aplikacji Java

W przypadku napisania aplikacji (klasy zawierającej metodę main ()), przy użyciu trybu klienta lub powiązań, należy uruchomić program przy użyciu interpretera języka Java.

Użyj komendy:

```
java -Djava.library.path=library_path MyClass
```

gdzie *ścieżka_biblioteki* jest ścieżką do klas WebSphere MQ dla bibliotek Java (patrz Klasy WebSphere MQ dla bibliotek Java).

Klasy produktu WebSphere MQ dla zachowania zależnego od środowiska Java

Klasy produktu WebSphere MQ dla języka Java umożliwiają tworzenie aplikacji, które mogą być uruchamiane dla różnych wersji produktu WebSphere MQ. Ta kolekcja tematów opisuje zachowanie klas Java zależnych od tych różnych wersji.

Klasy WebSphere MQ classes for Java udostępniają podstawowe klasy klasy, które zapewniają spójną funkcję i zachowanie we wszystkich środowiskach. Funkcje poza tym rdzeniem zależą od możliwości menedżera kolejek, z którym połączona jest aplikacja.

Z wyjątkiem sytuacji, w których w tym miejscu wspomniano, zachowanie jest opisane w podręczniku Application Programming Reference odpowiednim dla menedżera kolejek.

Klasy podstawowe w klasach produktu WebSphere MQ dla języka Java

Klasy WebSphere MQ classes for Java zawierają podstawowy zestaw klas, które mogą być używane we wszystkich środowiskach.

Następujący zestaw klas jest uważany za klasy podstawowe i może być używany we wszystkich środowiskach, w których znajdują się tylko niewielkie różnice wymienione w sekcji “Ograniczenia i warianty podstawowych klas klas WebSphere MQ dla języka Java” na stronie 729.

- Środowisko MQEnvironment
- Wyjątek MQException
- Opcje MQGetMessage

  Wykluczanie:

  – MatchOptions
  – GroupStatus
  – SegmentStatus
  – Segmentacja
- MQManagedObject

  Wykluczanie:

  – inquire ()
  – set ()
- Komunikat MQMessage

  Wykluczanie:

  – groupId
  – messageFlags
  – messageSequenceLiczba
  – Przesunięcie
  – originalLength
- MQPoolServices
- Zdarzenie MQPoolServices
- MQPoolServicesEventListener
- MQPoolToken
- Opcje MQPutMessage

  Wykluczanie:

  – Liczba knownDest
  – Liczba unknownDest
  – Liczba invalidDest
  – recordFields
- Proces MQProcess
- MQQUEUE
- MQQueueManager

  Wykluczanie:

  – początek ()
  – Lista accessDistribution()
- Menedżer MQSimpleConnection
- Temat MQTopic
- MQC

Uwaga:

1. Niektóre stałe nie są zawarte w rdzeniu (szczegółowe informacje na ten temat znajdują się w sekcji “Ograniczenia i warianty podstawowych klas klas WebSphere MQ dla języka Java” na stronie 729). Nie należy ich używać w programach przenośnych.
2. Niektóre platformy nie obsługują wszystkich trybów połączenia. Na tych platformach można używać tylko klas i opcji podstawowych, które odnoszą się do obsługiwanych trybów. (Patrz “Opcje połączenia dla klas WebSphere MQ dla języka Java” na stronie 670).

Ograniczenia i warianty podstawowych klas klas WebSphere MQ dla języka Java

Klasy podstawowe zwykle zachowują się w sposób spójny we wszystkich środowiskach, nawet jeśli równoważne wywołania MQI zwykle mają różnice w środowisku. The behavior is as if a Windows, UNIX or Linux WebSphere MQ queue manager is used, except for the following minor restrictions and variations.

*Ograniczenia dotyczące wartości MQGMO_ * w klasach produktu WebSphere MQ dla języka Java*
Niektóre wartości MQGMO_ * nie są obsługiwane przez wszystkie menedżery kolejek.

Użycie następujących wartości MQGMO_ * może spowodować zgłoszenie wyjątku MQException z obiektu MQQueue.get():

MQGMO_SYNCPOINT_IF_PERSISTENT
MQGMO_MARK_SKIP_BACKOUT
MQGMO_BROWSE_MSG_UNDER_CURSOR
Blokada MQGMO_LOCK
MQGMO_UNLOCK
MQGMO_LOGICAL_ORDER
MQGMO_COMPLETE_MESSAGE
MQGMO_ALL_MSGS_AVAILABLE
MQGMO_ALL_SEGMENTS_AVAILABLE
MQGMO_UNMARKED_BROWSE_MSG
MQGMO_MARK_BROWSE_HANDLE
MQGMO_MARK_BROWSE_CO_OP
MQGMO_UNMARK_BROWSE_HANDLE
MQGMO_UNMARK_BROWSE_CO_OP

Dodatkowo, parametr MQGMO_SET_SIGNAL nie jest obsługiwany, gdy jest używany z języka Java.

*Ograniczenia dotyczące wartości MQPMRF_ * w klasach produktu WebSphere MQ dla języka Java*
Są one używane tylko podczas umieszczania komunikatów na liście dystrybucyjnej i są obsługiwane tylko przez menedżery kolejek obsługujące listy dystrybucyjne. Na przykład menedżery kolejek systemu z/OS nie obsługują list dystrybucyjnych.

*Ograniczenia dotyczące wartości MQPMO_ * w klasach produktu WebSphere MQ dla języka Java*
Niektóre wartości MQPMO_ * nie są obsługiwane przez wszystkie menedżery kolejek

Użycie następujących wartości MQPMO_ * może spowodować zgłoszenie wyjątku MQException z obiektu MQQueue.put() lub MQQueueManager.put ():

MQPMO_LOGICAL_ORDER
MQPMO_NEW_CORREL_ID
MQPMO_NEW_MESSAGE_ID
MQPMO_RESOLVE_LOCAL_Q

*Ograniczenia i zmiany wartości MQCNO_ * w klasach produktu WebSphere MQ dla języka Java*
Niektóre wartości `MQCNO_*` nie są obsługiwane.

- Automatyczne ponowne łączenie klienta nie jest obsługiwane przez klasy produktu WebSphere MQ dla języka Java. Niezależnie od ustawionej wartości `MQCNO_RECONNECT_*` połączenie zachowuje się tak, jak w przypadku ustawienia `MQCNO_RECONNECT_DISABLED`.

- Produkt MQCNO_FASTPATH jest ignorowany w menedżerach kolejek, które nie obsługują produktu MQCNO_FASTPATH. Jest ona również ignorowana przez połączenia klienckie.

*Ograniczenia dotyczące wartości MQRO_ * w klasach produktu WebSphere MQ dla języka Java*
Można ustawić następujące opcje raportu.

MQRO_EXCEPTION_WITH_FULL_DATA
MQRO_EXPIRATION_WITH_FULL_DATA
MQRO_COA_WITH_FULL_DATA
MQRO_COD_WITH_FULL_DATA
MQRO_DISCARD_MSG
MQRO_PASS_DISCARD_AND_WAŻNOŚCI

Więcej informacji na ten temat zawiera sekcja Raport.

Składniki znajdujące się poza klasami podstawowymi klas produktu WebSphere MQ dla języka Java

Klasy WebSphere MQ classes for Java zawierają pewne funkcje zaprojektowane specjalnie do korzystania z rozszerzeń interfejsu API, które nie są obsługiwane przez wszystkie menedżery kolejek. W tej kolekcji tematów opisano, w jaki sposób zachowują się one podczas korzystania z menedżera kolejek, który *nie* obsługuje tych menedżerów.

Zmiany w opcji konstruktora MQQueueManager
Niektóre konstruktory produktu MQQueueManager zawierają opcjonalny argument będący liczbą całkowitą. Niektóre wartości tego argumentu nie są akceptowane na wszystkich platformach.

W przypadku, gdy konstruktor MQQueueManager zawiera opcjonalny argument całkowitoliczbowy, jest on odwzorowyany na pole opcji MQCNO interfejsu MQI i jest używany do przełączania się między normalnym i szybkim połączeniem ścieżki. Ta rozszerzona forma konstruktora jest akceptowana we wszystkich środowiskach, jeśli jedynymi używanymi opcjami są: MQCNO_STANDARD_BINDING lub MQCNO_FASTPATH_BINDING. Inne opcje powodują, że konstruktor nie powiedzie się i zostanie zakończony błąd MQRC_OPTIONS_ERROR. Opcja krótkiej ścieżki CMQC.MQCNO_FASTPATH_BINDING jest honorowane tylko z powiązaniami powiązania z menedżerem kolejek, który go obsługuje. W innych środowiskach jest on ignorowany.

Ograniczenia dotyczące metody MQQueueManager.begin ()
Ta metoda może być używana tylko w przypadku menedżera kolejek produktu WebSphere MQ w systemach UNIX, Linuxlub Windows w trybie powiązań. W przeciwnym razie błąd komendy MQRC_ENVIRONMENT_ERROR nie powiedzie się.

Więcej szczegółów na ten temat zawiera sekcja “Koordynacja JTA/JDBC przy użyciu klas produktu WebSphere MQ dla języka Java” na stronie 716.

Zmiany w polach opcji MQGetMessage
Niektóre menedżery kolejek nie obsługują struktury MQGMO w wersji 2, dlatego konieczne jest ustawienie niektórych pól na ich wartości domyślne.

W przypadku korzystania z menedżera kolejek, który nie obsługuje struktury MQGMO w wersji 2, należy pozostawić następujące pola ustawione na wartości domyślne:

GroupStatus
SegmentStatus
Segmentacja

Ponadto w polu MatchOptions obsługiwane są tylko parametry MQMO_MATCH_MSG_ID i MQMO_MATCH_CORREL_ID. Jeśli w tych polach zostaną umieszczone nieobsługiwane wartości, to kolejna operacja MQDestination.get() nie powiedzie się i zostanie ona zakończona niepowodzeniem z błędem MQRC_GMO_ERROR. Jeśli menedżer kolejek nie obsługuje struktury MQGMO w wersji 2, to pola te nie są aktualizowane po pomyślnym zakończeniu operacji MQDestination.get().

Ograniczenia dotyczące list dystrybucyjnych w klasach produktu WebSphere MQ dla języka Java
Nie wszystkie menedżery kolejek umożliwiają otwarcie listy MQDistributionList.

Do tworzenia list dystrybucyjnych używane są następujące klasy:

MQDistributionList
Element MQDistributionList
MQMessageTracker

Istnieje możliwość utworzenia i zapełnienia elementów MQDistributionLists i MQDistributionListw dowolnym środowisku, ale nie wszystkie menedżery kolejek umożliwiają otwarcie listy MQDistributionList. W szczególności menedżery kolejek systemu z/OS nie obsługują list dystrybucyjnych. Próba otwarcia listy MQDistributionList podczas korzystania z takiego menedżera kolejek powoduje błąd MQRC_OD_ERROR.

Zmiany w polach opcji MQPutMessage
Jeśli menedżer kolejek nie obsługuje list dystrybucyjnych, niektóre pola MQPMO są traktowane inaczej.

Cztery pola w strukturze MQPMO są renderowane jako następujące zmienne składowe w klasie opcji MQPutMessage:

Liczba knownDest
Liczba unknownDest
Liczba invalidDest
recordFields

Te pola są przede wszystkim przeznaczone do użycia z listami dystrybucyjną. Jednak menedżer kolejek, który obsługuje listy dystrybucyjne, wypełnia pola DestCount po operacji MQPUT w jednej kolejce. Na przykład, jeśli kolejka jest tłumaczona na kolejkę lokalną, liczba knownDestjest ustawiona na 1, a pozostałe dwa pola licznika są ustawione na 0.

Jeśli menedżer kolejek nie obsługuje list dystrybucyjnych, wartości te są symulowane w następujący sposób:

- Jeśli operacja put () powiedzie się, wartość unknownDestCount jest ustawiona na 1, a pozostałe są ustawione na 0.
- Jeśli wykonanie komendy put () nie powiedzie się, wartość invalidDestLiczba jest ustawiona na 1, a pozostałe są ustawione na 0.

Zmienna recordFields jest używana z listami dystrybucyjną. Wartość może być zapisana w recordFields w dowolnym momencie, niezależnie od środowiska. Jest ona ignorowana, jeśli obiekt opcji MQPutMessagejest używany w kolejnych MQDestination.put() lub MQQueueManager.put (), a nie MQDistributionList.put ().

Ograniczenia w polach MQMD z klasami produktu WebSphere MQ dla języka Java
Niektóre pola MQMD, których dotyczy segmentacja komunikatów, powinny być pozostawiane w wartości domyślnej, gdy używany jest menedżer kolejek, który nie obsługuje segmentacji.

Następujące pola MQMD są w dużej mierze związane z segmentacją komunikatu:

GroupId
Numer_kolejny_komunikatu
Depozycja
MsgFlags
OriginalLength

Jeśli aplikacja ustawia dowolne z tych pól MQMD na wartości inne niż wartości domyślne, a następnie wykonuje komendę put () lub get () dla menedżera kolejek, który nie obsługuje tych pól, to put () lub get () zgłasza wyjątek MQException z MQRC_MD_ERROR. Pomyślne ustawienie put () lub get () z takim menedżerem kolejek zawsze pozostawia pola MQMD ustawione na wartości domyślne. Nie należy wysyłać zgrupowanych ani segmentowanych komunikatów do aplikacji Java, która jest uruchamiana względem menedżera kolejek, który nie obsługuje grupowania komunikatów i segmentacji.

Jeśli aplikacja Java próbuje uzyskać () komunikat z menedżera kolejek, który nie obsługuje tych pól, a komunikat fizyczny, który ma zostać pobrany, jest częścią grupy komunikatów segmentowanych (czyli ma wartości inne niż domyślne dla pól MQMD), jest on pobierany bez błędów. Jednak pola MQMD w komunikacie MQMessage nie są aktualizowane. Właściwość formatu MQMessage jest ustawiona na wartość MQFMT_MD_EXTENSION, a prawdziwe dane komunikatu są poprzedzane strukturą MQMDE, która zawiera wartości dla nowych pól.

Ograniczenia dotyczące klas produktu WebSphere MQ dla języka Java w produkcie CICS Transaction Server

W środowisku produktu CICS Transaction Server for z/OS tylko główny (pierwszy) wątek może wydawać wywołania CICS lub WebSphere MQ .

Należy zauważyć, że klasy WebSphere MQ JMS nie są obsługiwane w przypadku aplikacji CICS Java.

Dlatego nie jest możliwe współużytkowanie obiektów MQQueueManager i MQQueue między wątkami w tym środowisku lub utworzenie nowego menedżera MQQueueManager w wątku potomnym.

Uruchamianie klas IBM WebSphere MQ dla aplikacji Java na platformie Java na platformie Enterprise Edition

Przed użyciem klas produktu IBM WebSphere MQ dla produktu Java w środowisku Java EE należy wziąć pod uwagę pewne ograniczenia i uwagi dotyczące projektowania.

Klasy produktu IBM WebSphere MQ dla produktu Java mają ograniczenia w przypadku użycia w środowisku Java EE . Istnieją również dodatkowe uwagi, które należy wziąć pod uwagę podczas projektowania, implementowania i zarządzania klasami produktu IBM WebSphere MQ dla aplikacji Java , które działają w środowisku Java EE . Te ograniczenia i uwagi są opisane w poniższych sekcjach.

Ograniczenia transakcji JTA

Jedynym obsługiwanym menedżerem transakcji dla aplikacji korzystających z klas IBM WebSphere MQ dla produktu Java jest sam produkt IBM WebSphere MQ . Mimo że aplikacja w ramach elementu sterującego JTA może używać klas IBM WebSphere MQ dla produktu Java, każda praca wykonana przez te klasy nie jest kontrolowana przez jednostki pracy JTA. Zamiast tego tworzą one lokalne jednostki pracy oddzielone od tych zarządzanych przez serwer aplikacji za pośrednictwem interfejsów JTA. W szczególności wycofanie zmian w transakcji JTA nie powoduje wycofania żadnych wysłanych ani odebranych komunikatów. To ograniczenie dotyczy transakcji zarządzanych aplikacji lub komponentów bean oraz do kontenerów zarządzanych przez kontener i wszystkich kontenerów Java EE . Aby można było wykonywać operacje przesyłania komunikatów bezpośrednio z produktem IBM WebSphere MQ w ramach transakcji koordynowanych przez serwer aplikacji, należy zamiast nich użyć klas IBM WebSphere MQ dla usługi JMS.

Tworzenie wątków

Klasy IBM WebSphere MQ dla produktu Java są tworzone wewnętrznie przez wątki dla różnych operacji. Na przykład podczas uruchamiania w trybie BINDINGS w celu bezpośredniego wywołania w lokalnym menedżerze kolejek wywołania są wykonywane w wątku 'worker' utworzonym wewnętrznie przez klasy programu IBM WebSphere MQ dla produktu Java. Inne wątki mogą być tworzone wewnętrznie, na przykład w celu usunięcia nieużywanych połączeń z puli połączeń lub usunięcia subskrypcji dla zakończonych aplikacji publikowania/subskrypcji.

Niektóre aplikacje Java EE (na przykład te działające w kontenerach EJB i WWW) nie mogą tworzyć nowych wątków. Zamiast tego wszystkie prace muszą być wykonywane na głównych wątkach aplikacji zarządzanych przez serwer aplikacji. Gdy aplikacje korzystają z klas produktu IBM WebSphere MQ dla produktu Java, serwer aplikacji może nie być w stanie rozróżnić kodu aplikacji i klas IBM WebSphere MQ dla kodu Java , tak więc wątki opisane wcześniej powodują, że aplikacja nie jest zgodna ze specyfikacją kontenera. Klasy produktu IBM WebSphere MQ classes for JMS nie łamią tych specyfikacji Java EE , a więc mogą być używane.

Ograniczenia dotyczące bezpieczeństwa

Strategie bezpieczeństwa zaimplementowane przez serwer aplikacji mogą uniemożliwić wykonywanie niektórych operacji wykonywanych przez klasy IBM WebSphere MQ dla interfejsu API Java , takich jak tworzenie i wykonywanie nowych wątków sterowania (zgodnie z opisem w poprzednich sekcjach).

Na przykład serwery aplikacji zwykle działają z wyłączonymi zabezpieczeniami produktu Java i umożliwiają jej włączenie za pomocą konkretnej konfiguracji specyficznej dla serwera aplikacji (niektóre serwery aplikacji zezwalają również na bardziej szczegółową konfigurację strategii używanych w zabezpieczeniach produktu Java). Gdy zabezpieczenia produktu Java są włączone, klasy produktu IBM WebSphere MQ dla produktu Java mogą przerwać reguły wielowątkowości strategii zabezpieczeń produktu Java zdefiniowane dla serwera aplikacji, a interfejs API może nie być w stanie utworzyć wszystkich wątków, których potrzebuje w celu funkcjonowania. Aby zapobiec problemom z zarządzaniem wątkami, korzystanie z klas IBM WebSphere MQ dla produktu Java nie jest obsługiwane w środowiskach, w których włączono zabezpieczenia produktu Java .

Uwagi dotyczące odseparowania aplikacji

Zamierzonym korzyścią dla działających aplikacji w środowisku Java EE jest izolacja aplikacji. Projektowanie i implementacja klas produktu IBM WebSphere MQ dla produktu Java jest wcześniejsza niż środowisko Java EE . IBM WebSphere MQ Klasy dla produktu Java mogą być używane w sposób, który nie obsługuje pojęcia izolacji aplikacji. Do konkretnych przykładów rozważań w tym zakresie należą:

- Użycie ustawień statycznych (całego procesu maszyny JVM) w ramach klasy MQEnvironment, takich jak:
  - ID użytkownika i hasło, które mają być używane do identyfikacji i uwierzytelniania połączenia
  - nazwa hosta, port i kanał używany dla połączeń klienckich
  - Konfiguracja SSL dla bezpiecznych połączeń klientów

  Zmodyfikowanie wszystkich właściwości MQEnvironment na korzyść jednej aplikacji ma również wpływ na inne aplikacje korzystające z tych samych właściwości. W środowisku z wieloma aplikacjami, takim jak Java EE, każda aplikacja musi korzystać z własnej odrębnej konfiguracji poprzez utworzenie obiektów MQQueueManager z określonym zestawem właściwości, a nie z wartościami domyślnymi skonfigurowanymi w klasie MQEnvironment całej procesu.
- Klasa MQEnvironment wprowadza wiele metod statycznych, które działają globalnie dla wszystkich aplikacji korzystających z klas IBM WebSphere MQ dla produktu Java w ramach tego samego procesu maszyny JVM i nie ma możliwości przesłonięcia tego zachowania dla konkretnych aplikacji. Przykłady takich ograniczeń to:
  - konfigurowanie właściwości SSL, takich jak położenie pliku kluczy
  - konfigurowanie wyjść kanału klienta
  - włączanie lub wyłączanie śledzenia diagnostycznego
  - zarządzanie domyślną pulą połączeń używaną do optymalizacji korzystania z połączeń z menedżerami kolejek

  Wywoływanie takich metod ma wpływ na wszystkie aplikacje działające w tym samym środowisku Java EE .
- Zestawianie połączeń jest włączone w celu zoptymalizowania procesu nawiązywania wielu połączeń z tym samym menedżerem kolejek. Domyślny menedżer puli połączeń jest przetwarzalny w całym procesie i współużytkowany przez wiele aplikacji. Zmiany w konfiguracji puli połączeń, takie jak zastąpienie domyślnego menedżera połączeń dla jednej aplikacji za pomocą metody MQEnvironment.setDefaultConnectionManager(), wpływają w związku z tym na inne aplikacje działające na tym samym serwerze aplikacji Java EE .
- Protokół SSL jest konfigurowany dla aplikacji korzystających z klas IBM WebSphere MQ dla produktu Java przy użyciu klasy MQEnvironment i właściwości obiektu MQQueueManager . Nie jest ona zintegrowana z konfiguracją zabezpieczeń zarządzanych samego serwera aplikacji. Należy upewnić się, że klasy produktu IBM WebSphere MQ dla produktu Java są odpowiednio skonfigurowane, aby zapewnić wymagany poziom zabezpieczeń, a także nie korzystać z konfiguracji serwera aplikacji.

Ograniczenia trybu powiązań

Produkty IBM WebSphere MQ i WebSphere Application Server mogą być zainstalowane na tym samym komputerze, na którym są różne wersje główne menedżera kolejek i adaptera zasobów produktu IBM WebSphere MQ dostarczanego w produkcie WebSphere Application Server. Na przykład WebSphere Serwer aplikacji Version 7.0, który jest dostarczany na poziomie IBM WebSphere MQ RA 7.0.1, może być zainstalowany na tym samym komputerze, co menedżer kolejek produktu Version 6.0 .

Jeśli główne wersje menedżera kolejek i adaptera zasobów są różne, nie można używać połączeń powiązań. Wszystkie połączenia z serwera WebSphere Application Server z menedżerem kolejek przy użyciu adaptera zasobów muszą używać połączeń typu klienta. Połączenia powiązań mogą być używane, jeśli wersje są takie same.

Korzystanie z klas produktu WebSphere MQ dla usługi JMS

Klasy WebSphere MQ classes for Java Message Service (klasy WebSphere MQ classes for JMS) są dostawcą JMS dostarczonym z produktem WebSphere MQ. Oprócz implementowania interfejsów zdefiniowanych w pakiecie javax.jms , klasy WebSphere MQ classes for JMS udostępniają dwa zestawy rozszerzeń do interfejsu API JMS.

Specyfikacja JMS definiuje zestaw interfejsów, których aplikacje mogą używać do wykonywania operacji przesyłania komunikatów. Pakiet javax.jms definiuje interfejsy JMS, a dostawca JMS implementuje te interfejsy dla konkretnego produktu przesyłania komunikatów. Produkt WebSphere MQ w wersji 7.5 obecnie korzysta ze specyfikacji JMS 1.1 . Klasy WebSphere MQ classes for JMS są dostawcą JMS, który implementuje interfejsy JMS dla produktu WebSphere MQ.

Specyfikacja JMS oczekuje, że obiekty ConnectionFactory i Destination mają być administrowane obiektami. Administrator tworzy i obsługuje administrowane obiekty w centralnym repozytorium, a aplikacja JMS pobiera te obiekty przy użyciu interfejsu JNDI (Java Naming and Directory Interface). Klasy WebSphere MQ classes for JMS obsługują użycie administrowanych obiektów, a administrator może tworzyć i obsługiwać administrowane obiekty za pomocą narzędzia administracyjnego WebSphere MQ JMS lub WebSphere MQ Explorer.

Klasy WebSphere MQ classes for JMS udostępniają również dwa zestawy rozszerzeń do interfejsu API JMS. Główne uwagi dotyczące tych rozszerzeń dotyczą tworzenia i konfigurowania fabryk połączeń i miejsc docelowych dynamicznie w czasie wykonywania, ale rozszerzenia udostępniają również funkcje, które nie są bezpośrednio związane z przesyłaniem komunikatów, takie jak funkcja określania problemu.

Rozszerzenia JMS produktu WebSphere MQ
Poprzednie wersje klas WebSphere MQ classes for JMS zawierają rozszerzenia implementowane w obiektach, takich jak obiekty MQConnectionFactory, MQQueue i MQTopic. Te obiekty mają właściwości i metody specyficzne dla produktu WebSphere MQ. Obiektami mogą być obiekty administrowane, a aplikacja może tworzyć obiekty dynamicznie w czasie wykonywania. Ta wersja klas produktu WebSphere MQ dla usługi JMS utrzymuje te rozszerzenia, które są teraz nazywane rozszerzeniami JMS produktu WebSphere MQ . Można nadal używać, bez zmian, żadnych aplikacji, które używają tych rozszerzeń.

Rozszerzenia JMS produktu IBM
Ta wersja klas produktu WebSphere MQ dla usługi JMS udostępnia bardziej ogólny zestaw rozszerzeń interfejsu API JMS, które nie są specyficzne dla produktu WebSphere MQ jako systemu przesyłania komunikatów. Rozszerzenia te są nazywane rozszerzeniami JMS IBM i mają następujące ogólne cele:

- Zapewnienie większego poziomu spójności między dostawcami JMS IBM
- Aby ułatwić zapis aplikacji pomostowej między dwoma systemami przesyłania komunikatów IBM
- Aby ułatwić port aplikacji od jednego dostawcy IBM JMS do innego

Rozszerzenia udostępniają funkcje, które są podobne do funkcji udostępnionych w Message Service Client for C/C++ i Message Service Client for .NET.

Dlaczego warto korzystać z klas WebSphere MQ dla usługi JMS?

Korzystanie z klas produktu WebSphere MQ dla usługi JMS ma następujące zalety:

- Możliwe jest ponowne wykorzystanie umiejętności JMS.

  Klasy WebSphere MQ classes for JMS są dostawcą JMS, który implementuje interfejsy JMS dla produktu WebSphere MQ jako system przesyłania komunikatów. Jeśli organizacja jest nowa w produkcie WebSphere MQ, ale zawiera już umiejętności programistyczne aplikacji JMS, można łatwiej użyć znanego interfejsu API JMS w celu uzyskania dostępu do zasobów produktu WebSphere MQ , a nie do jednego z innych interfejsów API dostarczonych z produktem WebSphere MQ.

- Usługa JMS jest integralną częścią platformy Java Platform, Enterprise Edition (Java EE).

  Usługa JMS jest naturalnym interfejsem API, który ma być używany na potrzeby przesyłania komunikatów na platformie Java EE . Każdy serwer aplikacji, który jest zgodny z Java EE , musi zawierać dostawcę JMS. Usługi JMS można używać w klientach aplikacji, serwletach, stronach JavaServer (JSP), komponentach EJB i komponentach bean sterowanych komunikatami (message driven bean-MDBs). Należy zwrócić uwagę, w szczególności, że aplikacje Java EE używają komponentów MDB do asynchronicznego przetwarzania komunikatów, a wszystkie komunikaty są dostarczane do komponentów MDB jako komunikaty JMS.

- Administrator może tworzyć i obsługiwać administrowane obiekty JMS w centralnym repozytorium, a klasy WebSphere MQ dla aplikacji JMS mogą pobierać te obiekty przy użyciu interfejsu JNDI (Java Naming and Directory Interface).

  Fabryki połączeń JMS i miejsca docelowe obudowują informacje specyficzne dla produktu WebSphere MQ , takie jak nazwy menedżerów kolejek, nazwy kanałów, opcje połączeń, nazwy kolejek i nazwy tematów. Jeśli fabryki połączeń i miejsca docelowe są przechowywane jako obiekty administrowane, to informacje te nie są zakodowane na stałe w aplikacji. W związku z tym układ ten udostępnia aplikację o stopniu niezależności od bazowej konfiguracji produktu WebSphere MQ .

- Usługa JMS to standardowy interfejs API, który może zapewnić przenośność aplikacji.

  Aplikacja JMS może używać interfejsu JNDI do pobierania fabryk połączeń i miejsc docelowych, które są przechowywane jako obiekty administrowane, a także używać tylko interfejsów zdefiniowanych w pakiecie javax.jms w celu wykonywania operacji przesyłania komunikatów. Aplikacja jest wtedy całkowicie niezależna od dowolnego dostawcy JMS, takiego jak klasy produktu WebSphere MQ dla usługi JMS, a następnie może być importowana z jednego dostawcy JMS do innego bez żadnych zmian w aplikacji.

  Jeśli interfejs JNDI nie jest dostępny w określonym środowisku aplikacji, klasy produktu WebSphere MQ dla aplikacji JMS mogą używać rozszerzeń do interfejsu API JMS w celu dynamicznego tworzenia i konfigurowania fabryk połączeń i miejsc docelowych w czasie wykonywania. Aplikacja jest całkowicie samodzielna, ale jest powiązana z klasami produktu WebSphere MQ dla usługi JMS jako dostawcą JMS.

- Aplikacje pomostowe mogą być łatwiejsze do pisania przy użyciu usługi JMS.

  Aplikacja pomostowa to aplikacja, która odbiera komunikaty z jednego systemu przesyłania komunikatów i wysyła je do innego systemu przesyłania komunikatów. Pisanie aplikacji pomostowej może być skomplikowane przy użyciu specyficznych dla produktu interfejsów API i formatów komunikatów. Zamiast tego można napisać aplikację pomostową przy użyciu dwóch dostawców JMS, po jednym dla każdego systemu przesyłania komunikatów. Aplikacja używa wtedy tylko jednego interfejsu API, interfejsu API JMS i przetwarza tylko komunikaty JMS.

Pierwsze kroki z klasami produktu WebSphere MQ dla usługi JMS

W tym temacie przedstawiono przegląd klas produktu WebSphere MQ dla usługi JMS oraz informacje o tym, co należy wiedzieć przed użyciem klas produktu WebSphere MQ dla usługi JMS.

Wymagania wstępne dla klas produktu WebSphere MQ dla usługi JMS

Aby tworzyć i uruchamiać klasy produktu WebSphere MQ dla aplikacji JMS, należy spełnić wymagania wstępne niektórych komponentów oprogramowania.

Najnowsze informacje na temat wymagań wstępnych dla klas produktu WebSphere MQ dla usługi JMS zawiera plik readme produktu WebSphere MQ .

Aby utworzyć klasy produktu WebSphere MQ dla aplikacji JMS, potrzebny jest pakiet Java 2 Software Development Kit (SDK). Szczegółowe informacje na temat pakietów JDK obsługiwanych przez system operacyjny można znaleźć na stronie wymagań systemowych produktu WebSphere MQ . Patrz sekcja Wymagania produktu WebSphere MQ.

Aby uruchomić klasy produktu WebSphere MQ dla aplikacji JMS, należy użyć następujących komponentów oprogramowania:

- Menedżer kolejek produktu WebSphere MQ
- Środowisko Java Runtime Environment (JRE) dla każdego systemu, w którym uruchamiane są aplikacje

Jeśli do korzystania z modułów szyfrujących zgodnych z certyfikatem FIPS 140-2 wymagane są połączenia SSL, wymagany jest dostawca IBM Java JSSE FIPS (IBMJSSEFIPS). Każdy pakiet IBM Java 2 SDK i JRE w wersji 5 lub nowszej zawiera IBMJSSEFIPS.

Adresy Internet Protocol w wersji 6 (IPv6) w klasach WebSphere MQ dla aplikacji JMS pod warunkiem, że adresy IPv6 są obsługiwane przez wirtualną maszynę języka Java (JVM) oraz implementację protokołu TCP/IP w systemie operacyjnym użytkownika. Narzędzie administracyjne JMS produktu WebSphere MQ (patrz sekcja “Korzystanie z narzędzia administracyjnego JMS produktu WebSphere MQ” na stronie 960) akceptuje także adresy IPv6 .

Narzędzie administracyjne WebSphere MQ JMS oraz WebSphere MQ Explorer korzystają z interfejsu JNDI (Java Naming and Directory Interface) w celu uzyskania dostępu do usługi katalogowej, w której przechowywane są administrowane obiekty. Klasy produktu WebSphere MQ dla aplikacji JMS mogą również używać interfejsu JNDI do pobierania administrowanych obiektów z usługi katalogowej. Dostawca usług to kod, który zapewnia dostęp do usługi katalogowej przez odwzorowanie wywołań JNDI na wywołania usługi katalogowej. Następujące dostawcy usług są dostarczane z klasami produktu WebSphere MQ dla usługi JMS:

- Dostawca usług Lightweight Directory Access Protocol (LDAP) w plikach ldap.jar i providerutil.jar. Dostawca usług LDAP zapewnia dostęp do usługi katalogowej opartej na serwerze LDAP.
- Dostawca usług systemu plików w plikach fscontext.jar i providerutil.jar. Dostawca usług systemu plików zapewnia dostęp do usługi katalogowej w oparciu o lokalny system plików.

Jeśli planowane jest korzystanie z usługi katalogowej opartej na serwerze LDAP, należy zainstalować i skonfigurować serwer LDAP lub mieć dostęp do istniejącego serwera LDAP. W szczególności należy skonfigurować serwer LDAP do przechowywania obiektów Java. Informacje na temat instalowania i konfigurowania serwera LDAP można znaleźć w dokumentacji dostarczonej wraz z serwerem.

Przygotowywanie programów JMS dla klienta IBM WebSphere MQ dla produktu HP Integrity NonStop Server

W tym temacie opisano, co należy wiedzieć przed opracowaniu i uruchomieniem programów JMS dla klienta IBM WebSphere MQ dla produktu HP Integrity NonStop Server.

Klasy produktu IBM WebSphere MQ dla usługi JMS są instalowane jako część instalacji klienta produktu IBM WebSphere MQ dla produktu HP Integrity NonStop Server . Szczegółowe informacje na temat podsumowania zawartości instalacji zawiera sekcja System plików.

Niektóre aspekty funkcji klienta są specyficzne dla systemu operacyjnego hosta. Więcej informacji na temat obsługiwanych opcji dla klienta IBM WebSphere MQ dla produktu HP Integrity NonStop Serverzawiera sekcja Środowiska i opcje obsługiwane przez klienta IBM WebSphere MQ dla produktu HP Integrity NonStop Server.

Wymagania wstępne

Aby można było budować i uruchamiać aplikacje JMS, musi być zainstalowany i dostępny komponent *HP Integrity NonStop Server for Java* .

Konfiguracja

Więcej informacji na temat konfigurowania środowiska do uruchamiania i budowania aplikacji, w których można używać klas IBM WebSphere MQ dla usługi JMS, zawiera sekcja “Zmienne środowiskowe używane przez klasy produktu IBM WebSphere MQ dla usługi JMS” na stronie 741.

Więcej informacji na temat kroków wymaganych do skonfigurowania menedżera kolejek w celu akceptowania połączeń z aplikacji klienckich zawiera sekcja “Konfigurowanie po instalacji dla klas produktu WebSphere MQ dla aplikacji JMS” na stronie 792.

Więcej informacji na temat sprawdzania poprawności klas produktu IBM WebSphere MQ dla środowiska JMS zawiera sekcja “Test weryfikujący instalację punkt-punkt dla klas WebSphere MQ dla usługi JMS” na stronie 796.

Pisanie aplikacji

Więcej informacji na temat pisania aplikacji JMS zawiera sekcja “Zapisywanie klas produktu WebSphere MQ dla aplikacji JMS” na stronie 827.

Więcej informacji na temat korzystania z narzędzia administracyjnego JMS produktu IBM WebSphere MQ zawiera sekcja “Korzystanie z narzędzia administracyjnego JMS produktu WebSphere MQ” na stronie 960.

Przykłady

Przykładowe aplikacje znajdują się w następującym podkatalogu instalacji: `opt/mqm/samp/jms`.

Więcej informacji na temat kroków konfiguracji wymaganych do uruchomienia przykładów zawiera sekcja “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113.

Rozwiązywanie problemów

Aby uzyskać informacje na temat rozwiązywania problemów, patrz “Rozwiązywanie problemów z klasami produktu IBM WebSphere MQ dla usługi JMS” na stronie 817.

Instalowanie i konfigurowanie klas produktu WebSphere MQ dla usługi JMS

W tej sekcji opisano katalogi i pliki, które są tworzone podczas instalowania klas produktu WebSphere MQ dla usługi JMS, a także wyjaśniono sposób konfigurowania klas produktu WebSphere MQ dla usługi JMS po instalacji.

Pojęcia pokrewne

“Co jest zainstalowane dla klas IBM WebSphere MQ dla usługi JMS” na stronie 739
Podczas instalowania klas IBM WebSphere MQ dla usługi JMS tworzona jest pewna liczba plików i katalogów. W systemie Windows pewna konfiguracja jest wykonywana podczas instalacji, automatycznie ustawiając zmienne środowiskowe. W przypadku innych platform, w niektórych środowiskach Windows, należy ustawić zmienne środowiskowe, aby możliwe było uruchamianie klas produktu IBM WebSphere MQ dla aplikacji JMS.

“Uruchamianie klas produktu WebSphere MQ dla aplikacji JMS w ramach menedżera zabezpieczeń produktu Java” na stronie 748
Klasy produktu WebSphere MQ dla usługi JMS mogą być uruchamiane z włączonym menedżerem zabezpieczeń produktu Java . Aby pomyślnie uruchomić aplikacje z włączonym menedżerem zabezpieczeń, należy skonfigurować wirtualną maszynę języka Java (JVM) z odpowiednim plikiem konfiguracyjnym strategii.

“Adapter zasobów produktu IBM WebSphere MQ” na stronie 752
Adapter zasobów umożliwia aplikacjom działającym na serwerze aplikacji uzyskiwanie dostępu do zasobów produktu IBM WebSphere MQ . Obsługuje komunikację przychodzącą i wychodzącą.

“Konfigurowanie po instalacji dla klas produktu WebSphere MQ dla aplikacji JMS” na stronie 792
Ten temat zawiera informacje o tym, jakie uprawnienia potrzebne są do klas WebSphere MQ dla aplikacji JMS w celu uzyskania dostępu do zasobów menedżera kolejek. Przedstawiono również tryby połączenia

i opisano sposób konfigurowania menedżera kolejek w taki sposób, aby aplikacje mogły łączyć się w trybie klienta.

“Test weryfikujący instalację punkt-punkt dla klas WebSphere MQ dla usługi JMS” na stronie 796
Program IVT (point-to-point installation verification test) jest dostarczany z klasami WebSphere MQ dla usługi JMS. Program nawiązuje połączenie z menedżerem kolejek w trybie powiązań lub w trybie klienta i wysyła komunikat do kolejki o nazwie SYSTEM.DEFAULT.LOCAL.QUEUE, a następnie odbiera komunikat z kolejki. Program może tworzyć i konfigurować wszystkie obiekty, które są wymagane dynamicznie w czasie wykonywania, lub może używać interfejsu JNDI do pobierania obiektów administrowanych z usługi katalogowej.

“Test weryfikujący instalację publikowania/subskrypcji dla klas WebSphere MQ dla usługi JMS” na stronie 800
Program testu sprawdzającego instalację publikowania/subskrypcji (IVT) jest dostarczany z klasami produktu WebSphere MQ dla usługi JMS. Program nawiązuje połączenie z menedżerem kolejek w trybie powiązań lub w trybie klienta, subskrybuje temat, publikuje komunikat w temacie, a następnie odbiera komunikat, który właśnie opublikował. Program może tworzyć i konfigurować wszystkie obiekty, które są wymagane dynamicznie w czasie wykonywania, lub może używać interfejsu JNDI do pobierania obiektów administrowanych z usługi katalogowej.

“Program testowy weryfikujący instalację dla adaptera zasobów produktu WebSphere MQ” na stronie 803
Program IVT jest dostarczany jako plik EAR. Aby korzystać z programu, należy go wdrożyć i zdefiniować niektóre obiekty jako zasoby JCA.

“Konfigurowanie adaptera zasobów na potrzeby komunikacji wychodzącej” na stronie 771
Aby skonfigurować komunikację wychodzącą, należy zdefiniować właściwości obiektu ConnectionFactory i administrowanego obiektu docelowego.

“Obsługa środowiska OSGi” na stronie 816
Środowisko OSGi udostępnia środowisko, które obsługuje wdrażanie aplikacji w postaci pakunków. W ramach produktu IBM WebSphere MQ classes for JMS dostarczane są dziewięć pakunków OSGi.

“Rozwiązywanie problemów z klasami produktu IBM WebSphere MQ dla usługi JMS” na stronie 817
Problemy można śledzić, uruchamiając programy weryfikujące instalację i korzystając z narzędzi śledzenia i dziennika.

Zadania pokrewne

“Instalowanie i testowanie adaptera zasobów produktu MQ w produkcie WAS CE” na stronie 806
Instalowanie adaptera zasobów produktu IBM WebSphere MQ i uruchomienie testu sprawdzania poprawności instalacji (IVT) w produkcie WebSphere Application Server CE.

“Wdrażanie aplikacji IVT na serwerze WAS CE z niestandardowym środowiskiem MQ” na stronie 808
Jeśli zamiast trybu klienta ma być używana inna kolejka, menedżer kolejek, port, host, kanał lub tryb powiązań, należy zmodyfikować aplikację IVT i powiązane skrypty w produkcie WebSphere Application Server CE przed wdrożeniem adaptera zasobów lub aplikacji IVT.

“Wdrażanie aplikacji IVT w produkcie JBoss przy użyciu niestandardowego środowiska produktu IBM WebSphere MQ” na stronie 811
Jeśli adapter zasobów produktu IBM WebSphere MQ jest instalowany w produkcie JBoss, jeśli zamiast trybu klienta ma być używany inny tryb kolejki, menedżer kolejek, port, host, kanał lub powiązania, należy najpierw zmodyfikować aplikację IVT i powiązane skrypty w produkcie JBoss przed wdrożeniem adaptera zasobów lub aplikacji IVT.

Określanie problemu dla adaptera zasobów produktu IBM WebSphere MQ

Odsyłacze pokrewne

“Skrypty dostarczane z klasami produktu WebSphere MQ dla usługi JMS” na stronie 815
Udostępniono wiele skryptów ułatwiających wykonywanie typowych zadań, które należy wykonać podczas korzystania z klas WebSphere MQ dla usługi JMS.

Co jest zainstalowane dla klas IBM WebSphere MQ dla usługi JMS

Podczas instalowania klas IBM WebSphere MQ dla usługi JMS tworzona jest pewna liczba plików i katalogów. W systemie Windows pewna konfiguracja jest wykonywana podczas instalacji, automatycznie ustawiając zmienne środowiskowe. W przypadku innych platform, w niektórych środowiskach Windows, należy ustawić zmienne środowiskowe, aby możliwe było uruchamianie klas produktu IBM WebSphere MQ dla aplikacji JMS.

W przypadku większości systemów operacyjnych klasy IBM WebSphere MQ dla usługi JMS są instalowane jako opcjonalny komponent podczas instalowania produktu IBM WebSphere MQ. W przypadku klienta IBM WebSphere MQ dla produktu HP Integrity NonStop Serverdomyślnie instalowane są klasy IBM WebSphere MQ dla usługi JMS. Więcej informacji na temat instalowania produktu IBM WebSphere MQzawiera sekcja:

Instalowanie serwera WebSphere MQ
Instalowanie klienta IBM WebSphere MQ

Tabela 90 na stronie 739 pokazuje, gdzie klasy IBM WebSphere MQ dla plików JMS są instalowane na każdej platformie.

Tabela 90. Klasy produktu IBM WebSphere MQ dla katalogów instalacyjnych JMS

Platforma	**Katalog**
AIX	*MQ_INSTALLATION_PATH*/java
HP Integrity NonStop Server	*MQ_INSTALLATION_PATH*/opt/mqm/java
HP-UX, Linuxi Solaris	*MQ_INSTALLATION_PATH*/java
Windows	*MQ_INSTALLATION_PATH*\java
MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt IBM WebSphere MQ .	

Katalog instalacyjny zawiera następujące elementy:

- Klasy IBM WebSphere MQ dla plików JAR JMS, które znajdują się w katalogu *MQ_INSTALLATION_PATH*\java\lib.
- Biblioteki rodzime produktu IBM WebSphere MQ , które są używane przez aplikacje, które korzystają z interfejsu rodzimego języka Java.

  32-bitowe biblioteki rodzime są instalowane w katalogu *MQ_INSTALLATION_PATH*\java\lib, a 64-bitowe biblioteki rodzime można znaleźć w katalogu *MQ_INSTALLATION_PATH*\java\lib64 .

  Więcej informacji na temat rodzimych bibliotek produktu IBM WebSphere MQ zawiera sekcja “Konfigurowanie bibliotek Java Native Interface (JNI)” na stronie 743.
- Dodatkowe skrypty opisane w sekcji “Skrypty dostarczane z klasami produktu WebSphere MQ dla usługi JMS” na stronie 815. Skrypty te znajdują się w katalogu *MQ_INSTALLATION_PATH*\java\bin.
- Specyfikacje klas IBM WebSphere MQ dla interfejsu API JMS. Narzędzie Javadoc zostało użyte do wygenerowania stron HTML zawierających specyfikacje interfejsu API.

  Strony HTML znajdują się w katalogu *MQ_INSTALLATION_PATH*\java\doc\WMQJMSClasses.

  W systemach UNIX, Linuxi Windows podkatalog ten zawiera poszczególne strony HTML.
- Wsparcie dla OGSi. Pakunki OSGi są instalowane w katalogu java \lib\OSGi i są opisane w sekcji “Obsługa środowiska OSGi” na stronie 816.
- Adapter zasobów IBM WebSphere MQ , który może zostać wdrożony na dowolnym serwerze aplikacji zgodnym z architekturą JCA 1.5 (lub nowszym).

  Adapter zasobów produktu IBM WebSphere MQ znajduje się w katalogu *MQ_INSTALLATION_PATH*\java\lib\jca; więcej informacji na ten temat zawiera sekcja “Adapter zasobów produktu IBM WebSphere MQ” na stronie 752 .

- W systemie Windows symbole, które mogą być używane do debugowania, są instalowane w katalogu *MQ_INSTALLATION_PATH*\java\lib\symbole.

Katalog instalacyjny zawiera również niektóre pliki należące do innych komponentów produktu IBM WebSphere MQ . Są to następujące katalogi:

- Transport produktu IBM WebSphere MQ dla protokołu SOAP, który udostępnia transport JMS dla protokołu SOAP, jest instalowany w katalogu *MQ_INSTALLATION_PATH*\java\lib\soap. Więcej informacji na temat transportu IBM WebSphere MQ dla protokołu SOAP można znaleźć w sekcji centrum informacyjnego opisującej "Transport produktu WebSphere MQ dla protokołu SOAP" na stronie 973.
- Na platformach rozproszonych most IBM WebSphere MQ dla protokołu HTTP jest instalowany w katalogu *MQ_INSTALLATION_PATH*\java\lib\http. Więcej informacji na temat mostu IBM WebSphere MQ dla protokołu HTTP można znaleźć w sekcji centrum informacyjnego opisującej "Most produktu WebSphere MQ dla protokołu HTTP" na stronie 1052 .

Niektóre aplikacje przykładowe są dostarczane z klasami produktu IBM WebSphere MQ dla usługi JMS. Tabela 91 na stronie 740 pokazuje, gdzie przykładowe aplikacje są zainstalowane na każdej platformie.

Tabela 91. Katalogi przykładów

Platforma	**Katalog**
AIX	*MQ_INSTALLATION_PATH*/samp/jms
HP Integrity NonStop Server	*MQ_INSTALLATION_PATH*/opt/mqm/samp/jms
HP-UX, Linuxi Solaris	*MQ_INSTALLATION_PATH*/samp/jms
Windows	*MQ_INSTALLATION_PATH*\tools\jms
MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt IBM WebSphere MQ .	

Po zakończeniu instalacji może być konieczne wykonanie niektórych zadań konfiguracyjnych w celu skompilowania i uruchomienia aplikacji.

W produkcie "Zmienne środowiskowe używane przez klasy produktu IBM WebSphere MQ dla usługi JMS" na stronie 741 opisano ścieżkę klasy wymaganą do uruchamiania prostych klas produktu IBM WebSphere MQ dla aplikacji JMS. W tym temacie opisano także dodatkowe pliki JAR, do których należy się odwoływać w specjalnych okolicznościach, oraz zmienne środowiskowe, które należy ustawić w celu uruchomienia skryptów udostępnionych z klasami produktu IBM WebSphere MQ dla usługi JMS.

Jeśli wymagane jest, aby klasy produktu IBM WebSphere MQ dla aplikacji JMS były dowiązane do kodu napisanego w językach innych niż Java (na przykład w celu korzystania z transportu powiązań podczas nawiązywania połączenia z menedżerem kolejek), produkt "Konfigurowanie bibliotek Java Native Interface (JNI)" na stronie 743 wyjaśnia, gdzie znaleźć położenie bibliotek JNI (Java Native Interface), aby określić jako parametr komendy Java.

Aby sterować właściwościami, takimi jak śledzenie i rejestrowanie aplikacji, należy podać plik właściwości konfiguracyjnych. Plik właściwości konfiguracji klas IBM WebSphere MQ dla usługi JMS jest opisany w sekcji "Plik konfiguracyjny IBM WebSphere MQ classes for JMS" na stronie 745.

Instalowanie i aktualizowanie klas produktu WebSphere MQ dla plików JAR JMS

Jedynym obsługiwanym sposobem pobrania klas IBM WebSphere MQ dla plików JAR JMS w systemie jest zainstalowanie produktu IBM WebSphere MQ lub klienta WebSphere MQ V7.5 Clients SupportPac- MQC75lub za pomocą narzędzia do zarządzania oprogramowaniem, takiego jak Apache Maven, aby uzyskać więcej informacji na ten temat "Narzędzia IBM WebSphere MQ classes for JMS i narzędzia do zarządzania oprogramowaniem" na stronie 747.

Nie należy przenosić ani kopiować klas IBM WebSphere MQ dla plików JAR JMS lub rodzimych bibliotek, do innych komputerów lub do innego miejsca na komputerze, na którym zainstalowano klasy produktu IBM WebSphere MQ dla usługi JMS, chyba że używane jest narzędzie do zarządzania oprogramowaniem.

- Pakietów poprawek nie można zastosować do "instalacji", w której pliki JAR zostały skopiowane z innego komputera, ponieważ znacznie trudniej jest zapewnić, że wszystkie pliki JAR są przechowywane razem ze sobą, i są na kompatybilnych poziomach.
- Skopiowanie klas IBM WebSphere MQ dla plików JAR JMS między maszynami może również spowodować wiele kopii plików znajdujących się na tym samym komputerze, co może powodować problemy z obsługą kodu i debugowaniem problemów.
- Komenda dspmqver służąca do wyświetlania informacji o wersji z instalacji produktu IBM WebSphere MQ powoduje wyświetlenie tylko informacji o wersji klas IBM WebSphere MQ dla usługi JMS zainstalowanych w katalogu `\java\lib` .

  Jeśli wiele kopii plików znajduje się na tym samym komputerze, uruchomienie produktu **dspmqver** może nie dać dokładnych informacji na temat wersji klas produktu IBM WebSphere MQ dla usługi JMS używanej przez aplikację.

Nie należy uwzględniać klas IBM WebSphere MQ dla plików JAR JMS w archiwach aplikacji (takich jak archiwa aplikacji korporacyjnej lub pliki EAR).

- Aktualizacje klas IBM WebSphere MQ dla usługi JMS nie mogą być stosowane przy użyciu pakietu poprawek IBM WebSphere MQ .
- Nie jest możliwe, aby dział wsparcia IBM mógł łatwo określić wersję klas IBM WebSphere MQ dla usługi JMS, które są używane przez aplikację.
- Mogą wystąpić problemy, jeśli wiele aplikacji działających w tym samym środowisku Java Runtime Environment obejmuje różne wersje klas produktu IBM WebSphere MQ dla usługi JMS, ponieważ wiele wersji klas produktu IBM WebSphere MQ dla usługi JMS jest ładowanych do środowiska wykonawczego programów Java jednocześnie.

  Przykłady tych problemów obejmują następujące wyjątki:

```
java.lang.ClassCastException :
com.ibm.mq.jmqi.system.JmqiSystemEnvironment incompatible with
com.ibm.mq.jmqi.system.JmqiSystemEnvironment

java.lang.ClassCastException :
com.ibm.mq.jms.MQQueue incompatible with com.ibm.mq.jms.MQQueue
```

- Jeśli aplikacja korzysta z transportu BINDINGS w celu nawiązania połączenia z menedżerem kolejek, wszystkie główne aktualizacje menedżera kolejek wymagają zaktualizowania aplikacji tak, aby uwzględniała odpowiedni poziom klas IBM WebSphere MQ dla usługi JMS.

  Jeśli na przykład menedżer kolejek jest aktualizowany do wersji IBM WebSphere MQ 7.5 , wszystkie aplikacje, które łączą się z menedżerem kolejek przy użyciu transportu BINDINGS, również muszą zostać zaktualizowane w celu uwzględnienia klas produktu IBM WebSphere MQ w wersji 7.5 dla usługi JMS.

Zmienne środowiskowe używane przez klasy produktu IBM WebSphere MQ dla usługi JMS

Zanim możliwe będzie kompilowanie i uruchamianie klas produktu IBM WebSphere MQ dla aplikacji JMS, ustawienie zmiennej środowiskowej CLASSPATH musi zawierać klasy produktu IBM WebSphere MQ dla pliku archiwum Java (JAR) JMS. W zależności od wymagań może być konieczne dodanie innych plików JAR do ścieżki klasy. Aby uruchomić skrypty dostarczane z klasami produktu IBM WebSphere MQ dla usługi JMS, należy ustawić inne zmienne środowiskowe.

Aby skompilować i uruchomić klasy produktu IBM WebSphere MQ dla aplikacji JMS, należy użyć ustawienia CLASSPATH dla używanej platformy w sposób przedstawiony w sekcji Tabela 92 na stronie 742. To ustawienie obejmuje katalog przykładów, dzięki czemu można kompilować i uruchamiać klasy produktu IBM WebSphere MQ dla przykładowych aplikacji JMS. Alternatywnie można określić ścieżkę klasy w komendzie **java** zamiast używać zmiennej środowiskowej.

Tabela 92. Ustawienie CLASSPATH w celu kompilowania i uruchamiania klas produktu IBM WebSphere MQ dla aplikacji JMS, w tym przykładowych aplikacji	
Platforma	**Ustawienie CLASSPATH**
AIX	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms:
HP Integrity NonStop Server	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms:
HP-UX, Linuxi Solaris	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms:
IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar: /QIBM/ProdData/mqm/java/samples/jms:
Windows	CLASSPATH=*MQ_INSTALLATION_PATH*\java\lib\com.ibm.mqjms.jar; *MQ_INSTALLATION_PATH*\tools\jms;
z/OS	CLASSPATH=*MQ_INSTALLATION_PATH*/mqm/V7R0M0/java/lib/ com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/mqm/V6R0M0/java/samples/jms:
MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt IBM WebSphere MQ .	

Manifest pliku JAR com.ibm.mqjms.jar zawiera odwołania do większości innych plików JAR wymaganych przez klasy produktu IBM WebSphere MQ dla aplikacji JMS, a więc nie ma potrzeby dodawania tych plików JAR do ścieżki klasy. Te pliki JAR obejmują te pliki wymagane przez aplikacje, które używają interfejsu JNDI (Java Naming and Directory Interface) do pobierania administrowanych obiektów z usługi katalogowej oraz przez aplikacje, które korzystają z interfejsu API Java Transaction API (JTA).

Należy jednak uwzględnić dodatkowe pliki JAR w ścieżce klasy w następujących okolicznościach:

- Jeśli używane są klasy wyjścia kanału, które implementują interfejsy wyjścia kanału zdefiniowane w pakiecie com.ibm.mq , zamiast tych zdefiniowanych w pakiecie com.ibm.mq.exits , należy dodać klasy IBM WebSphere MQ dla pliku JAR Java (com.ibm.mq.jar) do ścieżki klasy.
- Jeśli kod Java jest kompilowany przy użyciu pakietu Java 2 Software Development Kit (SDK) w wersji 1.4.2, należy dodać następujące pliki JAR do ścieżki klasy:
  - jms.jar
  - com.ibm.mq.jmqi.jar

  Ponadto, jeśli aplikacja używa interfejsu JNDI do pobierania administrowanych obiektów z usługi katalogowej, do ścieżki klasy należy również dodać następujące pliki JAR:
  - fscontext.jar
  - jndi.jar
  - ldap.jar
  - providerutil.jar

  Jeśli aplikacja korzysta z narzędzia JTA, należy również dodać plik jta.jar do ścieżki klasy.

  Należy pamiętać, że te dodatkowe pliki JAR są wymagane tylko do kompilowania aplikacji, a nie dla ich uruchamiania.

W przypadku kompilacji za pomocą opcji -Xlint może zostać wyświetlony komunikat z ostrzeżeniem, że plik com.ibm.mq.ese.jar nie jest obecny. Ostrzeżenie można zignorować. Ten plik jest dostępny tylko wtedy, gdy zainstalowano produkt Extended Security Edition.

Skrypty udostępniane z klasami produktu IBM WebSphere MQ dla usługi JMS korzystają z następujących zmiennych środowiskowych:

MQ_JAVA_DATA_PATH
Ta zmienna środowiskowa określa katalog dla danych wyjściowych dziennika i śledzenia.

MQ_JAVA_INSTALL_PATH
Ta zmienna środowiskowa określa katalog, w którym zainstalowano klasy produktu WebSphere MQ dla usługi JMS.

MQ_JAVA_LIB_PATH
Ta zmienna środowiskowa określa katalog, w którym przechowywane są klasy produktu WebSphere MQ dla bibliotek JMS, jak to pokazano w sekcji Tabela 93 na stronie 744.

W systemie Windowswszystkie zmienne środowiskowe są ustawiane automatycznie podczas instalacji. Na każdej innej platformie, musisz ustawić je samodzielnie.

Aby ustawić zmienne środowiskowe w przypadku korzystania z 32-bitowej maszyny JVM w systemach UNIX, HP Integrity NonStop Server, lub Linux , można użyć skryptu setjmsenv. Aby ustawić zmienne środowiskowe w przypadku korzystania z 64-bitowej maszyny JVM w systemie UNIX lub Linux , można użyć skryptu setjmsenv64. Te skrypty znajdują się w katalogu`/java/bin` produktu *MQ_INSTALLATION_PATH*, gdzie *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym jest zainstalowany produkt IBM WebSphere MQ .

Można użyć skryptu setjmsenv lub setjmsenv64 na wiele sposobów: można go użyć jako podstawy do ustawienia wymaganych zmiennych środowiskowych, tak jak pokazano to w tabeli, lub dodać je do programu `.profile` przy użyciu edytora tekstu. Jeśli konfiguracja jest nietypowa, należy w razie potrzeby edytować treść skryptu. Alternatywnie można uruchomić skrypt w każdej sesji, z której mają być uruchamiane skrypty startowe JMS. W przypadku wybrania tej opcji należy uruchomić skrypt w każdym uruchomieniu okna powłoki, w trakcie procesu weryfikacji JMS, wpisując `. ./setjmsenv` lub `. ./setjmsenv64` .

Konfigurowanie bibliotek Java Native Interface (JNI)

Klasy produktu IBM WebSphere MQ dla aplikacji JMS, które łączą się z menedżerem kolejek przy użyciu transportu powiązań lub łączą się z menedżerem kolejek przy użyciu transportu klienta i używają programów obsługi wyjścia kanału napisanych w językach innych niż Java, muszą być uruchamiane w środowisku, które uzyskuje dostęp do bibliotek JNI (Java Native Interface).

O tym zadaniu

Aby skonfigurować to środowisko, należy skonfigurować ścieżkę do bibliotek środowiska, tak aby wirtualna maszyna języka Java (JVM) mogła załadować bibliotekę mqjbnd przed uruchomieniem klas produktu IBM WebSphere MQ dla aplikacji JMS.

Produkt IBM WebSphere MQ udostępnia dwa biblioteki JNI (Java Native Interface):

mqjbnd
Ta biblioteka jest używana przez aplikacje, które łączą się z menedżerem kolejek przy użyciu transportu powiązań. Udostępnia on interfejs między klasami produktu IBM WebSphere MQ dla usługi JMS i menedżerem kolejek. Biblioteka mqjbnd zainstalowana z produktem IBM WebSphere MQ w wersji 7.5 może być używana do łączenia się z dowolnym menedżerem kolejek produktu IBM WebSphere MQ w wersji 7.5 (lub wcześniejszej).

mqjexitstub02
Biblioteka mqjexitstub02 jest ładowana przez klasy IBM WebSphere MQ dla usługi JMS, gdy aplikacja łączy się z menedżerem kolejek przy użyciu transportu klienta i korzysta z programu obsługi wyjścia kanału napisanego w języku innym niż Java.

Na niektórych platformach produkt IBM WebSphere MQ instaluje 32-bitowe i 64-bitowe wersje tych bibliotek JNI. Położenie bibliotek dla każdej platformy jest przedstawione w Tabeli 1 .

Tabela 93. Położenie klas produktu IBM WebSphere MQ dla bibliotek JMS dla każdej platformy	
Platforma	**Katalog zawierający klasy produktu IBM WebSphere MQ dla bibliotek JMS**
AIX	*MQ_INSTALLATION_PATH*/java/lib (biblioteki 32-bitowe) *MQ_INSTALLATION_PATH*/java/lib64 (biblioteki 64-bitowe)
HP-UX	*MQ_INSTALLATION_PATH*/java/lib (biblioteki 32-bitowe) *MQ_INSTALLATION_PATH*/java/lib64 (biblioteki 64-bitowe)
Linux (POTĘGA , x86-64 i Platformy zSeries s390x)	*MQ_INSTALLATION_PATH*/java/lib (biblioteki 32-bitowe) *MQ_INSTALLATION_PATH*/java/lib64 (biblioteki 64-bitowe)
Linux (platformax86) Linux (Platforma zSeries)	*MQ_INSTALLATION_PATH* /java/lib
Solaris (platformyx86-64 i SPARC)	*MQ_INSTALLATION_PATH*/java/lib (biblioteki 32-bitowe) *MQ_INSTALLATION_PATH*/java/lib64 (biblioteki 64-bitowe)
Windows	*MQ_INSTALLATION_PATH*\java\lib (biblioteki 32-bitowe) *MQ_INSTALLATION_PATH*\java\lib64 (biblioteki 64-bitowe)
MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt IBM WebSphere MQ .	

Procedura

1. Skonfiguruj właściwość **java.library.path** maszyny JVM, która może być wykonana na dwa sposoby:
   - Określając argument maszyny JVM w sposób przedstawiony w poniższym przykładzie:

     ```
     -Djava.library.path=<path_to_library_directory>
     ```

     Linux Na przykład w przypadku 64-bitowej maszyny JVM w systemie Linux dla domyślnej instalacji położenia należy określić:

     ```
     -Djava.library.path=/opt/mqm/java/lib64
     ```

   - Skonfigurowanie środowiska powłoki w taki sposób, aby maszyna JVM skonfigurowała własny serwer `java.library.path`. Ta ścieżka różni się w zależności od platformy i położenia, w którym zainstalowano produkt IBM WebSphere MQ. Na przykład w przypadku 64-bitowej maszyny JVM i domyślnej lokalizacji instalacji produktu IBM WebSphere MQ można użyć następujących ustawień:

AIX `export LIBPATH=/usr/mqm/java/lib64:$LIBPATH`

Solaris HP-UX Linux
```
export LD_LIBRARY_PATH=/opt/mqm/java/
lib64:$LD_LIBRARY_PATH
```

Windows `set PATH=C:\Program Files\IBM\MQ\java\lib64;%PATH%`

Poniżej przedstawiono przykład stosu wyjątków, który jest używany w sytuacji, gdy środowisko nie zostało poprawnie skonfigurowane:

```
Spowodowane przez: com.ibm.mq.jmqi.local.LocalMQ$4: CC=2;RC=2495;
AMQ8598: Nie powiodła się próba załadowania rodzimej biblioteki JNI produktu WebSphere MQ :
mqjbnd.
    w pliku com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1268)
    w pliku com.ibm.mq.jmqi.local.LocalMQ$1.run(LocalMQ.java:309)
    na poziomie java.security.AccessController.doPrivileged(AccessController.java:400)
    w pliku com.ibm.mq.jmqi.local.LocalMQ.initialise_inner(LocalMQ.java:259)
    w pliku com.ibm.mq.jmqi.local.LocalMQ.initialise(LocalMQ.java:221)
    w pliku com.ibm.mq.jmqi.local.LocalMQ. < init> (LocalMQ.java:1350)
    w pliku com.ibm.mq.jmqi.local.LocalServer. < init> (LocalServer.java:230)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    w pliku
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:86)

    w pliku
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.jav
a:58)
    w klasie java.lang.reflect.Constructor.newInstance(Constructor.java:542)
    w metodzie com.ibm.mq.jmqi.JmqiEnvironment.getInstance(JmqiEnvironment.java:706)
    w elemencie com.ibm.mq.jmqi.JmqiEnvironment.getMQI(JmqiEnvironment.java:640)
    w pliku
com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createV7ProviderConnection(WMQConnectionF
actory.java:8437)
    ... 7 więcej
Spowodowany przez: java.lang.UnsatisfiedLinkError: mqjbnd (Nie znaleziono w pliku
java.library.path)
    w klasie java.lang.ClassLoader.loadLibraryWithPath(ClassLoader.java:1235)
    w klasie java.lang.ClassLoader.loadLibraryWithClassLoader(ClassLoader.java:1205)
    w java.lang.System.loadLibrary(System.java:534)
    w pliku com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1240)
    ... 20 więcej
```

2. Po ustawieniu środowiska 32-bitowego lub 64-bitowego uruchom klasy produktu IBM WebSphere MQ dla aplikacji JMS, używając komendy:

```
java application-name
```

gdzie *nazwa-aplikacji* jest nazwą klas IBM WebSphere MQ dla aplikacji JMS, które mają zostać uruchomione.

Wyjątek zawierający kod przyczyny IBM WebSphere MQ 2495 (MQRC_MODULE_NOT_FOUND) jest zgłaszany przez klasy produktu IBM WebSphere MQ dla usługi JMS, jeśli:

- Klasy IBM WebSphere MQ dla aplikacji JMS są uruchamiane w 32-bitowym środowisku wykonawczym Java, a środowisko 64-bitowe zostało skonfigurowane dla klas IBM WebSphere MQ dla usługi JMS, ponieważ 32-bitowe środowisko wykonawcze Java nie może załadować 64-bitowej biblioteki rodzimej Java.
- Klasy IBM WebSphere MQ dla aplikacji JMS są uruchamiane w 64-bitowym środowisku wykonawczym Java, a środowisko 32-bitowe zostało skonfigurowane dla klas IBM WebSphere MQ dla usługi JMS, ponieważ 64-bitowe środowisko wykonawcze Java nie może załadować 32-bitowej biblioteki rodzimej Java.

Plik konfiguracyjny IBM WebSphere MQ classes for JMS

Plik konfiguracyjny klasy WebSphere MQ dla pliku konfiguracyjnego JMS określa właściwości używane do konfigurowania klas WebSphere MQ dla usługi JMS.

Format klas WebSphere MQ dla pliku konfiguracyjnego JMS jest taki sam, jak standardowy plik właściwości Java. Przykładowy plik konfiguracyjny o nazwie jms.config jest dostarczany w podkatalogu

`bin` katalogu instalacyjnego produktu WebSphere MQ dla katalogu instalacyjnego JMS. Ten plik dokumentuje wszystkie obsługiwane właściwości i ich wartości domyślne.

Można wybrać nazwę i położenie klas produktu WebSphere MQ dla pliku konfiguracyjnego JMS. Podczas uruchamiania aplikacji należy użyć komendy **java** z następującym formatem:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

W komendzie *config_file_url* jest jednostajnym lokalizatorem zasobów (URL), który określa nazwę i położenie klas WebSphere MQ dla pliku konfiguracyjnego JMS. Obsługiwane są adresy URL następujących typów: http, file, ftp i jar.

Poniżej przedstawiono przykład komendy **java** :

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config MyAppClass
```

Ta komenda identyfikuje klasy produktu WebSphere MQ dla pliku konfiguracyjnego JMS jako plik D:\mydir\mjms.config w lokalnym systemie Windows .

Po uruchomieniu aplikacji klasy WebSphere MQ classes for JMS odczytuje treść pliku konfiguracyjnego i zapisuje określone właściwości w wewnętrznej składnicy właściwości. Jeśli komenda **java** nie identyfikuje pliku konfiguracyjnego lub jeśli nie można znaleźć pliku konfiguracyjnego, klasy WebSphere MQ classes for JMS korzystają z wartości domyślnych dla wszystkich właściwości. Jeśli jest to wymagane, można przesłonić dowolną właściwość w pliku konfiguracyjnym, określając ją jako właściwość systemową w komendzie **java** .

Klasy WebSphere MQ dla pliku konfiguracyjnego JMS mogą być używane z dowolnym obsługiwanym transportami między aplikacją a menedżerem kolejek lub brokerem.

Należy zauważyć, że nie można określić śledzenia uruchamiania przez ustawienie właściwości w klasach WebSphere MQ dla pliku konfiguracyjnego JMS. Funkcję śledzenia uruchamiania można określić tylko przez ustawienie właściwości systemowej w komendzie **java** , tak jak pokazano to w poniższym przykładzie:

```
java -Dcom.ibm.msg.client.commonservices.trace.startup=true
     -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config
      MyAppClass
```

Nadpisywanie właściwości określonych w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ

Plik konfiguracyjny klienta MQI produktu WebSphere MQ może również określać właściwości, które są używane do konfigurowania klas WebSphere MQ dla usługi JMS. Jednak właściwości określone w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ mają zastosowanie tylko wtedy, gdy aplikacja łączy się z menedżerem kolejek w trybie klienta.

Jeśli jest to wymagane, można przesłonić dowolny atrybut w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ , określając go jako właściwość w klasach WebSphere MQ dla pliku konfiguracyjnego JMS. Aby przesłonić atrybut w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ , należy użyć wpisu o następującym formacie w klasach WebSphere MQ dla pliku konfiguracyjnego JMS:

```
com.ibm.mq.cfg.stanza.propName=propValue
```

Zmienne w pozycji mają następujące znaczenia:

sekcja
Nazwa sekcji w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ , który zawiera atrybut.

propName
Nazwa atrybutu określona w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ .

propValue
Wartość właściwości, która nadpisuje wartość atrybutu określonego w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ .

Alternatywnie można przesłonić atrybut w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ , określając właściwość jako właściwość systemową w komendzie **java** . Aby określić właściwość jako właściwość systemową, należy użyć poprzedniego formatu.

Tylko następujące atrybuty w pliku konfiguracyjnym klienta MQI produktu WebSphere MQ są istotne dla klas produktu WebSphere MQ dla usługi JMS. Jeśli zostaną określone lub nadpisane inne atrybuty, nie będzie to miało żadnego wpływu.

sekcja	**Atrybut**
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	ExitsDefaultPath
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	ExitsDefaultPath64
ClientExitSekcja ścieżki pliku konfiguracyjnego klienta	Ścieżka klasy JavaExits
Sekcja MessageBuffer pliku konfiguracyjnego klienta	MaximumSize
Sekcja MessageBuffer pliku konfiguracyjnego klienta	PurgeTime
Sekcja MessageBuffer pliku konfiguracyjnego klienta	UpdatePercentage
Sekcja TCP pliku konfiguracyjnego klienta	ClntRcvBufSize
Sekcja TCP pliku konfiguracyjnego klienta	ClntSndBufSize
Sekcja TCP pliku konfiguracyjnego klienta	Limit_czasu_połączenia
Sekcja TCP pliku konfiguracyjnego klienta	KeepAlive

Narzędzia IBM WebSphere MQ classes for JMS i narzędzia do zarządzania oprogramowaniem

Narzędzia do zarządzania oprogramowaniem, takie jak Apache Maven, mogą być używane razem z produktem IBM WebSphere MQ classes for JMS.

Wiele dużych organizacji programistycznych używa tych narzędzi do centralnego zarządzania repozytoriami bibliotek innych firm.

IBM WebSphere MQ classes for JMS składa się z wielu plików JAR. Podczas tworzenia aplikacji w języku Java za pomocą tego interfejsu API, na komputerze, na którym aplikacja jest rozwijana, wymagana jest instalacja serwera IBM WebSphere MQ , klienta lub klienta SupportPac .

Aby użyć takiego narzędzia i dodać pliki JAR, które tworzą IBM WebSphere MQ classes for JMS w repozytorium zarządzanym centralnie, należy zaobserwować następujące punkty:

- Repozytorium lub kontener muszą być dostępne tylko dla programistów w organizacji. Dowolna dystrybucja poza organizacją jest niedozwolona.
- Repozytorium musi zawierać kompletny i spójny zestaw plików JAR z pojedynczego wydania produktu IBM WebSphere MQ lub pakietu poprawek.
- Użytkownik jest odpowiedzialny za aktualizację repozytorium za pomocą obsługi technicznej udostępnianej przez dział wsparcia IBM .

W przypadku produktu IBM WebSphere MQ Version 7.5do repozytorium muszą zostać zainstalowane następujące pliki JAR:

- com.ibm.mqjms.jar.
- com.ibm.mq.jar
- com.ibm.mq.jmqi.jar

- com.ibm.mq.pcf.jar
- com.ibm.mq.headers.jar
- Plik CL3Export.jar jest wymagany, jeśli używany jest produkt IBM WebSphere MQ classes for JMS.
- Plik CL3Nonexport.jar jest wymagany, jeśli używany jest produkt IBM WebSphere MQ classes for JMS.
- Plik jndi.jar jest wymagany, jeśli używany jest produkt IBM WebSphere MQ classes for JMS.
- Plik ldap.jar jest wymagany, jeśli używany jest produkt IBM WebSphere MQ classes for JMS.
- Plik rmm.jar jest wymagany, jeśli używany jest produkt IBM WebSphere MQ classes for JMS.
- Plik dhbcore.jar jest wymagany, jeśli używany jest produkt IBM WebSphere MQ classes for JMS.
- Plik jms.jar jest wymagany, jeśli używany jest produkt IBM WebSphere MQ classes for JMS.
- Plik fscontext.jar jest wymagany, jeśli używany jest produkt IBM WebSphere MQ classes for JMS i dostęp do obiektów administrowanych JMS, które są przechowywane w kontekście JNDI systemu plików.
- providerutil.jar , jeśli używany jest produkt IBM WebSphere MQ classes for JMS i dostęp do obiektów administrowanych JMS, które są przechowywane w kontekście JNDI systemu plików.

Uruchamianie klas produktu WebSphere MQ dla aplikacji JMS w ramach menedżera zabezpieczeń produktu Java

Klasy produktu WebSphere MQ dla usługi JMS mogą być uruchamiane z włączonym menedżerem zabezpieczeń produktu Java . Aby pomyślnie uruchomić aplikacje z włączonym menedżerem zabezpieczeń, należy skonfigurować wirtualną maszynę języka Java (JVM) z odpowiednim plikiem konfiguracyjnym strategii.

Najprostszym sposobem na to jest zmiana pliku konfiguracyjnego strategii dostarczonego ze środowiskiem JRE. W większości systemów plik ten jest przechowywany w ścieżce `lib/security/java.policy`, w porównaniu do katalogu JRE. Plik konfiguracyjny strategii można edytować, korzystając z preferowanego edytora lub programu `policytool` dostarczonego ze środowiskiem JRE.

Ważne: V 7.5.0.8 Tam, gdzie to możliwe, termin *lista zaakceptowanych* (allowlist) jest stosowany w miejsce terminu *biała lista* (whitelist). Jednym z wyjątków są następujące nazwy właściwości systemowych produktu Java .

W przypadku korzystania z mechanizmu menedżera zabezpieczeń produktu Java z aplikacją, należy nadać następujące uprawnienia:

- FilePermission dla każdego pliku listy allowlist, który jest używany, z uprawnieniem do odczytu w trybie WYMUSZENIA, uprawnienia do zapisu w trybie DISCOVER.
- Właściwość PropertyPermission (odczyt) w przypadku właściwości com.ibm.mq.jms.whitelist, com.ibm.mq.jms.whitelist.discoveri com.ibm.mq.jms.whitelist.mode .

Opcja allowlisting ClassName jest obsługiwana w przypadku APAR IT14385 i IBM WebSphere MQ Version 7.5.0, pakiet poprawek 8. Więcej informacji na ten temat zawiera sekcja “ClassName allowlisting w JMS ObjectMessage” na stronie 749.

Poniżej przedstawiono przykład dwóch wpisów w pliku konfiguracyjnym strategii, które pozwalają na pomyślne uruchomienie klas WebSphere MQ dla usługi JMS w ramach domyślnego menedżera zabezpieczeń:

```
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.jmqi.jar" {
  //Required
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  //Required if mqclient.ini/mqs.ini configuration files are used
  permission java.io.FilePermission "/var/mqm/mqclient.ini","read";
  permission java.io.FilePermission "/var/mqm/mqs.ini","read";
  //For the client transport type.
  permission java.net.SocketPermission "*","connect";
  //For the bindings transport type.
  permission java.lang.RuntimePermission "loadLibrary.*";
  //For applications that use CCDT tables (access to the CCDT
```

```
AMQCLCHL.TAB)
  permission java.io.FilePermission
"/var/mqm/qmgrs/QMGR/@ipcc/AMQCLCHL.TAB","read";
  //For applications that use User Exits
  permission java.io.FilePermission "/var/mqm/exits/*","read";
  permission java.lang.RuntimePermission "createClassLoader";
  //Required for the z/OS platform
  permission java.util.PropertyPermission
"com.ibm.vm.bitmode","read";
};
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mqjms.jar" {
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  permission java.util.PropertyPermission "console.encoding","read";
  permission java.lang.RuntimePermission "setContextClassLoader";
  //tracing permissions
  permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.*","read";
  permission java.util.PropertyPermission "MQJMS_TRACE_LEVEL","read";
  permission java.util.logging.LoggingPermission "control";
  //Wherever trace output is expected
  permission java.io.FilePermission "/tmp/*","read,write";
  //Required for the z/OS platform
  permission java.util.PropertyPermission
"com.ibm.vm.bitmode","read";
};
```

Uwagi:

- *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .
- Pierwsza instrukcja `grant` zawiera uprawnienia wymagane przez klasy produktu WebSphere MQ dla usługi JMS, a druga instrukcja `grant` zawiera uprawnienia wymagane przez klasy produktu WebSphere MQ dla aplikacji JMS.
- Aby umożliwić klasy WebSphere MQ classes for JMS uzyskiwanie dostępu do plików archiwum produktu Java (JAR) aplikacji, należy dodać następujące uprawnienie do pierwszej instrukcji `grant` :

  ```
  permission java.io.FilePermission "/path_to_your_app/-", "read";
  ```

- Aby użyć tych instrukcji `grant` w pliku konfiguracyjnym strategii, może być konieczne zmodyfikowanie nazw ścieżek w zależności od miejsca, w którym zainstalowano klasy produktu WebSphere MQ dla usługi JMS, a także w zależności od miejsca, w którym są przechowywane aplikacje.
- Przykładowe aplikacje dostarczone wraz z klasami WebSphere MQ dla usługi JMS i skryptami do ich uruchamiania nie pozwalają na włączenie menedżera zabezpieczeń.

V 7.5.0.8 ClassName allowlisting w JMS ObjectMessage

V 7.5.0.8 W klasach WebSphere MQ classes for JMS obsługa dopuszczania klas w implementacji interfejsu JMS ObjectMessage umożliwia ograniczenie niektórych rodzajów ryzyka związanego z bezpieczeństwem, które mogą być powiązane z mechanizmem przekształcania do postaci szeregowej obiektów Java i mechanizmem deserializacji.

Uwaga: Tam, gdzie to możliwe, termin *lista zaakceptowanych* (allowlist) jest stosowany w miejsce terminu *biała lista* (whitelist). Wyjątek stanowią nazwy właściwości systemowych Java, o których mowa w niniejszym temacie.

Mechanizm przekształcania do postaci szeregowej i przekształcania z postaci szeregowej produktu Java został zidentyfikowany jako potencjalne ryzyko związane z bezpieczeństwem, ponieważ deserializacja tworzy instancje dowolnych obiektów produktu Java , w których istnieje możliwość złośliwego wysyłania danych w celu spowodowania różnych problemów. W programie Java Message Service (JMS) ObjectMessages , które używają serializacji do hermetyzacji i przenoszenia dowolnych obiektów, można zauważyć, że jedna z nich aplikacja do serializacji jest używana do przekształcania do postaci szeregowej.

Dopuszczanie do postaci szeregowej umożliwia ograniczenie niektórych rodzajów ryzyka, które mogą być szeregowane. Jawnie określając, które klasy mogą być hermetyzowane, a wyodrębnione z ObjectMessages, allowlisting zapewnia pewną ochronę przed pewnymi zagrożeniami do serializacji.

Allowlisting w klasach produktu WebSphere MQ dla usługi JMS

W przypadku APAR IT14385 i IBM WebSphere MQ Version 7.5.0, pakiet poprawek 8klasy WebSphere MQ classes for JMS obsługują opcję allowlisting klas w implementacji interfejsu JMS ObjectMessage . Lista allowlist definiuje, które klasy produktu Java mogą być przekształcane do postaci szeregowej za pomocą ObjectMessage.setObject() i zdeserializowane z parametrem ObjectMessage.getObject().

Próby przekształcenia do postaci szeregowej lub przekształcenia z postaci szeregowej instancji klasy, która nie znajduje się na liście allowlist z elementem ObjectMessage , powodują zgłoszenie wyjątku javax.jms.MessageFormatException z java.io.InvalidClassException jako jego przyczyną.

Tworzenie listy allowlist

Ważne: Klasy WebSphere MQ classes for JMS nie mogą być dystrybuowane z listą allowlist. Wybór klas, które mają zostać przesłane przy użyciu ObjectMessages , jest wyborem projektu aplikacji, a program IBM WebSphere MQ nie może tego wcześniej wyemitywać.

Z tego powodu mechanizm allowlistingu pozwala na dwa tryby pracy:

Wykrywanie
W tym trybie mechanizm generuje listę pełnych nazw klas, raportując wszystkie klasy, które zostały zaobserwowane do postaci szeregowej lub zesterializowane w obszarze ObjectMessages.

Egzekwowanie
W tym trybie mechanizm wymusza zezwalanie na dopuszczanie, odrzucanie prób serializowania lub deserializowania klas, które nie znajdują się na liście allowlist.

Jeśli chcesz użyć tego mechanizmu, musisz najpierw uruchomić w trybie DISCOVERY, aby zebrać listę aktualnie serializowanych i zdeserializowanych klas, przejrzeć listę i użyć go jako podstawy dla listy allowlist. Korzystanie z listy może być nawet niezmienione, ale lista musi zostać najpierw przejrzana, zanim zdecydujesz się na to.

Sterowanie mechanizmem allowlistingu

Dostępne są trzy właściwości systemowe umożliwiające sterowanie mechanizmem allowlistingu:

com.ibm.mq.jms.whitelist
Tę właściwość można określić w jeden z następujących sposobów:

- Nazwa ścieżki do pliku, który zawiera listę allowlist, w formacie identyfikatora URI pliku (czyli począwszy od `file:`). W trybie DISCOVERY plik ten jest zapisywany w mechanizmie allowlistingowym. Plik nie może istnieć. Jeśli plik istnieje, mechanizm zgłasza wyjątek, a nie nadpisuje go. W trybie WYMUSZENIE ten plik jest odczytyany przez mechanizm allowlistingu.
- Rozdzielana przecinkami pełna nazwa klasy, która stanowi listę allowlist.

Jeśli ta właściwość nie jest ustawiona, mechanizm listy allowlist jest nieaktywny.

Jeśli używany jest menedżer zabezpieczeń produktu Java , należy upewnić się, że klasy produktu WebSphere MQ dla plików JAR JMS mają dostęp do odczytu i zapisu do tego pliku.

com.ibm.mq.jms.whitelist.discover

- Jeśli ta właściwość nie jest ustawiona lub ma wartość false, mechanizm listy allowlist działa w trybie WYMUSZENIE.
- Jeśli ta właściwość jest ustawiona na wartość true, a lista allowlist została określona jako identyfikator URI pliku, mechanizm listy allowlist działa w trybie DISCOVERY.
- Jeśli ta właściwość jest ustawiona na wartość true, a lista allowlist została określona jako lista nazw klas, to mechanizm listy allowlist zgłasza odpowiedni wyjątek.
- Jeśli ta właściwość jest ustawiona na wartość true, a lista allowlist nie została określona przy użyciu właściwości com.ibm.mq.jms.whitelist, mechanizm listy allowlist jest nieaktywny.
- Jeśli ta właściwość jest ustawiona na wartość true, a plik allowlist już istnieje, mechanizm listy allowlist zgłasza wyjątek java.io.InvalidClassException , a wpisy nie są dodawane do pliku.

com.ibm.mq.jms.whitelist.mode
Tę właściwość łańcucha można określić na jeden z trzech sposobów:

- Jeśli ta właściwość jest ustawiona na wartość SERIALIZE, to tryb WYMUSZENIA wykonuje sprawdzanie poprawności listy allowlist tylko w metodzie ObjectMessage.setObject().
- Jeśli ta właściwość jest ustawiona na wartość DESERIALIZE, to tryb WYMUSZENIA wykonuje sprawdzanie poprawności listy allowlist tylko w metodzie ObjectMessage.getObject().
- Jeśli ta właściwość jest nieustawiona lub ustawiona na inną wartość, tryb WYMUSZENIA wykonuje sprawdzanie poprawności listy dozwolonych dla metod ObjectMessage.getObject() i ObjectMessage.setObject().

Format pliku allowlist

Są to główne cechy formatu pliku allowlist:

- Plik allowlist jest domyślnym kodowaniem plików platformy z końcami linii właściwymi dla platformy.

  Uwaga: Plik może wymagać konwersji, jeśli jest przenoszony między różnymi systemami heterogenicznymi.
- Każda niepusta linia zawiera pełną nazwę klasy. Puste wiersze są ignorowane.
- Komentarze można dołączać-wszystko po znaku '#', na końcu wiersza, jest ignorowane.
- Istnieje bardzo podstawowy mechanizm dzikiej przyrody:
  - Element '*' może być elementem **ostatniego** nazwy klasy.
  - Znak '*' jest zgodny z **pojedynczym** elementem nazwy klasy, to znaczy klasy, ale nie ma części pakietu.

  Oznacza to, że `com.ibm.mq.*` pasuje do `com.ibm.mq.MQMessage` , ale nie `com.ibm.mq.jmqi.remote.api.RemoteFAP`.

  Znaki wieloznaczne nie działają dla klas w pakiecie domyślnym, który jest przeznaczony dla klas bez jawnej nazwy pakietu, dlatego nazwa klasy "*" jest odrzucana.
- Źle sformatowane pliki allowlist, na przykład pliki zawierające wpis, taki jak `com.ibm.mq.*.Message`, gdzie znak wieloznaczny nie jest ostatnim elementem, powodują zgłoszenie wyjątku java.lang.IllegalArgumentException .
- Pusty plik allowlist ma wpływ na całkowite wyłączenie użycia obiektu ObjectMessage.

Format listy allowlist w postaci listy rozdzielanej przecinkami

Ten sam mechanizm tworzenia znaków wieloznacznych jest dostępny dla listy allowlist, która jest listą rozdzielaną przecinkami.

- Znak '*' może być rozwijany przez system operacyjny, jeśli jest określony w wierszu komend lub w skrypcie powłoki lub w pliku wsadowym, dlatego może wymagać specjalnej obsługi.
- Znak komentarza '#' ma zastosowanie tylko wtedy, gdy określony jest plik. Jeśli lista allowlist jest określona jako rozdzielona przecinkami lista nazw klas, to zakładając, że system operacyjny lub powłoka nie przetworzy go, ponieważ jest to domyślny znak komentarza w wielu powłokach systemu UNIX lub Linux , jest traktowany jako normalny znak.

Kiedy dozwolone jest wyświetlanie listy allowlistingu?

Allowlisting jest inicjowany, gdy aplikacja po raz pierwszy uruchamia metodę ObjectMessage setMessage() lub getMessage().

Właściwości systemowe są wartościowane, plik allowlist jest otwierany i w trybie wymuszania, lista klas allowlisty jest ładowana po zainicjowaniu mechanizmu. W tym momencie wpis jest zapisywany w pliku dziennika JMS produktu IBM WebSphere MQ dla aplikacji.

Gdy mechanizm jest inicjowany, jego parametry mogą nie zostać zmienione. Ponieważ czas inicjowania nie jest łatwo przewidywany, ponieważ zależy on od zachowania aplikacji. Ustawienia właściwości systemu i zawartość pliku allowlist powinny być zatem traktowane jako ustalone od czasu uruchomienia aplikacji. Nie należy zmieniać właściwości ani zawartości pliku allowlist w czasie, gdy aplikacja jest uruchomiona, ponieważ wyniki nie są gwarantowane.

Punkty do rozważenia

Najlepszym podejściem do minimalizowania ryzyka związanego z mechanizmem przekształcania do postaci szeregowej produktu Java byłoby zbadanie alternatywnych metod przesyłania danych, takich jak format JSON zamiast ObjectMessage. Korzystanie z mechanizmów IBM WebSphere MQ Advanced Message Security (AMS) umożliwia dodawanie kolejnych zabezpieczeń poprzez zapewnienie, że komunikaty pochodzą z zaufanych źródeł.

W przypadku korzystania z mechanizmu menedżera zabezpieczeń produktu Java z aplikacją, należy nadać następujące uprawnienia:

- FilePermission dla każdego pliku listy allowlist, który jest używany, z uprawnieniem do odczytu w trybie WYMUSZENIA, uprawnienia do zapisu w trybie DISCOVER.
- Właściwość PropertyPermission (odczyt) w przypadku właściwości com.ibm.mq.jms.whitelist, com.ibm.mq.jms.whitelist.discoveri com.ibm.mq.jms.whitelist.mode .

Pojęcia pokrewne

“Uruchamianie klas produktu WebSphere MQ dla aplikacji JMS w ramach menedżera zabezpieczeń produktu Java” na stronie 748

Klasy produktu WebSphere MQ dla usługi JMS mogą być uruchamiane z włączonym menedżerem zabezpieczeń produktu Java . Aby pomyślnie uruchomić aplikacje z włączonym menedżerem zabezpieczeń, należy skonfigurować wirtualną maszynę języka Java (JVM) z odpowiednim plikiem konfiguracyjnym strategii.

Adapter zasobów produktu IBM WebSphere MQ

Adapter zasobów umożliwia aplikacjom działającym na serwerze aplikacji uzyskiwanie dostępu do zasobów produktu IBM WebSphere MQ . Obsługuje komunikację przychodzącą i wychodzącą.

Architektura JCA (Java Platform, Enterprise Edition - Java EE) Connector Architecture (JCA) udostępnia standardowy sposób łączenia aplikacji działających w środowisku Java EE z systemem informacyjnym przedsiębiorstwa (Enterprise Information System-EIS), takim jak IBM WebSphere MQ lub Db2. Adapter zasobów produktu IBM WebSphere MQ implementuje interfejsy JCA 1.5 i zawiera IBM WebSphere MQ classes for JMS. Umożliwia on aplikacjom JMS i komponentami bean sterowanymi komunikatami (MDB), działającym na serwerze aplikacji, uzyskiwanie dostępu do zasobów menedżera kolejek produktu IBM WebSphere MQ . Adapter zasobów obsługuje zarówno domenę punkt z punktem, jak i domenę publikowania/subskrybowania.

Adapter zasobów produktu IBM WebSphere MQ obsługuje dwa typy komunikacji między aplikacją a menedżerem kolejek:

Komunikacja wychodząca
Aplikacja uruchamia połączenie z menedżerem kolejek, a następnie wysyła komunikaty JMS do miejsc docelowych JMS i odbiera komunikaty JMS z miejsc docelowych JMS w sposób synchroniczny.

Komunikacja przychodząca
Komunikat JMS, który przylatuje do miejsca docelowego JMS, jest dostarczany do komponentu MDB, który przetwarza komunikat asynchronicznie.

Aby uzyskać więcej informacji o serwerze IBM WebSphere MQ classes for JMS, patrz “Korzystanie z klas produktu WebSphere MQ dla usługi JMS” na stronie 734.

Adapter zasobów zawiera również IBM WebSphere MQ classes for Java. Klasy są automatycznie dostępne dla aplikacji działających na serwerze aplikacji, na którym wdrożono adapter zasobów, i umożliwiają aplikacjom działającym na tym serwerze aplikacji korzystanie z interfejsu API produktu IBM WebSphere MQ classes for Java podczas uzyskiwania dostępu do zasobów menedżera kolejek produktu IBM

WebSphere MQ . Więcej informacji na temat IBM WebSphere MQ classes for Java zawiera sekcja “Korzystanie z klas produktu WebSphere MQ dla języka Java” na stronie 669.

Korzystanie z IBM WebSphere MQ classes for Java w środowisku Java EE jest obsługiwane z ograniczeniami. Więcej informacji na temat tych ograniczeń można znaleźć w sekcji “Uruchamianie klas IBM WebSphere MQ dla aplikacji Java na platformie Java na platformie Enterprise Edition” na stronie 732.

Inna wymagana dokumentacja do obsługi adaptera zasobów JCA

Informacje na temat konfigurowania adaptera zasobów JCA można znaleźć w dokumentacji serwera aplikacji.

Każdy serwer aplikacji udostępnia własny zestaw interfejsów administracyjnych. Niektóre serwery aplikacji udostępniają graficzne interfejsy użytkownika służące do definiowania zasobów JCA, ale inne wymagają, aby administrator zapisał plany wdrożenia XML. W związku z tym, poza zakresem niniejszej dokumentacji, informacje na temat konfigurowania adaptera zasobów WebSphere MQ dla każdego serwera aplikacji można znaleźć w dokumentacji. Ta dokumentacja skupia się tylko na tym, co należy skonfigurować. Informacje na temat konfigurowania adaptera zasobów JCA można znaleźć w dokumentacji dostarczonej wraz z serwerem aplikacji.

Aby zapoznać się z tą dokumentacją, należy zapoznać się z klasami JMS i WebSphere MQ dla usługi JMS. Wiele właściwości używanych do konfigurowania adaptera zasobów produktu WebSphere MQ jest odpowiednikiem właściwości klas produktu WebSphere MQ dla obiektów JMS i ma tę samą funkcję.

Instalowanie adaptera zasobów produktu WebSphere MQ

Adapter zasobów produktu WebSphere MQ jest dostarczany jako plik archiwum zasobów (Resource Archive-RAR). Zainstaluj plik RAR na serwerze aplikacji. Może być konieczne dodanie katalogów do ścieżki systemowej.

Adapter zasobów produktu WebSphere MQ jest dostarczany jako plik archiwum zasobów (Resource Archive-RAR) o nazwie wmq.jmsra.rar. Ten plik jest instalowany razem z klasami produktu WebSphere MQ dla usługi JMS w katalogu przedstawionym w sekcji Tabela 94 na stronie 753.

Tabela 94. Katalog, w którym znajduje się plik wmq.jmsra.rar dla każdej platformy

Platforma	**Katalog**
AIX, HP-UX, Linux i Solaris	*MQ_INSTALLATION_PATH* /java/lib/jca
IBM i	/QIBM/ProdData/mqm/java/lib/jca
Windows	*MQ_INSTALLATION_PATH* \java\lib\jca
MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .	

Plik RAR zawiera klasy produktu WebSphere MQ dla usługi JMS oraz implementację interfejsu JCA w produkcie WebSphere MQ .

Należy zainstalować plik RAR adaptera zasobów produktu WebSphere MQ na serwerze aplikacji, ale sposób, w jaki to jest możliwe, zależy od serwera aplikacji. Informacje na temat instalowania pliku RAR adaptera zasobów można znaleźć w dokumentacji serwera aplikacji.

W przypadku połączeń powiązań w systemach UNIX and Linux należy upewnić się, że katalog zawierający biblioteki JNI (Java Native Interface) znajduje się w ścieżce systemowej. Informacje na temat położenia tego katalogu, który zawiera również klasy produktu WebSphere MQ dla bibliotek JMS, zawiera sekcja Tabela 93 na stronie 744. W systemie Windowsten katalog jest automatycznie dodawany do ścieżki systemowej podczas instalacji klas produktu WebSphere MQ dla usługi JMS.

Transakcje są obsługiwane zarówno w trybie klienta, jak i w trybie powiązań.

Adapter zasobów produktu WebSphere MQ i wersja klas WebSphere MQ dla usługi JMS używanych przez adapter zasobów muszą być na tym samym poziomie wydania.

WebSphere Application Server i adapter zasobów produktu WebSphere MQ
Nie należy używać adaptera zasobów produktu WebSphere MQ z serwerem WebSphere Application Server w wersji 6. WebSphere Application Server, V7, includes a version of the WebSphere MQ V7 Resource Adapter.

Nie należy używać adaptera zasobów produktu WebSphere MQ w produkcie WebSphere Application Server, V6. Aby uzyskać dostęp do zasobów menedżera kolejek produktu WebSphere MQ z poziomu serwera WebSphere Application Server z poziomu aplikacji JMS, należy użyć dostawcy przesyłania komunikatów produktu WebSphere MQ . Dostawca przesyłania komunikatów produktu WebSphere MQ zawiera wersję klas produktu WebSphere MQ dla usługi JMS.

WebSphere Application Server, V7, includes a version of the WebSphere MQ V7 Resource Adapter.

Więcej informacji na ten temat zawiera nota techniczna Która wersja adaptera zasobów WebSphere MQ (RA) jest dostarczana z produktem WebSphere Application Server ?.

WebSphere Application Server Liberty i adapter zasobów IBM WebSphere MQ
Adapter zasobów produktu IBM WebSphere MQ Version 7.5 może zostać zainstalowany w produkcie WebSphere Application Server Liberty w wersji 8.5.5z pakietem poprawek 2 lub nowszym przy użyciu funkcji wmqJmsClient-1.1 . Ewentualnie można, z zastrzeżeniem pewnych ograniczeń, zainstalować adapter zasobów przy użyciu ogólnej obsługi produktu Java Platform, Enterprise Edition Connector Architecture (Java EE JCA).

Ogólne ograniczenia dotyczące instalowania adaptera zasobów w profilu Liberty

Poniższe ograniczenia mają zastosowanie do adaptera zasobów produktu Version 7.5 w przypadku korzystania z funkcji wmqJmsClient-1.1 , a także w przypadku korzystania z ogólnego wsparcia JCA:

- Produkt IBM WebSphere MQ classes for Java nie jest obsługiwany w profilu Liberty. Nie mogą one być używane z funkcją przesyłania komunikatów serwera IBM WebSphere MQ Liberty ani z ogólną obsługą JCA. Więcej informacji na ten temat zawiera sekcja Korzystanie z interfejsów Java produktu WebSphere MQ w środowiskach J2EE/JEE.
- Adapter zasobów IBM WebSphere MQ ma typ transportu BINDINGS_THEN_CLIENT. Ten typ transportu nie jest obsługiwany w ramach funkcji przesyłania komunikatów produktu IBM WebSphere MQ Liberty.
- Składnik IBM WebSphere MQ Advanced Message Security (IBM WebSphere MQ AMS) nie jest dołączony do funkcji przesyłania komunikatów produktu IBM WebSphere MQ Liberty.

Adapter zasobów produktu IBM WebSphere MQ Version 7.5 nie może być używany z opcją wmqJmsClient-2.0 .

Konfiguracja adaptera zasobów produktu WebSphere MQ

Aby skonfigurować adapter zasobów produktu WebSphere MQ , należy zdefiniować różne zasoby JCA i właściwości systemowe.

Zdefiniuj zasoby JCA w następujących kategoriach:

- Właściwości obiektu ResourceAdapter , które reprezentują właściwości globalne adaptera zasobów, takie jak poziom śledzenia diagnostycznego. Te właściwości są opisane w sekcji “Konfiguracja obiektu ResourceAdapter” na stronie 755.
- Właściwości obiektu ActivationSpec , które określają sposób aktywowania komponentu MDB na potrzeby komunikacji przychodzącej. Te właściwości są opisane w sekcji “Konfigurowanie adaptera zasobów na potrzeby komunikacji przychodzącej” na stronie 757.
- Właściwości obiektu ConnectionFactory , z którego korzysta serwer aplikacji w celu utworzenia obiektu JMS ConnectionFactory na potrzeby komunikacji wychodzącej. Te właściwości są opisane w sekcji “Konfigurowanie adaptera zasobów na potrzeby komunikacji wychodzącej” na stronie 771.
- Właściwości administrowanego obiektu docelowego, z którego korzysta serwer aplikacji w celu utworzenia obiektu kolejki JMS lub obiektu tematu JMS na potrzeby komunikacji wychodzącej. Te właściwości są również opisane w sekcji “Konfigurowanie adaptera zasobów na potrzeby komunikacji wychodzącej” na stronie 771.

Plik RAR adaptera zasobów produktu WebSphere MQ zawiera plik o nazwie META-INF/ra.xml, który zawiera deskryptor wdrażania dla adaptera zasobów. Ten deskryptor wdrażania jest definiowany przez schemat XML w produkcie https://java.sun.com/xml/ns/j2ee/connector_1_5.xsd i zawiera informacje na temat adaptera zasobów i udostępnianych przez niego usług. Serwer aplikacji może również wymagać planu wdrożenia dla adaptera zasobów. Ten plan wdrożenia jest specyficzny dla serwera aplikacji. Na przykład serwer WebSphere Application Server Community Edition wymaga planu wdrożenia o nazwie geronimo-ra.xml.

Jeśli używany jest protokół SSL (Secure Sockets Layer), określ położenia pliku kluczy i pliku zaufanych certyfikatów jako właściwości systemu JVM, jak w następującym przykładzie:

```
java ... -Djavax.net.ssl.keyStore=
key_store_location
         -Djavax.net.ssl.trustStore=trust_store_location
         -Djavax.net.ssl.keyStorePassword=key_store_password
```

Te właściwości nie mogą być właściwościami obiektu ActivationSpec ani obiektu ConnectionFactory i nie można określić więcej niż jednego magazynu kluczy dla serwera aplikacji. Właściwości mają zastosowanie do całej maszyny JVM, a zatem mogą mieć wpływ na serwer aplikacji, jeśli inne aplikacje działające na serwerze aplikacji korzystają z połączeń SSL. Serwer aplikacji może również resetować te właściwości do różnych wartości. Więcej informacji na temat używania protokołu SSL z klasami produktu WebSphere MQ dla usługi JMS zawiera sekcja “Korzystanie z protokołu SSL (Secure Sockets Layer) z klasami produktu WebSphere MQ dla usługi JMS” na stronie 932.

Program do testowania instalacji (IVT) jest dostarczany razem z adapterem zasobów produktu WebSphere MQ , ale przed uruchomieniem programu należy skonfigurować adapter zasobów. Informacje na temat tego, co należy skonfigurować w celu uruchomienia programu IVT, zawiera sekcja “Program testowy weryfikujący instalację dla adaptera zasobów produktu WebSphere MQ” na stronie 803.

Dzienniki adaptera zasobów, ostrzeżenia i komunikaty o błędach korzystają z tego samego mechanizmu, co klasy produktu IBM WebSphere MQ dla usługi JMS, aby uzyskać szczegółowe informacje na ten temat, patrz sekcja “Rejestrowanie i IBM WebSphere MQ classes for JMS” na stronie 817. W przypadku produktu WebSphere Application Serverkomunikaty te są automatycznie przekierowane do dziennika danych wyjściowych serwera aplikacji. W przypadku innych serwerów aplikacji, takich jak WAS CE i JBoss, ta opcja będzie domyślnie przejść do pliku o nazwie mqjms.log. Aby skonfigurować adapter zasobów w celu dodatkowego rejestrowania komunikatów ostrzegawczych w standardowym dzienniku wyjściowym serwerów aplikacji, należy ustawić następującą właściwość systemową maszyny JVM dla serwera aplikacji:

```
-Dcom.ibm.msg.client.commonservices.log.outputName=mqjms.log,stdout
```

Szczegółowe informacje na temat ustawiania właściwości systemowej maszyny JVM można znaleźć w dokumentacji serwera aplikacji.

Konfiguracja obiektu ResourceAdapter
Obiekt ResourceAdapter hermetyzuje globalne właściwości adaptera zasobów WebSphere MQ . Zdefiniuj te właściwości, korzystając z narzędzi adaptera zasobów.

Obiekt ResourceAdapter ma dwa zestawy właściwości:

- Właściwości powiązane z śledzeniem diagnostycznym
- Właściwości powiązane z pulą połączeń zarządzaną przez adapter zasobów

Sposób definiowania tych właściwości zależy od interfejsów administracyjnych udostępnianych przez serwer aplikacji.

Więcej informacji na temat definiowania właściwości powiązanych ze śledzeniem diagnostycznym znajduje się w temacie Śledzenie adaptera zasobów produktu IBM WebSphere MQ.

Adapter zasobów zarządza wewnętrzną pulą połączeń połączeń JMS, które są używane do dostarczania komunikatów do komponentów MDB. Tabela 95 na stronie 756 Wyświetla listę właściwości obiektu ResourceAdapter , które są powiązane z pulą połączeń.

Tabela 95. Właściwości obiektu ResourceAdapter , które są powiązane z pulą połączeń			
Nazwa właściwości	**Typ**	**Wartość domyślna**	**Opis**
maxConnections	Łańcuch	50	Maksymalna liczba połączeń z menedżerem kolejek produktu WebSphere MQ i maksymalna liczba wdrożonych komponentów MDB.
connectionConcurrency	Łańcuch	1	Maksymalna liczba komponentów MDB, które mają współużytkować połączenie JMS. Współużytkowanie połączeń nie jest możliwe, a ta właściwość zawsze ma wartość 1.
Liczba reconnectionRetry	Łańcuch	5	Maksymalna liczba prób nawiązania połączenia przez adapter zasobów z menedżerem kolejek produktu WebSphere MQ , jeśli nawiązanie połączenia nie powiedzie się.
Odstęp czasu reconnectionRetry	Łańcuch	300 000	Mierzony w milisekundach czas, przez jaki adapter zasobów czeka przed podjęciem próby ponownego nawiązania połączenia z menedżerem kolejek produktu WebSphere MQ .
Liczba startupRetry	Łańcuch	0	Domyślna liczba prób nawiązania połączenia z komponentem MDB podczas uruchamiania, jeśli menedżer kolejek nie jest uruchomiony, gdy serwer aplikacji jest uruchomiony.
Przedział czasu startupRetry	Łańcuch	30 000	Domyślny czas uśpienia między kolejnymi próbami nawiązania połączenia startowego (w milisekundach).

Gdy komponent MDB jest wdrożony na serwerze aplikacji, tworzone jest nowe połączenie JMS, a konwersacja uruchomiona z menedżerem kolejek, pod warunkiem, że maksymalna liczba połączeń określona przez właściwość maxConnection nie została przekroczona. W związku z tym maksymalna liczba komponentów MDB jest równa maksymalnej liczbie połączeń. Jeśli liczba wdrożonych komponentów MDB osiągnie tę wartość maksymalną, wszelkie próby wdrożenia innego komponentu MDB nie powiodą się. Jeśli komponent MDB jest zatrzymany, jego połączenie może być używane przez inny komponent MDB.

W ogólnym przypadku, jeśli wdrażana jest wiele komponentów MDB, należy zwiększyć wartość właściwości maxConnections .

Właściwości reconnectionRetryCount i reconnectionRetryokreślają zachowanie adaptera zasobów, gdy połączenia z menedżerem kolejek produktu WebSphere MQ nie powiodą się, ponieważ na przykład awaria sieci. Jeśli nawiązanie połączenia nie powiedzie się, adapter zasobów zawiesi dostarczanie komunikatów do wszystkich komponentów MDB dostarczonych przez to połączenie przez okres określony przez właściwość reconnectionRetry. Następnie adapter zasobów podejmuje próbę ponownego nawiązania połączenia z menedżerem kolejek. Jeśli próba nie powiedzie się, adapter zasobów podejmuje dalsze próby ponownego połączenia w odstępach czasu określonych przez właściwość reconnectionRetry, dopóki nie zostanie osiągnięty limit określony przez właściwość reconnectionRetryCount. Jeśli wszystkie próby nie powiodą się, dostarczenie zostanie zatrzymane na stałe do momentu ręcznego zrestartowania komponentów MDB.

W ogólnym przypadku obiekt ResourceAdapter nie wymaga administrowania. Aby na przykład włączyć śledzenie diagnostyczne w systemach UNIX and Linux , można ustawić następujące właściwości:

```
traceEnabled:       true
traceLevel:         10
```

Te właściwości nie mają wpływu na to, że adapter zasobów nie został uruchomiony, co ma miejsce na przykład wtedy, gdy aplikacje korzystające z zasobów WebSphere MQ działają tylko w kontenerze klienta. W tej sytuacji można ustawić właściwości dla śledzenia diagnostycznego jako właściwości systemu wirtualnej maszyny języka Java (JVM). Właściwości można ustawić za pomocą opcji -D w komendzie **java** , jak w następującym przykładzie:

```
java ... -DtraceEnabled=true -DtraceLevel=6
```

Nie ma potrzeby definiowania wszystkich właściwości obiektu ResourceAdapter . Wszystkie właściwości pozostawione nieokreślonym przyjmują wartości domyślne. W środowisku zarządzanym lepiej nie mieszać dwóch sposobów określania właściwości. Jeśli zostaną one wymieszane, właściwości systemowe maszyny JVM mają pierwszeństwo przed właściwościami obiektu ResourceAdapter .

Konfigurowanie adaptera zasobów na potrzeby komunikacji przychodzącej
Aby skonfigurować komunikację przychodzącą, należy zdefiniować właściwości jednego lub większej liczby obiektów ActivationSpec .

Właściwości obiektu ActivationSpec określają sposób, w jaki komponent bean sterowany komunikatami (MDB) odbiera komunikaty JMS z kolejki WebSphere MQ . Zachowanie transakcyjne komponentu MDB jest zdefiniowane w jego deskryptorze wdrażania.

Obiekt ActivationSpec ma dwa zestawy właściwości:

- Właściwości używane do tworzenia połączenia JMS z menedżerem kolejek produktu WebSphere MQ
- Właściwości używane do tworzenia konsumenta połączenia JMS, który dostarcza komunikaty asynchronicznie po ich odebraniu w określonej kolejce

Sposób definiowania właściwości obiektu ActivationSpec zależy od interfejsów administracyjnych udostępnianych przez serwer aplikacji.

Tabela 96 na stronie 757 zawiera listę właściwości obiektu ActivationSpec , które są używane do tworzenia połączenia JMS z menedżerem kolejek produktu WebSphere MQ .

Tabela 96. Właściwości obiektu ActivationSpec używane do tworzenia połączenia JMS

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
applicationName	Łańcuch	• Nazwa klasy wywołującej, jeśli jest dostępna, nie może być dłuższa niż 28 znaków. Jeśli nie jest dostępny, zostanie użyty łańcuch `WebSphere MQ Client for Java` .	Nazwa, pod którą aplikacja jest rejestrowana w menedżerze kolejek. Ta nazwa aplikacji jest wyświetlana przez komendę **DISPLAY CONN MQSC/PCF** (gdzie pole ma nazwę **APPLTAG**) lub na ekranie **Połączenia aplikacji** w Eksploratorze IBM WebSphere MQ (gdzie pole ma nazwę **App name**).
brokerCCDurSubQueue[1]	Łańcuch	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Nazwa kolejki	Nazwa kolejki, z której konsument połączenia odbiera komunikaty subskrypcji trwałej
brokerCCSubbrokerCCSub[1]	Łańcuch	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Nazwa kolejki	Nazwa kolejki, z której konsument połączenia odbiera komunikaty nietrwałej subskrypcji

Tabela 96. Właściwości obiektu ActivationSpec używane do tworzenia połączenia JMS (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
brokerControlbrokerControl[1]	Łańcuch	• **SYSTEM.BROKER.CONTROL.QUEUE** • Nazwa kolejki	Nazwa kolejki sterującej brokera
brokerQueuekolejki brokera[1]	Łańcuch	• **"" (pusty łańcuch)** • Nazwa menedżera kolejek	Nazwa menedżera kolejek, w którym działa broker
brokerSubQueue[1]	Łańcuch	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Nazwa kolejki	Nazwa kolejki, z której konsument komunikatów nietrwałych odbiera komunikaty
Wersja brokera[1]	Łańcuch	• **unspecified** (nieokreślony)-po przeprowadzeniu migracji brokera z wersji V6 do wersji V7należy ustawić tę właściwość, aby nagłówki RFH2 nie były już używane. Po migracji ta właściwość nie jest już istotna. • **V1** -aby użyć brokera publikowania/ subskrybowania produktu WebSphere MQ . lub użyć brokera produktu WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker lub WebSphere Business Integration Message Broker w trybie zgodności. Ta wartość jest wartością domyślną, jeśli dla parametru TRANSPORT ustawiono wartość BIND lub CLIENT. • **V2** -aby użyć brokera produktu WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker lub WebSphere Business Integration Message Broker w trybie rodzimym. Ta wartość jest wartością domyślną, jeśli parametr TRANSPORT ma wartość DIRECT lub DIRECTHTTP.	Wersja używanego brokera
ccdtURL	Łańcuch	• **Null** • Adres URL (Uniform Resource Locator)	Adres URL identyfikujący nazwę i położenie pliku zawierającego tabelę definicji kanału klienta (CCDT) oraz określający sposób dostępu do pliku
CCSID	Łańcuch	• **819** • Identyfikator kodowanego zestawu znaków obsługiwany przez wirtualną maszynę języka Java (JVM)	Identyfikator kodowanego zestawu znaków dla połączenia
kanał	Łańcuch	• **SYSTEM.DEF.SVRCONN** • Nazwa kanału MQI	Nazwa kanału MQI, który ma być używany

Tabela 96. Właściwości obiektu ActivationSpec używane do tworzenia połączenia JMS (kontynuacja)			
Nazwa właściwości	**Typ**	**Poprawne wartości (wartość domyślna pogrubiona)**	**Opis**
cleanupInterval[1]	int	• **3 600 000** • Dodatnia liczba całkowita	Odstęp czasu (w milisekundach) między uruchomieniami w tle programu narzędziowego procedury czyszczącej publikowania/subskrypcji
cleanupLevel[1] ,	Łańcuch	• **Bezpieczne** • BRAK • silny • Wymuszenie • NIEDUR	Poziom procedury czyszczącej dla składnicy subskrypcji opartej na brokerze
clientID	Łańcuch	• **Null** • Identyfikator klienta	Identyfikator klienta dla połączenia
cloneSupport	Łańcuch	• **DISABLED** -w danym momencie może działać tylko jedna instancja trwałego subskrybenta tematów. • ENABLED-dwie lub więcej instancji tego samego stałego subskrybenta tematów może działać jednocześnie, ale każda instancja musi działać w oddzielnej wirtualnej maszynie języka Java (JVM).	Określa, czy dwie lub więcej instancji tego samego stałego subskrybenta tematów może działać równocześnie.

Tabela 96. Właściwości obiektu ActivationSpec używane do tworzenia połączenia JMS (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
Lista connectionName	Łańcuch	• **host lokalny (1414)** • Łańcuch składający się z elementów oddzielonych przecinkami, w którym każdy element ma następujący format: `HOSTNAME(PORT)` gdzie *NAZWAHOSTA* jest nazwą DNS lub adresem IP.	Lista nazw połączeń TCP/IP używanych dla komunikacji przychodzącej. Jeśli ta opcja jest określona, **connectionNameList** zastępuje właściwości **hostname** i **port** . Ta właściwość jest używana do ponownego nawiązywania połączenia z menedżerami kolejek z wieloma instancjami. Format **connectionNameList** jest podobny do formatu **localAddress**, ale nie można go z nim mylić. Parametr **localAddress** określa parametry komunikacji lokalnej, natomiast parametr **connectionNameList** określa sposób nawiązania połączenia ze zdalnym menedżerem kolejek.
FAILIFQUIESCE	wartość boolowska	• **Prawda** • Fałsz	Określa, czy wywołania niektórych metod nie powiodą się, jeśli menedżer kolejek jest w stanie wyciszania
headerCompression	Łańcuch	• **Brak** • SYSTEM-kompresja nagłówka komunikatu RLE jest wykonywana	Lista technik, których można użyć do kompresji danych nagłówka w połączeniu
hostName	Łańcuch	• **localhost** • Nazwa hosta • Adres IP	Nazwa hosta lub adres IP systemu, w którym znajduje się menedżer kolejek. Właściwości **hostname** i **port** są zastępowane przez właściwość **connectionNameList** , gdy jest ona określona.

Tabela 96. Właściwości obiektu ActivationSpec używane do tworzenia połączenia JMS (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
localAddress	Łańcuch	• **Null** • Łańcuch w formacie: `[host_name][(low_port[,high_port])]` gdzie *nazwa_hosta* jest nazwą hosta lub adresem IP, *low_port* i *high_port* są numerami portów TCP, a nawiasy kwadratowe oznaczają komponent opcjonalny.	W przypadku połączenia z menedżerem kolejek ta właściwość określa jedną lub obie następujące elementy: • Lokalny interfejs sieciowy, który ma być używany • Port lokalny lub zakres portów lokalnych, które mają być używane Format **localAddress** jest podobny do formatu **connectionNameList**, ale nie można go z nim mylić. Parametr **localAddress** określa parametry komunikacji lokalnej, natomiast parametr **connectionNameList** określa sposób nawiązania połączenia ze zdalnym menedżerem kolejek.
messageCompression	Łańcuch	• **Brak** • Lista zawierająca jedną lub więcej następujących wartości rozdzielonych znakami odstępu: RLE ZLIBFAST ZLIBHIGH	Lista technik, których można użyć do kompresowania danych komunikatu w połączeniu
Atrybut messageRetention[1]	wartość boolowska	• **true** (prawda)-w kolejce wejściowej pozostają niepotrzebne komunikaty. • false-Nieżądane komunikaty są przetwarzane zgodnie z ich opcjami dyspozycji	Określa, czy konsument połączenia przechowuje niepożądane komunikaty w kolejce wejściowej.
messageSelection[1]	Łańcuch	• **client** • BROKER	Określa, czy wybór komunikatów jest dokonywany przez klasy WebSphere MQ dla usługi JMS, czy przez broker. Wybór komunikatu przez broker nie jest obsługiwany, jeśli właściwość brokerVersion ma wartość 1.

Tabela 96. Właściwości obiektu ActivationSpec używane do tworzenia połączenia JMS (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
Hasło	Łańcuch	• **Null** • Hasło	Domyślne hasło używane podczas tworzenia połączenia z menedżerem kolejek
pollingInterval[1]	int	• **5000** • Dowolna dodatnia liczba całkowita	Jeśli kolejka żadnego procesu nasłuchującego w ramach sesji nie zawiera odpowiedniego komunikatu, jest to maksymalny odstęp czasu (w milisekundach) między kolejnymi próbami pobrania komunikatu z kolejki przez każdy z procesów nasłuchujących komunikatów. Jeśli często zdarza się, że dla żadnego z obiektów nasłuchiwania komunikatów w sesji nie jest dostępny odpowiedni komunikat, należy rozważyć zwiększenie wartości tej właściwości. Ta właściwość ma zastosowanie tylko wtedy, gdy właściwość TRANSPORT ma wartość BIND lub CLIENT.
Port	int	• **1414** • Numer portu TCP	Port, na którym nasłuchuje menedżer kolejek. Właściwości **hostname** i **port** są zastępowane przez właściwość **connectionNameList** , gdy jest ona określona.
providerVersion	string (łańcuch)	• **nieokreślona** • Łańcuch w jednym z następujących formatów – V.R.M.F – V.R.M – V.R – V gdzie V, R, M i F są wartościami całkowitymi większymi lub równymi zeru.	Wersja, wydanie, poziom modyfikacji i pakiet poprawek menedżera kolejek, z którym ma zostać połączony komponent MDB.
queueManager	Łańcuch	• **"" (pusty łańcuch)** • Nazwa menedżera kolejek	Nazwa menedżera kolejek, z którym ma zostać nawiązane połączenie

Tabela 96. Właściwości obiektu ActivationSpec używane do tworzenia połączenia JMS (kontynuacja)			
Nazwa właściwości	**Typ**	**Poprawne wartości (wartość domyślna pogrubiona)**	**Opis**
receiveExit [3]	Łańcuch	• **Null** • Łańcuch składający się z jednego lub wielu elementów oddzielonych przecinkami, gdzie każdy element jest pełną nazwą klasy implementującej klasy WebSphere MQ dla interfejsu Java (MQReceiveExit).	Identyfikuje program obsługi wyjścia odbierania kanału lub sekwencję programów obsługi wyjścia odbierania, które mają być uruchamiane po sobie
Inicjowanie receiveExit	Łańcuch	• **Null** • Łańcuch składający się z jednego lub kilku elementów danych użytkownika oddzielonych przecinkami	Dane użytkownika przekazywane do programów obsługi wyjścia odbierania kanału podczas ich wywołania
rescanInterval[1]	int	• **5000** • Dowolna dodatnia liczba całkowita	Jeśli konsument komunikatów w domenie typu punkt z punktem używa selektora komunikatów do wybrania komunikatów, które mają być odbierane, klasy WebSphere MQ classes for JMS przeszukują kolejkę WebSphere MQ pod kątem odpowiednich komunikatów w kolejności określonej przez atrybut *MsgDeliverySequence* kolejki. Gdy klasy WebSphere MQ classes for JMS znajdą odpowiedni komunikat i dostarczą go do konsumenta, klasy WebSphere MQ classes for JMS wznowią wyszukiwanie następnego odpowiedniego komunikatu od jego bieżącej pozycji w kolejce. WebSphere Klasy MQ dla usługi JMS będą przeszukiwać kolejkę w ten sposób, dopóki nie osiągnie końca kolejki lub dopóki nie upłynie odstęp czasu w milisekundach określony przez wartość tej właściwości. W każdym przypadku klasy WebSphere MQ classes for JMS powracają do początku kolejki, aby kontynuować wyszukiwanie, oraz rozpoczyna się nowy przedział czasu.

Tabela 96. Właściwości obiektu ActivationSpec używane do tworzenia połączenia JMS (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
securityExit [3]	Łańcuch	• **Null** • Pełna nazwa klasy implementującej klasy WebSphere MQ dla interfejsu Java, MQSecurityExit	Identyfikuje program obsługi wyjścia zabezpieczeń kanału
securityExit-inicjowanie	Łańcuch	• **Null** • Łańcuch danych użytkownika	Dane użytkownika przekazywane do programu obsługi wyjścia zabezpieczeń kanału podczas jego wywołania
sendExit [3]	Łańcuch	• **Null** • Łańcuch składający się z jednego lub większej liczby elementów oddzielonych przecinkami, gdzie każdy element jest pełną nazwą klasy implementującej klasy WebSphere MQ dla interfejsu Java, MQSendExit	Identyfikuje program obsługi wyjścia wysyłania kanału lub sekwencję programów obsługi wyjścia wysyłania, które mają być uruchamiane po sobie
SENDEXITINIT	Łańcuch	• **Null** • Łańcuch składający się z jednego lub kilku elementów danych użytkownika oddzielonych przecinkami	Dane użytkownika przekazywane do programów obsługi wyjścia wysyłania kanału podczas ich wywołania
SHARECONVALLOWED	wartość boolowska	• NIE-Połączenie klienta nie może współużytkować swojego gniazda. • **YES** -połączenie klienta może współużytkować swoje gniazdo.	Określa, czy połączenie klienta może współużytkować gniazdo z innymi połączeniami JMS najwyższego poziomu z tego samego procesu do tego samego menedżera kolejek, jeśli definicje kanałów są zgodne.
sparseSubscriptions[1]	wartość boolowska	• **false** -subskrypcje otrzymują często zgodne komunikaty. • true-subskrypcje otrzymują rzadko zgodne komunikaty. Ta wartość wymaga, aby kolejka subskrypcji mogła zostać otwarta do przeglądania.	Steruje strategią pobierania komunikatów obiektu TopicSubscriber

Tabela 96. Właściwości obiektu ActivationSpec używane do tworzenia połączenia JMS (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
Magazyny sslCert	Łańcuch	• **Null** • Łańcuch zawierający jeden lub więcej adresów URL LDAP rozdzielonych odstępami. Każdy adres URL LDAP ma następujący format: `ldap://host_name[:port]` gdzie *nazwa_hosta* jest nazwą hosta lub adresem IP, *port* jest numerem portu TCP, a nawiasy kwadratowe oznaczają komponent opcjonalny.	Serwery LDAP (Lightweight Directory Access Protocol), które przechowują listy odwołań certyfikatów (CRL) do użycia w połączeniu SSL.
SSLCIPHERSUITE	Łańcuch	• **Null** • Nazwa CipherSuite	CipherSuite , który ma być używany na potrzeby połączenia SSL.
sslFipsWymagane[2]	wartość boolowska	• **Fałsz** • Prawda	Określa, czy połączenie SSL musi używać pakietu CipherSuite obsługiwanego przez dostawcę IBM Java JSSE FIPS (IBMJSSEFIPS).
SSLPEERNAME	Łańcuch	• **Null** • Szablon nazw wyróżniających	W przypadku połączenia SSL: szablon używany do sprawdzania nazwy wyróżniającej w certyfikacie cyfrowym udostępnianym przez menedżer kolejek.
SSLRESETCOUNT	int	• **0** • Liczba całkowita z zakresu od 0 do 999 999 999	Łączna liczba bajtów wysłanych i odebranych przez połączenie SSL przed ponownym negocjowaniem kluczy tajnych używanych przez SSL
Fabryka sslSocket	Łańcuch	Łańcuch reprezentujący pełną nazwę klasy udostępniającej implementację interfejsu javax.net.ssl.SSLSocketFactory . Opcjonalnie z dołączonym argumentem, który ma zostać przekazany do metody konstruktora, ujętym w nawiasy.	Wszystkie połączenia nawiązane w zasięgu obiektu administrowanego używają gniazd uzyskanych z tej implementacji interfejsu SSLSocketFactory .

Tabela 96. Właściwości obiektu ActivationSpec używane do tworzenia połączenia JMS (kontynuacja)			
Nazwa właściwości	**Typ**	**Poprawne wartości (wartość domyślna pogrubiona)**	**Opis**
statusRefreshInterval[1]	int	• **60000** • Dowolna dodatnia liczba całkowita	Odstęp czasu (w milisekundach) między operacjami odświeżania długotrwałych transakcji, które wykrywają, kiedy subskrybent traci połączenie z menedżerem kolejek. Ta właściwość ma zastosowanie tylko wtedy, gdy właściwość subscriptionStore ma wartość QUEUE.
subscriptionStore[1]	Łańcuch	• **BROKER** • MIGRATE • QUEUE	Określa, gdzie klasy WebSphere MQ classes for JMS przechowują dane trwałe dotyczące aktywnych subskrypcji.
transportType	Łańcuch	• **client** • POWIĄZANIA • BINDINGS_THEN_CLIENT (POWIĄZANIA, NASTĘPNIE KLIENT)	Określa, czy połączenie z menedżerem kolejek używa trybu klienta, czy trybu powiązań. Jeśli zostanie podana wartość BINDINGS_THEN_CLIENT, adapter zasobów najpierw próbuje nawiązać połączenie w trybie powiązań. Jeśli ta próba nawiązania połączenia nie powiedzie się, adapter zasobów podejmie próbę nawiązania połączenia w trybie klienta.
Nazwa użytkownika	Łańcuch	• **Null** • Nazwa użytkownika	Domyślna nazwa użytkownika, która ma być używana podczas tworzenia połączenia z menedżerem kolejek
wildcardFormat	Łańcuch	• CHAR-rozpoznaje tylko znaki wieloznaczne używane w brokerze w wersji 1 • **TOPIC** -rozpoznaje tylko znaki wieloznaczne na poziomie tematu używane w wersji 2 brokera.	Która wersja składni znaku wieloznacznego ma być używana

Uwagi:

1. Ta właściwość może być używana z wersją 7.0 klas WebSphere MQ classes for JMS. Nie ma to wpływu na aplikację połączoną z menedżerem kolejek w wersji 7.0 , chyba że właściwość providerVersion jest ustawiona na numer wersji mniejszy niż 7.
2. Ważne informacje na temat używania wymaganej właściwości sslFipszawiera sekcja “Ograniczenia adaptera zasobów produktu IBM WebSphere MQ” na stronie 792.

3. Informacje na temat konfigurowania adaptera zasobów w taki sposób, aby mógł on znaleźć wyjście, zawiera sekcja “Konfigurowanie produktu IBM WebSphere MQ classes for JMS do korzystania z wyjść kanału” na stronie 939.

Tabela 97 na stronie 767 zawiera listę właściwości obiektu ActivationSpec , które są używane do tworzenia konsumenta połączenia JMS.

Tabela 97. Właściwości obiektu ActivationSpec używane do tworzenia konsumenta połączenia JMS

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
miejsce docelowe	Łańcuch	Nazwa miejsca docelowego	Miejsce docelowe, z którego mają być odbierane komunikaty. Właściwość useJNDI określa sposób interpretowania wartości tej właściwości.
destinationType	Łańcuch	• javax.jms.Queue • javax.jms.Topic	Typ miejsca docelowego, kolejki lub tematu
maxMessages	int	• **1** • Dodatnia liczba całkowita	Maksymalna liczba komunikatów, które można jednocześnie przypisać do sesji serwera. Jeśli specyfikacja aktywowania dostarcza komunikaty do komponentu MDB w transakcji XA, niezależnie od ustawienia tej właściwości używana jest wartość 1.
maxPoolGłębokość	int	• **10** • Dodatnia liczba całkowita	Maksymalna liczba sesji serwera w puli sesji serwera używanych przez konsumenta połączenia
messageSelector	Łańcuch	• **Null** • Wyrażenie selektora komunikatów SQL92	Wyrażenie selektora komunikatów określające, które komunikaty mają zostać dostarczone
nonASFTimeout	int	• **0** • Dodatnia liczba całkowita	Wartość dodatnia wskazuje, że używana jest dostawa bez ASF. Wartością jest czas (w milisekundach), przez który żądanie pobrania oczekuje na komunikaty, które nie zostały jeszcze wysłane (wywołanie pobrania z oczekiwaniem). Wartość domyślna, 0, wskazuje, że używane jest dostarczanie ASF. Ten parametr jest poprawny tylko wtedy, gdy aplikacja działa na serwerze WebSphere Application Server w wersji 7 lub nowszej.

Tabela 97. Właściwości obiektu ActivationSpec używane do tworzenia konsumenta połączenia JMS (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
Włączona opcja nonASFRollback	wartość boolowska	• **false** (fałsz)-komunikat jest pobierany nawet wtedy, gdy komponent MDB zakończy działanie niepowodzeniem. • true-Niepowodzenie w obrębie komponentu MDB powoduje wycofanie komunikatu do kolejki.	Określa, czy dostarczanie komunikatów znajduje się w punkcie synchronizacji WebSphere MQ , jeśli komponent MDB nie jest transakcją. Ignorowany, jeśli komponent MDB jest transakcją lub jeśli parametr nonASFTimeout ma wartość 0.
poolTimeout	int	• **300000** • Dodatnia liczba całkowita	Czas (w milisekundach), przez który nieużywana sesja serwera jest utrzymywana w puli sesji serwera przed zamknięciem z powodu nieaktywności
READAHEADALLOWED	int	• **DESTINATION** -określa, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji kolejki lub tematu. • DISABLED-odczyt z wyprzedzeniem jest niedozwolony. • ENABLED-odczyt z wyprzedzeniem jest dozwolony. • QUEUE-określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji kolejki. • TOPIC-Określenie, czy odczyt z wyprzedzeniem jest dozwolony, poprzez odwołanie się do definicji tematu.	Określa, czy komponent MDB może używać odczytu z wyprzedzeniem do pobierania nietrwałych komunikatów z miejsca docelowego do buforu wewnętrznego przed ich odebraniem.
readAheadClosePolicy	int	• **ALL** -wszystkie komunikaty w wewnętrznym buforze odczytu z wyprzedzeniem są dostarczane do komponentu MDB przed jego zatrzymaniem. • CURRENT-kończy się tylko bieżące wywołanie komponentu MDB, potencjalnie pozostawiając komunikaty w wewnętrznym buforze odczytu z wyprzedzeniem, które są następnie usuwane.	Co się dzieje z komunikatami w wewnętrznym buforze odczytu z wyprzedzeniem, gdy komponent MDB zostanie zatrzymany przez administratora.

Tabela 97. Właściwości obiektu ActivationSpec używane do tworzenia konsumenta połączenia JMS (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
receiveCCSID	int	• **0** -użyj maszyny JVM `Charset.defaultCharset` • 1208- UTF-8 • Obsługiwany identyfikator kodowanego zestawu znaków	Właściwość miejsca docelowego, która ustawia docelowy identyfikator CCSID dla konwersji komunikatów menedżera kolejek. Wartość jest ignorowana, chyba że parametr **`receiveConversion`** ma wartość `QMGR`
receiveConversion	Łańcuch	• **KOMUNIKAT KLIENTA** • QMGR	Właściwość miejsca docelowego, która określa, czy konwersja danych będzie wykonywana przez menedżer kolejek.
startTimeout	int	• **10 000** • Dodatnia liczba całkowita	Czas (w milisekundach), w którym dostarczanie komunikatu do komponentu MDB musi zostać rozpoczęte po zaplanowaniu pracy nad dostarczeniem komunikatu. Jeśli ten czas upłynie, komunikat zostanie wycofany do kolejki.
subscriptionDurability	Łańcuch	• **NonDurable** -Nietrwała subskrypcja jest używana do dostarczania komunikatów do komponentu MDB subskrybującego temat. • Trwała-trwała subskrypcja jest używana do dostarczania komunikatów do komponentu MDB subskrybującego temat.	Określa, czy do dostarczania komunikatów do komponentu MDB subskrybującego temat używana jest trwała, czy nietrwała subskrypcja.
subscriptionName	Łańcuch	• **"" (pusty łańcuch)** • Nazwa subskrypcji	Nazwa trwałej subskrypcji
useJNDI	wartość boolowska	• **false** -właściwość o nazwie destination jest interpretowana jako nazwa kolejki lub tematu produktu WebSphere MQ . • true-właściwość o nazwie destination jest interpretowana jako nazwa obiektu javax.jms.Queue lub obiektu javax.jms.Topic w przestrzeni nazw JNDI serwera aplikacji.	Określa sposób interpretowania wartości właściwości o nazwie destination.

Właściwości ActivationSpec o nazwie destination i destinationType muszą być jawnie zdefiniowane. Wszystkie pozostałe właściwości są opcjonalne.

Obiekt ActivationSpec może mieć właściwości powodujące konflikt. Na przykład można określić właściwości SSL dla połączenia w trybie powiązań. W takim przypadku zachowanie jest określane przez typ transportu i domenę przesyłania komunikatów, czyli punkt z punktem lub publikowanie/subskrypcja,

zgodnie z wartością właściwości destinationType . Wszystkie właściwości, które nie mają zastosowania do określonego typu transportu lub domeny przesyłania komunikatów, są ignorowane.

Jeśli zostanie zdefiniowana właściwość, która wymaga zdefiniowania innych właściwości, ale nie zostaną one zdefiniowane, obiekt ActivationSpec zgłosi wyjątek InvalidPropertypodczas wywoływania metody validate () podczas wdrażania komponentu MDB. Wyjątek jest zgłaszany administratorowi serwera aplikacji w sposób zależny od serwera aplikacji. Jeśli na przykład właściwość subscriptionDurability zostanie ustawiona na wartość Durable, co oznacza, że mają być używane trwałe subskrypcje, należy również zdefiniować właściwość subscriptionName .

Jeśli zdefiniowano zarówno właściwości o nazwie ccdtURL , jak i kanał, zgłaszany jest wyjątek InvalidProperty. Jednak w przypadku definiowania tylko właściwości ccdtURL , pozostawiając właściwość o nazwie channel z wartością domyślną SYSTEM.DEF.SVRCONN, nie jest zgłaszany żaden wyjątek, a tabela definicji kanału klienta identyfikowana przez właściwość ccdtURL jest używana do uruchamiania połączenia JMS.

Większość właściwości obiektu ActivationSpec są równoważne właściwościom klas WebSphere MQ dla obiektów JMS lub parametrów klas WebSphere MQ dla metod JMS. Jednak trzy właściwości strojenia i jedna właściwość łatwości używania nie mają odpowiedników w klasach WebSphere MQ dla usługi JMS:

startTimeout
Czas (w milisekundach), przez który menedżer pracy serwera aplikacji oczekuje na udostępnienie zasobów po zaplanowaniu przez adapter zasobów obiektu pracy w celu dostarczenia komunikatu do komponentu MDB. Jeśli ten czas upłynie przed rozpoczęciem dostarczania komunikatu, nastąpi przekroczenie limitu czasu obiektu roboczego, komunikat zostanie wycofany do kolejki, a adapter zasobów będzie mógł podjąć ponowną próbę dostarczenia komunikatu. Ostrzeżenie jest zapisywane w danych śledzenia diagnostycznego, jeśli jest włączone, ale nie ma wpływu na proces dostarczania komunikatów. Można oczekiwać, że ten warunek wystąpi tylko wtedy, gdy serwer aplikacji jest bardzo obciążony. Jeśli warunek występuje regularnie, należy rozważyć zwiększenie wartości tej właściwości, aby zapewnić menedżerowi pracy więcej czasu na zaplanowanie dostarczania komunikatów.

maxPoolGłębokość
Maksymalna liczba sesji serwera w puli sesji serwera używana przez konsumenta połączenia. Po utworzeniu sesji serwera rozpoczyna ona konwersację z menedżerem kolejek. Konsument połączenia używa sesji serwera do dostarczenia komunikatu do komponentu MDB. Większa głębokość puli pozwala na współbieżne dostarczanie większej liczby komunikatów w sytuacjach dużego wolumenu, ale zużywa więcej zasobów serwera aplikacji. Jeśli ma zostać wdrożonych wiele komponentów MDB, należy rozważyć zmniejszenie głębokości puli w celu utrzymania obciążenia serwera aplikacji na poziomie umożliwiającym zarządzanie. Każdy konsument połączenia używa własnej puli sesji serwera, dlatego ta właściwość nie definiuje łącznej liczby sesji serwera dostępnych dla wszystkich konsumentów połączenia.

poolTimeout
Czas (w milisekundach), przez który nieużywana sesja serwera jest otwarta w puli sesji serwera przed zamknięciem z powodu braku aktywności. Przejściowy wzrost obciążenia komunikatami powoduje utworzenie dodatkowych sesji serwera w celu rozdzielenia obciążenia, ale po powrocie do normalnego obciążenia komunikatami dodatkowe sesje serwera pozostają w puli i nie są używane.

Za każdym razem, gdy używana jest sesja serwera, jest ona oznaczana znacznikiem czasu. Okresowo wątek scavenger sprawdza, czy każda sesja serwera była używana w okresie określonym przez tę właściwość. Jeśli sesja serwera nie została użyta, zostanie zamknięta i usunięta z puli sesji serwera. Sesja serwera może nie zostać zamknięta natychmiast po upływie podanego okresu. Ta właściwość reprezentuje minimalny okres braku aktywności przed usunięciem.

useJNDI
Opis tej właściwości zawiera sekcja Tabela 97 na stronie 767.

Aby wdrożyć komponent MDB, należy najpierw zdefiniować właściwości obiektu ActivationSpec , określając właściwości wymagane przez komponent MDB. W poniższym przykładzie przedstawiono typowy zestaw właściwości, które można jawnie zdefiniować:

```
channel:          SYSTEM.DEF.SVRCONN
destination:      SYSTEM.DEFAULT.LOCAL.QUEUE
destinationType:  javax.jms.Queue
```

```
hostName:         192.168.0.42
messageSelector:  color='red'
port:             1414
queueManager:     ExampleQM
transportType:    CLIENT
```

Serwer aplikacji używa tych właściwości do utworzenia obiektu ActivationSpec , który jest następnie powiązany z komponentem MDB. Właściwości obiektu ActivationSpec określają sposób dostarczania komunikatów do komponentu MDB. Wdrożenie komponentu MDB kończy się niepowodzeniem, jeśli komponent MDB wymaga rozproszonych transakcji, ale adapter zasobów nie obsługuje rozproszonych transakcji. Informacje na temat instalowania adaptera zasobów w celu obsługi transakcji rozproszonych zawiera sekcja “Instalowanie adaptera zasobów produktu WebSphere MQ” na stronie 753.

Jeśli więcej niż jeden komponent MDB odbiera komunikaty z tego samego miejsca docelowego, komunikat wysłany w domenie typu punkt z punktem jest odbierany tylko przez jeden komponent MDB, nawet jeśli inne komponenty MDB są uprawnione do odbierania komunikatu. W szczególności, jeśli dwie bazy danych MDB używają różnych selektorów komunikatów, a komunikat przychodzący jest zgodny z obydwoma selektorami komunikatów, tylko jedna z nich odbiera komunikat. Komponent MDB wybrany do odebrania komunikatu jest niezdefiniowany i nie można polegać na konkretnym komunikacie odebrany przez komponent MDB. Komunikaty wysyłane w domenie publikowania/subskrypcji są odbierane przez wszystkie zakwalifikowane komponenty MDB.

Obsługa przychodzących nieprzetwarzalnych komunikatów w adapterze zasobów

W pewnych okolicznościach komunikat dostarczony do komponentu MDB może zostać wycofany do kolejki WebSphere MQ . Taka sytuacja może mieć miejsce na przykład wtedy, gdy komunikat jest dostarczany w ramach jednostki pracy, która jest następnie wycofywana. Wycofany komunikat jest ponownie dostarczany, ale niepoprawnie sformatowany komunikat może wielokrotnie spowodować niepowodzenie komponentu MDB i dlatego nie można go dostarczyć. Taka wiadomość jest nazywana wiadomością nieprzetwarzaną. Produkt WebSphere MQ można skonfigurować w taki sposób, aby klasy WebSphere MQ classes for JMS automatycznie przesyłały komunikat nieprzetwarzalny do innej kolejki w celu dalszego badania lub usunięcia komunikatu.

Szczegółowe informacje na temat obsługi komunikatów nieprzetwarzalnych zawiera sekcja “Obsługa komunikatów trujących w produkcie IBM WebSphere MQ classes for JMS” na stronie 914.

Zadania pokrewne

Określenie, że w czasie wykonywania na kliencie MQI będą używane tylko CipherSpecs z certyfikatem FIPS.

Odsyłacze pokrewne

Standardy FIPS (Federal Information Processing Standards) dla systemów UNIX, Linux i Windows

Konfigurowanie adaptera zasobów na potrzeby komunikacji wychodzącej

Aby skonfigurować komunikację wychodzącą, należy zdefiniować właściwości obiektu ConnectionFactory i administrowanego obiektu docelowego.

W przypadku korzystania z komunikacji wychodzącej aplikacja działająca na serwerze aplikacji uruchamia połączenie z menedżerem kolejek, a następnie wysyła komunikaty do swoich kolejek i odbiera komunikaty ze swoich kolejek w sposób synchroniczny. Na przykład następująca metoda serwletu, doGet(), używa komunikacji wychodzącej:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException {

...

// Look up ConnectionFactory and Queue objects from the JNDI namespace

     InitialContext ic = new InitialContext();
     ConnectionFactory cf = (javax.jms.ConnectionFactory) ic.lookup("myCF");
     Queue q = (javax.jms.Queue) ic.lookup("myQueue");

// Create and start a connection

     Connection c = cf.createConnection();
     c.start();
```

```
// Create a session and message producer

     Session s = c.createSession(false, Session.AUTO_ACKNOWLEDGE);
     MessageProducer pr = s.createProducer(q);

// Create and send a message

     Message m = s.createTextMessage("Hello, World!");
     pr.send(m);

// Create a message consumer and receive the message just sent

     MessageConsumer co = s.createConsumer(q);
     Message mr = co.receive(5000);

// Close the connection

      c.close();
}
```

Gdy serwlet odbiera żądanie HTTP GET, pobiera obiekt ConnectionFactory i obiekt Queue z przestrzeni nazw JNDI i używa tych obiektów do wysłania komunikatu do kolejki WebSphere MQ . Następnie serwlet odbiera wysłany przez niego komunikat.

Aby skonfigurować komunikację wychodzącą, należy zdefiniować zasoby JCA w następujących kategoriach:

- Właściwości obiektu ConnectionFactory , którego serwer aplikacji używa do utworzenia obiektu JMS ConnectionFactory .
- Właściwości administrowanego obiektu docelowego, którego serwer aplikacji używa do utworzenia obiektu kolejki JMS lub obiektu tematu JMS.

Sposób definiowania tych właściwości zależy od interfejsów administracyjnych udostępnianych przez serwer aplikacji. ConnectionFactory, Queue i Topic utworzone przez serwer aplikacji są powiązane z przestrzenią nazw JNDI, z której mogą być pobierane przez aplikację.

Zwykle dla każdego menedżera kolejek, z którym może być konieczne nawiązanie połączenia przez aplikacje, definiowany jest jeden obiekt ConnectionFactory . Należy zdefiniować jeden obiekt kolejki dla każdej kolejki, do której aplikacje mogą potrzebować dostępu w domenie typu punkt z punktem. Dla każdego tematu, który może być publikowany lub subskrybowany przez aplikacje, należy zdefiniować jeden obiekt tematu. Obiekt ConnectionFactory może być niezależny od domeny. Alternatywnie może to być obiekt specyficzny dla domeny, obiekt fabryki QueueConnectiondla domeny typu punkt z punktem lub obiekt fabryki TopicConnectiondla domeny publikowania/subskrypcji.

Tabela 98 na stronie 772 zawiera listę właściwości obiektu ConnectionFactory .

Tabela 98. Właściwości obiektu ConnectionFactory

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
applicationName	Łańcuch	• Nazwa klasy wywołującej, jeśli jest dostępna, nie może być dłuższa niż 28 znaków. Jeśli nie jest dostępny, zostanie użyty łańcuch `WebSphere MQ Client for Java` .	Nazwa, pod którą aplikacja jest rejestrowana w menedżerze kolejek. Ta nazwa aplikacji jest wyświetlana przez komendę **DISPLAY CONN MQSC/PCF** (gdzie pole ma nazwę **APPLTAG**) lub na ekranie **Połączenia aplikacji** w Eksploratorze IBM WebSphere MQ (gdzie pole ma nazwę **App name**).
brokerCCSubbrokerCCSub[1]	Łańcuch	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Nazwa kolejki	Nazwa kolejki, z której konsument połączenia odbiera komunikaty nietrwałej subskrypcji.

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
brokerControlbrokerControl[1]	Łańcuch	• **SYSTEM.BROKER.CONTROL.QUEUE** • Nazwa kolejki	Nazwa kolejki sterującej brokera.
Kolejka brokerPubQueue[1]	Łańcuch	• **SYSTEM.BROKER.DEFAULT.STREAM** • Nazwa kolejki	Nazwa kolejki, do której wysyłane są opublikowane komunikaty (kolejka strumienia).
brokerQueuekolejki brokera[1]	Łańcuch	• **"" (pusty łańcuch)** • Nazwa menedżera kolejek	Nazwa menedżera kolejek, w którym działa broker.
brokerSubQueue[1]	Łańcuch	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Nazwa kolejki	Nazwa kolejki, z której konsument nietrwałych komunikatów odbiera komunikaty. Więcej informacji na ten temat zawiera opis właściwości BROKERSUBQ .
Wersja brokera[1]	Łańcuch	• **unspecified** (nieokreślony)-po przeprowadzeniu migracji brokera z wersji V6 do wersji V7należy ustawić tę właściwość, aby nagłówki RFH2 nie były już używane. Po migracji ta właściwość nie jest już istotna. • **V1** -aby użyć brokera publikowania/subskrypcji produktu IBM WebSphere MQ . lub użyć brokera produktu IBM WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker lub WebSphere Business Integration Message Broker w trybie zgodności. Ta wartość jest wartością domyślną, jeśli dla parametru TRANSPORT ustawiono wartość BIND lub CLIENT. • **V2** -aby użyć brokera produktu IBM WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker lub WebSphere Business Integration Message Broker w trybie rodzimym. Ta wartość jest wartością domyślną, jeśli parametr TRANSPORT ma wartość DIRECT lub DIRECTHTTP.	Wersja używanego brokera.
ccdtURL	Łańcuch	• **Null** • Adres URL (Uniform Resource Locator)	Adres URL identyfikujący nazwę i położenie pliku zawierającego tabelę definicji kanału klienta (CCDT) oraz określający sposób dostępu do pliku.

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)			
Nazwa właściwości	**Typ**	**Poprawne wartości (wartość domyślna pogrubiona)**	**Opis**
CCSID	Łańcuch	• **819** • Identyfikator kodowanego zestawu znaków obsługiwany przez wirtualną maszynę języka Java (JVM)	Identyfikator kodowanego zestawu znaków dla połączenia.
kanał	Łańcuch	• **SYSTEM.DEF.SVRCONN** • Nazwa kanału MQI	Nazwa kanału MQI, który ma być używany.
cleanupInterval[1]	int	• **3 600 000** • Dodatnia liczba całkowita	Odstęp czasu (w milisekundach) między uruchomieniami w tle programu narzędziowego do czyszczenia publikowania/ subskrypcji.
cleanupLevel[1] ,	Łańcuch	• **Bezpieczne** • BRAK • silny • Wymuszenie • NIEDUR	Poziom procedury czyszczącej dla składnicy subskrypcji opartej na brokerze.
clientID	Łańcuch	• **Null** • Identyfikator klienta	Identyfikator klienta dla połączenia.
cloneSupport	Łańcuch	• **DISABLED** -w danym momencie może działać tylko jedna instancja trwałego subskrybenta tematów. • ENABLED-dwie lub więcej instancji tego samego trwałego subskrybenta tematów może działać równocześnie, ale każda instancja musi działać w oddzielnej wirtualnej maszynie języka Java (JVM).	Określa, czy dwie lub więcej instancji tego samego trwałego subskrybenta tematu może działać równocześnie.

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
Lista connectionName	Łańcuch	• **host lokalny (1414)** • Łańcuch składający się z elementów oddzielonych przecinkami, w którym każdy element ma następujący format: `HOSTNAME(PORT)` gdzie *NAZWAHOSTA* jest nazwą DNS lub adresem IP.	Lista nazw połączeń TCP/IP używanych do komunikacji wychodzącej. **connectionNameList** zastępuje właściwości **hostname** i **port** . Ta właściwość jest używana do ponownego nawiązywania połączenia z menedżerami kolejek z wieloma instancjami. Format **connectionNameList** jest podobny do formatu **localAddress**, ale nie można go z nim mylić. Parametr **localAddress** określa parametry komunikacji lokalnej, natomiast parametr **connectionNameList** określa sposób nawiązania połączenia ze zdalnym menedżerem kolejek.
FAILIFQUIESCE	wartość boolowska	• **Prawda** • Fałsz	Określa, czy wywołania niektórych metod nie powiodą się, jeśli menedżer kolejek jest w stanie wyciszania.
headerCompression	Łańcuch	• **Brak** • Kompresja nagłówka komunikatu SYSTEM-RLE jest wykonywana.	Lista technik, których można użyć do kompresji danych nagłówka w połączeniu.
hostName	Łańcuch	• **localhost** • Nazwa hosta • Adres IP	Nazwa hosta lub adres IP systemu, w którym znajduje się menedżer kolejek. Właściwości **hostname** i **port** są zastępowane przez właściwość **connectionNameList** , gdy jest ona określona.

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis

localAddress	Łańcuch	• **Null**
		• Łańcuch w formacie:
		`[host_name]`
		`[(low_port[,high_port])]`
		gdzie *nazwa_hosta* jest nazwą hosta lub adresem IP, *low_port* i *high_port* są numerami portów TCP, a nawiasy kwadratowe oznaczają komponent opcjonalny.
			W przypadku połączenia z menedżerem kolejek ta właściwość określa jedną lub obie z następujących wartości:
			• Lokalny interfejs sieciowy, który ma być używany
			• Port lokalny lub zakres portów lokalnych, które mają być używane
			Format **localAddress** jest podobny do formatu **connectionNameList**, ale nie można go z nim mylić. Parametr **localAddress** określa parametry komunikacji lokalnej, natomiast parametr **connectionNameList** określa sposób nawiązania połączenia ze zdalnym menedżerem kolejek.

messageCompression	Łańcuch	• **Brak** • Lista zawierająca jedną lub więcej następujących wartości rozdzielonych znakami odstępu: RLE ZLIBFAST ZLIBHIGH	Lista technik, których można użyć do kompresji danych komunikatu w połączeniu.
messageSelection[1]	Łańcuch	• **client** • BROKER	Określa, czy wybór komunikatów jest dokonywany przez klasy IBM WebSphere MQ dla usługi JMS, czy przez broker. Wybór komunikatu przez broker nie jest obsługiwany, jeśli właściwość brokerVersion ma wartość 1.
Hasło	Łańcuch	• **Null** • Hasło	Hasło domyślne używane podczas tworzenia połączenia z menedżerem kolejek.

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
pollingInterval[1]	int	• **5000** • Dowolna dodatnia liczba całkowita	Jeśli kolejka żadnego procesu nasłuchującego w ramach sesji nie zawiera odpowiedniego komunikatu, jest to maksymalny odstęp czasu (w milisekundach) między kolejnymi próbami pobrania komunikatu z kolejki przez każdy z procesów nasłuchujących komunikatów. Jeśli często zdarza się, że dla żadnego z obiektów nasłuchiwania komunikatów w sesji nie jest dostępny odpowiedni komunikat, należy rozważyć zwiększenie wartości tej właściwości. Ta właściwość ma zastosowanie tylko wtedy, gdy właściwość TRANSPORT ma wartość BIND lub CLIENT.
Port	int	• **1414** • Numer portu TCP	Port, na którym nasłuchuje menedżer kolejek. Właściwości **hostname** i **port** są zastępowane przez właściwość **connectionNameList** , gdy jest ona określona.
providerVersion	string (łańcuch)	• **nieokreślona** • Łańcuch w jednym z następujących formatów – V.R.M.F – V.R.M – V.R – V gdzie V, R, M i F są wartościami całkowitymi większymi lub równymi zeru.	Wersja, wydanie, poziom modyfikacji i pakiet poprawek menedżera kolejek, z którym aplikacja ma nawiązać połączenie.
pubAckInterval[1]	int	• **25** • Dodatnia liczba całkowita	Liczba komunikatów opublikowanych przez publikator, zanim produkt IBM WebSphere MQ classes for JMS zażąda potwierdzenia od brokera.
queueManager	Łańcuch	• **"" (pusty łańcuch)** • Nazwa menedżera kolejek	Nazwa menedżera kolejek, z którym ma zostać nawiązane połączenie.

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
receiveExit [3]	Łańcuch	• **Null** • Łańcuch składający się z jednego lub większej liczby elementów oddzielonych przecinkami, gdzie każdy element jest pełną nazwą klasy implementującej interfejs IBM WebSphere MQ classes for Java (MQReceiveExit).	Identyfikuje program obsługi wyjścia odbierania kanału lub sekwencję programów obsługi wyjścia odbierania, które mają być uruchamiane po sobie.
Inicjowanie receiveExit	Łańcuch	• **Null** • Łańcuch składający się z jednego lub kilku elementów danych użytkownika oddzielonych przecinkami	Dane użytkownika przekazywane do programów obsługi wyjścia odbierania kanału podczas ich wywołania.
rescanInterval[1]	int	• **5000** • Dowolna dodatnia liczba całkowita	Gdy konsument komunikatów w domenie typu punkt z punktem używa selektora komunikatów do wybrania komunikatów, które mają być odbierane, klasy WebSphere MQ dla usługi JMS przeszukują kolejkę IBM WebSphere MQ w poszukiwaniu odpowiednich komunikatów w kolejności określonej przez atrybut *MsgDeliverySequence* kolejki. Gdy klasy WebSphere MQ classes for JMS znajdą odpowiedni komunikat i dostarczą go do konsumenta, klasy WebSphere MQ classes for JMS wznowią wyszukiwanie następnego odpowiedniego komunikatu od jego bieżącej pozycji w kolejce. WebSphere Klasy MQ dla usługi JMS będą przeszukiwać kolejkę w ten sposób, dopóki nie osiągnie końca kolejki lub dopóki nie upłynie odstęp czasu w milisekundach określony przez wartość tej właściwości. W każdym przypadku klasy WebSphere MQ classes for JMS powracają do początku kolejki, aby kontynuować wyszukiwanie, oraz rozpoczyna się nowy przedział czasu.
securityExit [3]	Łańcuch	• **Null** • Pełna nazwa klasy implementującej klasy WebSphere MQ dla interfejsu Java, MQSecurityExit	Identyfikuje program obsługi wyjścia zabezpieczeń kanału.

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
securityExit-inicjowanie	Łańcuch	• **Null** • Łańcuch danych użytkownika	Dane użytkownika przekazywane do programu obsługi wyjścia zabezpieczeń kanału podczas jego wywołania.
SENDCHECKCOUNT	int	• **0** • Dowolna dodatnia liczba całkowita	Liczba dozwolonych wywołań wysyłania między sprawdzaniem występowania błędów asynchronicznego umieszczania w pojedynczej sesji JMS bez transakcji.
sendExit [3]	Łańcuch	• **Null** • Łańcuch składający się z jednego lub większej liczby elementów oddzielonych przecinkami, gdzie każdy element jest pełną nazwą klasy implementującej klasy WebSphere MQ dla interfejsu Java, MQSendExit	Identyfikuje program obsługi wyjścia wysyłania kanału lub sekwencję programów obsługi wyjścia wysyłania, które mają być uruchamiane po sobie.
SENDEXITINIT	Łańcuch	• **Null** • Łańcuch składający się z jednego lub kilku elementów danych użytkownika oddzielonych przecinkami	Dane użytkownika przekazywane do programów obsługi wyjścia wysyłania kanału podczas ich wywołania.
SHARECONVALLOWED	wartość boolowska	• NIE-Połączenie klienta nie może współużytkować swojego gniazda. • **YES** -połączenie klienta może współużytkować swoje gniazdo.	Określa, czy połączenie klienta może współużytkować gniazdo z innymi połączeniami JMS najwyższego poziomu z tego samego procesu do tego samego menedżera kolejek, jeśli definicje kanału są zgodne.
sparseSubscriptions[1]	wartość boolowska	• **false** -subskrypcje otrzymują często zgodne komunikaty. • true-subskrypcje otrzymują rzadko zgodne komunikaty. Ta wartość wymaga, aby kolejka subskrypcji mogła zostać otwarta do przeglądania.	Steruje strategią pobierania komunikatów obiektu TopicSubscriber .
Magazyny sslCert	Łańcuch	• **Null** • Łańcuch zawierający jeden lub więcej adresów URL LDAP rozdzielonych odstępami. Każdy adres URL LDAP ma następujący format: `ldap://host_name[:port]` gdzie *nazwa_hosta* jest nazwą hosta lub adresem IP, *port* jest numerem portu TCP, a nawiasy kwadratowe oznaczają komponent opcjonalny.	Serwery LDAP (Lightweight Directory Access Protocol), które przechowują listy odwołań certyfikatów (CRL) do użycia w połączeniu SSL.

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
SSLCIPHERSUITE	Łańcuch	• **Null** • Nazwa CipherSuite	CipherSuite , który ma być używany dla połączenia SSL.
sslFipsWymagane[2]	wartość boolowska	• **Fałsz** • Prawda	Określa, czy połączenie SSL musi używać pakietu CipherSuite obsługiwanego przez dostawcę IBM Java JSSE FIPS (IBMJSSEFIPS).
SSLPEERNAME	Łańcuch	• **Null** • Szablon nazw wyróżniających	W przypadku połączenia SSL jest to szablon używany do sprawdzania nazwy wyróżniającej w certyfikacie cyfrowym udostępnionym przez menedżer kolejek.
SSLRESETCOUNT	int	• **0** • Liczba całkowita z zakresu od 0 do 999 999 999	Łączna liczba bajtów wysłanych i odebranych przez połączenie SSL przed ponownym negocjowaniem kluczy tajnych używanych przez SSL.
Fabryka sslSocket	Łańcuch	Łańcuch reprezentujący pełną nazwę klasy udostępniającej implementację interfejsu javax.net.ssl.SSLSocketFactory , opcjonalnie zawierający argument, który ma zostać przekazany do metody konstruktora, ujęty w nawiasy.	Wszystkie połączenia nawiązane w zasięgu administrowanego obiektu docelowego używają gniazd uzyskanych z tej implementacji interfejsu SSLSocketFactory .
statusRefreshInterval[1]	int	• **60000** • Dowolna dodatnia liczba całkowita	Odstęp czasu (w milisekundach) między operacjami odświeżania długotrwałych transakcji, które wykrywają, kiedy subskrybent traci połączenie z menedżerem kolejek. Ta właściwość ma zastosowanie tylko wtedy, gdy SUBSTORE ma wartość QUEUE.
subscriptionStore[1]	Łańcuch	• **BROKER** • MIGRATE • QUEUE	Określa, gdzie klasy WebSphere MQ classes for JMS przechowują dane trwałe dotyczące aktywnych subskrypcji.
targetClientZgodne	wartość boolowska	• **Prawda** • Fałsz	Określa, czy komunikat odpowiedzi wysyłany do kolejki identyfikowanej przez pole nagłówka JMSReplyTo komunikatu przychodzącego ma nagłówek MQRFH2 tylko wtedy, gdy komunikat przychodzący ma nagłówek MQRFH2 .

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
temporaryModel	Łańcuch	• **SYSTEM.DEFAULT.MODEL.QUEUE** • SYSTEM.JMS.TEMPQ.MODEL • Dowolny łańcuch	Nazwa kolejki modelowej, na podstawie której tworzone są tymczasowe kolejki JMS. Użyj SYSTEM.DEFAULT.MODEL.QUEUE , jeśli spełnione są oba poniższe warunki: • Aplikacja używa kolejki tymczasowej, która będzie akceptować komunikaty nietrwałe. • Tylko jedna aplikacja utworzy w menedżerze kolejek kolejkę tymczasową, na którą wskazuje ConnectionFactory w danym momencie. Należy zauważyć, że SYSTEM.DEFAULT.MODEL.QUEUE może być jednocześnie otwierana tylko przez jedną aplikację. Użyj SYSTEM.JMS.TEMPQ.MODEL. w następujących sytuacjach: • Gdy aplikacja używa kolejki tymczasowej, która będzie akceptować komunikaty trwałe. • Jeśli wiele aplikacji może nawiązać połączenie z menedżerem kolejek, który wskazuje fabryka połączeń ConnectionFactory , a te aplikacje muszą jednocześnie tworzyć kolejki tymczasowe. Zdefiniuj nową kolejkę modelową z atrybutem DEFPSIST ustawionym na YES i atrybutem DEFSOPT ustawionym na SHARED w następującej sytuacji: • Jeśli aplikacja używa kolejki tymczasowej, która będzie akceptować nietrwałe komunikaty, a wiele aplikacji połączy się z menedżerem kolejek, do którego wskazuje fabryka połączeń ConnectionFactory , a te aplikacje będą musiały jednocześnie tworzyć kolejki tymczasowe. Podczas tworzenia nowej kolejki modelowej należy ustawić właściwość temporaryModel na nazwę nowej kolejki modelowej.

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
tempQPrefix	Łańcuch	• **"" (pusty łańcuch)** • Przedrostek, którego można użyć do utworzenia nazwy kolejki dynamicznej IBM WebSphere MQ . Reguły tworzenia przedrostka są takie same, jak reguły tworzenia treści pola *DynamicQName* w deskryptorze obiektu IBM WebSphere MQ o strukturze MQOD, ale ostatni niepusty znak musi być znakiem gwiazdki (*). Jeśli wartością właściwości jest pusty łańcuch, klasy produktu WebSphere MQ dla usługi JMS używają wartości AMQ.* podczas tworzenia kolejki dynamicznej.	Przedrostek używany do tworzenia nazwy kolejki dynamicznej IBM WebSphere MQ .
TEMPTOPICPREFIX	Łańcuch	Dowolny niepusty łańcuch składający się tylko z poprawnych znaków dla łańcucha tematu IBM WebSphere MQ	Podczas tworzenia tematów tymczasowych usługa JMS generuje łańcuch tematu w postaci "TEMP/*TEMPTOPICPREFIX*/*unikalny_identyfikator*" lub, jeśli ta właściwość ma wartość domyślną, tylko "TEMP/*unikalny_identyfikator*". Określenie niepustego parametru TEMPTOPICPREFIX umożliwia definiowanie konkretnych kolejek modelowych w celu tworzenia kolejek zarządzanych dla subskrybentów tematów tymczasowych utworzonych w ramach tego połączenia.
transportType	Łańcuch	• **client** • POWIĄZANIA • BINDINGS_THEN_CLIENT (POWIĄZANIA, NASTĘPNIE KLIENT)	Określa, czy połączenie z menedżerem kolejek używa trybu klienta, czy trybu powiązań. Jeśli zostanie podana wartość BINDINGS_THEN_CLIENT, adapter zasobów najpierw próbuje nawiązać połączenie w trybie powiązań. Jeśli ta próba nawiązania połączenia nie powiedzie się, adapter zasobów podejmie próbę nawiązania połączenia w trybie klienta.
Nazwa użytkownika	Łańcuch	• **Null** • Nazwa użytkownika	Domyślna nazwa użytkownika, która ma być używana podczas tworzenia połączenia z menedżerem kolejek.

Tabela 98. Właściwości obiektu ConnectionFactory (kontynuacja)			
Nazwa właściwości	**Typ**	**Poprawne wartości (wartość domyślna pogrubiona)**	**Opis**
wildcardFormat	int	• CHAR-rozpoznaje tylko znaki wieloznaczne używane w brokerze w wersji 1 • TOPIC-rozpoznaje tylko znaki wieloznaczne na poziomie tematu, używane w brokerze w wersji 2	Która wersja składni znaku wieloznacznego ma być używana.

Uwagi:

1. Ta właściwość może być używana z wersją 7.0 produktu IBM WebSphere MQ classes for JMS , ale nie ma wpływu na aplikację połączoną z menedżerem kolejek w wersji 7.0 , chyba że właściwość providerVersion jest ustawiona na numer wersji mniejszy niż 7.
2. Ważne informacje na temat używania wymaganej właściwości sslFipszawiera sekcja "Ograniczenia adaptera zasobów produktu IBM WebSphere MQ" na stronie 792.
3. Informacje na temat konfigurowania adaptera zasobów w taki sposób, aby mógł on znaleźć wyjście, zawiera sekcja "Konfigurowanie produktu IBM WebSphere MQ classes for JMS do korzystania z wyjść kanału" na stronie 939.

W poniższym przykładzie przedstawiono typowy zestaw właściwości obiektu ConnectionFactory :

```
channel:        SYSTEM.DEF.SVRCONN
hostName:       192.168.0.42
port:           1414
queueManager:   ExampleQM
transportType:  CLIENT
```

Tabela 99 na stronie 783 zawiera listę właściwości wspólnych dla obiektu kolejki i obiektu tematu.

Tabela 99. Właściwości wspólne dla obiektu kolejki i obiektu tematu			
Nazwa właściwości	**Typ**	**Poprawne wartości (wartość domyślna pogrubiona)**	**Opis**
CCSID	Łańcuch	• **1208** • Identyfikator kodowanego zestawu znaków obsługiwany przez wirtualną maszynę języka Java (JVM)	Identyfikator kodowanego zestawu znaków dla miejsca docelowego.

Tabela 99. Właściwości wspólne dla obiektu kolejki i obiektu tematu (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
kodowanie	Łańcuch	• **native** • Łańcuch składający się z trzech znaków: – Pierwszy znak określa reprezentację binarnych liczb całkowitych: - *N* oznacza normalne kodowanie. - *R* oznacza kodowanie odwrotne. – Drugi znak określa reprezentację upakowanych dziesiętnych liczb całkowitych: - *N* oznacza normalne kodowanie. - *R* oznacza kodowanie odwrotne. – Trzeci znak określa reprezentację liczb zmiennopozycyjnych: - *N* oznacza standardowe kodowanie IEEE. - *R* oznacza odwrotne kodowanie IEEE. - *3* oznacza kodowanie zSeries . NATIVE jest odpowiednikiem łańcucha NNN.	Reprezentacja binarnych liczb całkowitych, upakowanych liczb dziesiętnych i liczb zmiennopozycyjnych dla miejsca docelowego.
Utrata ważności	Łańcuch	• **APP** -czas utraty ważności komunikatu jest określany przez producenta komunikatów. • UNLIM-Komunikat nigdy nie traci ważności. • 0-komunikat nigdy nie traci ważności. • Dodatnia liczba całkowita reprezentująca czas utraty ważności komunikatu w milisekundach.	Czas utraty ważności komunikatu wysłanego do miejsca docelowego.
FAILIFQUIESCE	Łańcuch	• **Prawda** • Fałsz	Określa, czy próba uzyskania dostępu do miejsca docelowego nie powiedzie się, jeśli menedżer kolejek jest w stanie wyciszania.

Tabela 99. Właściwości wspólne dla obiektu kolejki i obiektu tematu (kontynuacja)

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
trwałość	Łańcuch	• **APP** -trwałość komunikatu jest określana przez producenta komunikatów. • QDEF-trwałość komunikatu jest określana przez atrybut *DefPersistence* kolejki produktu WebSphere MQ . • PERS-komunikat jest trwały. • NON-Komunikat jest nietrwały. • HIGH-Trwałość komunikatu jest określana przez atrybut *NonPersistentMessageClass* kolejki produktu WebSphere MQ zgodnie z wyjaśnieniem w sekcji "Trwałe komunikaty JMS" na stronie 931.	Trwałość komunikatu wysłanego do miejsca docelowego.
priorytet	Łańcuch	• **APP** -priorytet komunikatu jest określany przez producenta komunikatu. • QDEF-priorytet komunikatu jest określany przez atrybut *DefPriority* kolejki IBM WebSphere MQ . • Liczba całkowita z zakresu od 0, najniższy priorytet, do 9, najwyższy priorytet.	Priorytet komunikatu wysyłanego do miejsca docelowego.
PUTASYNCALLOWED	Łańcuch	• QUEUE-określa, czy asynchroniczne operacje umieszczania są dozwolone, odwołując się do definicji kolejki. • TOPIC-Określenie, czy asynchroniczne operacje umieszczania są dozwolone, poprzez odwołanie się do definicji tematu. • DESTINATION-określa, czy asynchroniczne operacje umieszczania są dozwolone, odwołując się do definicji kolejki lub tematu. • DISABLED-asynchroniczne operacje umieszczania nie są dozwolone. • ENABLED-asynchroniczne operacje put są dozwolone.	Określa, czy producenci komunikatów mogą używać asynchronicznych operacji umieszczania w celu wysyłania komunikatów do tego miejsca docelowego.

Tabela 99. Właściwości wspólne dla obiektu kolejki i obiektu tematu (kontynuacja)			
Nazwa właściwości	**Typ**	**Poprawne wartości (wartość domyślna pogrubiona)**	**Opis**
READAHEADALLOWED	int	• **DESTINATION** -określa, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji kolejki lub tematu. • DISABLED-odczyt z wyprzedzeniem jest niedozwolony. • ENABLED-odczyt z wyprzedzeniem jest dozwolony. • QUEUE-określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji kolejki. • TOPIC-Określenie, czy odczyt z wyprzedzeniem jest dozwolony, poprzez odwołanie się do definicji tematu.	Określa, czy konsumenty komunikatów i przeglądarki kolejek mogą używać odczytu z wyprzedzeniem do pobierania nietrwałych komunikatów z miejsca docelowego do buforu wewnętrznego przed ich odebraniem.
receiveCCSID	int	• **0** -użyj maszyny JVM `Charset.defaultCharset` • 1208- UTF-8 • Obsługiwany identyfikator kodowanego zestawu znaków	Właściwość miejsca docelowego, która ustawia docelowy identyfikator CCSID dla konwersji komunikatów menedżera kolejek. Wartość jest ignorowana, chyba że parametr **`receiveConversion`** ma wartość `QMGR`
receiveConversion	Łańcuch	• **KOMUNIKAT KLIENTA** • QMGR	Właściwość miejsca docelowego, która określa, czy konwersja danych będzie wykonywana przez menedżer kolejek.
targetClient	Łańcuch	• **JMS** -celem komunikatu jest aplikacja JMS. • MQ -celem komunikatu jest aplikacja IBM WebSphere MQ inna niż JMS.	Określa, czy miejscem docelowym komunikatu wysyłanego do miejsca docelowego jest aplikacja JMS. Komunikat z miejscem docelowym, który jest aplikacją JMS, zawiera nagłówek MQRFH2 .

Tabela 100 na stronie 786 zawiera listę właściwości, które są specyficzne dla obiektu kolejki.

Tabela 100. Właściwości specyficzne dla obiektu kolejki			
Nazwa właściwości	**Typ**	**Poprawne wartości (wartość domyślna pogrubiona)**	**Opis**
baseQueueManagerName	Łańcuch	• **"" (pusty łańcuch)** • Nazwa menedżera kolejek	Nazwa menedżera kolejek, który jest właścicielem bazowej kolejki produktu IBM WebSphere MQ .
Nazwa baseQueue	Łańcuch	• **"" (pusty łańcuch)** • Nazwa kolejki	Nazwa bazowej kolejki produktu IBM WebSphere MQ .

Tabela 101 na stronie 787 zawiera listę właściwości, które są specyficzne dla obiektu Topic.

Tabela 101. Właściwości specyficzne dla obiektu Topic

Nazwa właściwości	Typ	Poprawne wartości (wartość domyślna pogrubiona)	Opis
baseTopicNazwa	Łańcuch	• **"" (pusty łańcuch)** • Nazwa tematu	Nazwa tematu bazowego.
brokerCCDurSubQueue[1]	Łańcuch	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Nazwa kolejki	Nazwa kolejki, z której konsument połączenia odbiera komunikaty subskrypcji trwałej.
brokerDurSubQueue[1]	Łańcuch	• **SYSTEM.JMS.D.SUBSCRIBER.QUEUE** • Nazwa kolejki	Nazwa kolejki, z której trwały subskrybent tematu odbiera komunikaty. Więcej informacji na ten temat zawiera opis właściwości BROKEDURRSUBQ w dokumentacji programu WebSphere MQ Explorer.
Kolejka brokerPubQueue[1]	Łańcuch	• **Nie ustawiono** • Nazwa kolejki	Nazwa kolejki, do której wysyłane są opublikowane komunikaty (kolejka strumienia). Wartość tej właściwości nadpisuje wartość właściwości kolejki brokerPubobiektu ConnectionFactory . Jeśli jednak wartość tej właściwości nie zostanie ustawiona, zostanie użyta wartość właściwości kolejki brokerPubobiektu ConnectionFactory .
brokerPubQueueManager[1]	Łańcuch	• **"" (pusty łańcuch)** • Nazwa menedżera kolejek	Nazwa menedżera kolejek będącego właścicielem kolejki, do której są wysyłane komunikaty opublikowane w temacie.
Wersja brokera[1]	Łańcuch	• **Nie ustawiono** • 1 • 2	Wersja używanego brokera. Wartość tej właściwości nadpisuje wartość właściwości brokerVersion obiektu ConnectionFactory . Jeśli jednak wartość tej właściwości nie zostanie ustawiona, zostanie użyta wartość właściwości brokerVersion obiektu ConnectionFactory .

Tabela 101. Właściwości specyficzne dla obiektu Topic (kontynuacja)			
Nazwa właściwości	**Typ**	**Poprawne wartości (wartość domyślna pogrubiona)**	**Opis**
Uwaga: 1. Ta właściwość może być używana z wersją 7.0 produktu IBM WebSphere MQ classes for JMS , ale nie ma wpływu na aplikację połączoną z menedżerem kolejek w wersji 7.0 , chyba że właściwość providerVersion obiektu ConnectionFactory ma numer wersji mniejszy niż 7.			

W poniższym przykładzie przedstawiono zestaw właściwości obiektu Queue:

```
expiry:                 UNLIM
persistence:            QDEF
baseQueueManagerName:   ExampleQM
baseQueueName:          SYSTEM.JMS.TEMPQ.MODEL
```

W poniższym przykładzie przedstawiono zestaw właściwości obiektu Topic:

```
expiry:                 UNLIM
persistence:            NON
baseTopicName:          myTestTopic
```

Zadania pokrewne

Określenie, że w czasie wykonywania na kliencie MQI będą używane tylko CipherSpecs z certyfikatem FIPS.

Odsyłacze pokrewne

Standardy FIPS (Federal Information Processing Standards) dla systemów UNIX, Linux i Windows

V 7.5.0.9 *Konfigurowanie właściwości targetClientMatching dla specyfikacji aktywowania*

Istnieje możliwość skonfigurowania właściwości **targetClientMatching** dla specyfikacji aktywowania w taki sposób, aby nagłówek MQRFH2 był dołączany do komunikatów odpowiedzi, gdy komunikaty żądania nie zawierają nagłówka MQRFH2 . Oznacza to, że wszystkie właściwości komunikatu, które aplikacja definiuje w komunikacie odpowiedzi, są uwzględniane podczas wysyłania komunikatu.

O tym zadaniu

Jeśli aplikacja komponentu bean sterowanego komunikatami zużywa komunikaty, które nie zawierają nagłówka MQRFH2 , za pośrednictwem specyfikacji aktywowania adaptera zasobów JCA produktu IBM WebSphere MQ , a następnie wysyła komunikaty odpowiedzi do miejsca docelowego JMS utworzonego w polu `JMSReplyTo` komunikatu żądania, komunikaty odpowiedzi muszą zawierać nagłówek MQRFH2 , nawet jeśli komunikaty żądania nie zawierają żadnych właściwości komunikatu, które aplikacja zdefiniowała w komunikacie odpowiedzi, są tracone.

Właściwość **targetClientMatching** definiuje, czy komunikat odpowiedzi, wysyłany do kolejki identyfikowanej przez pole nagłówka JMSReplyTo komunikatu przychodzącego, ma nagłówek MQRFH2 tylko wtedy, gdy komunikat przychodzący ma nagłówek MQRFH2 . Tę właściwość można skonfigurować dla specyfikacji aktywowania, zarówno w produkcie WebSphere Application Server tradycyjnym, jak i w produkcie WebSphere Application Server Liberty.

Jeśli wartość właściwości **targetClientMatching** zostanie ustawiona na `false`, nagłówek MQRFH2 może zostać umieszczony w komunikacie odpowiedzi wysłanym do miejsca docelowego JMS utworzonego z nagłówka JMSReplyTo komunikatu przychodzącego żądania, który nie zawiera MQRFH2. Dzieje się tak dlatego, że właściwość **targetClient** w miejscu docelowym JMS jest ustawiona na wartość `0`, co oznacza, że komunikaty zawierają nagłówek MQRFH2 . Obecność nagłówka MQRFH2 w komunikacie wychodzącym pozwala na przechowywanie zdefiniowanych przez użytkownika właściwości komunikatu w komunikacie, gdy jest on wysyłany do kolejki produktu IBM WebSphere MQ .

Jeśli właściwość **targetClientMatching** jest ustawiona na wartość `true` , a komunikat żądania nie zawiera nagłówka MQRFH2 , to nagłówek MQRFH2 nie jest zawarty w komunikacie odpowiedzi.

Procedura

- W produkcie WebSphere Application Server tradycyjnym użyj konsoli administracyjnej, aby zdefiniować właściwość **targetClientMatching** jako właściwość niestandardową w specyfikacji aktywowania produktu IBM WebSphere MQ :
  a) W panelu nawigacyjnym kliknij opcję **Resources-> JMS-> Activation specifications**(Zasoby-> JMS-> Specyfikacje aktywowania).
  b) Wybierz nazwę specyfikacji aktywowania, która ma być wyświetlana lub zmieniona.
  c) Kliknij opcję **Właściwości niestandardowe-> Nowy** , a następnie wprowadź szczegóły nowej właściwości niestandardowej.
  Ustaw nazwę właściwości na `targetClientMatching`, typ `java.lang.Boolean` i wartość na `false`.
- W produkcie WebSphere Application Server Liberty należy określić właściwość **targetClientMatching** w definicji specyfikacji aktywowania w ramach produktu `server.xml`.
  Na przykład:

```
<jmsActivationSpec id="SimpleMDBApplication/SimpleEchoMDB/SimpleEchoMDB">
<properties.wmqJms destinationRef="MDBRequestQ"
queueManager="MY_QMGR" transportType="BINDINGS" targetClientMatching="false"/>
<authData password="********" user="tom"/>
</jmsActivationSpec>
```

Pojęcia pokrewne

“Tworzenie miejsc docelowych w aplikacji JMS” na stronie 902
Zamiast pobierania miejsc docelowych jako administrowanych obiektów z przestrzeni nazw JNDI (Java Naming and Directory Interface) aplikacja JMS może używać sesji w celu dynamicznego tworzenia miejsc docelowych w czasie wykonywania. Aplikacja może użyć identyfikatora URI (Uniform Resource Identifier) w celu zidentyfikowania kolejki WebSphere MQ lub tematu oraz, opcjonalnie, w celu określenia jednej lub większej liczby właściwości obiektu Queue lub Topic.

“Konfigurowanie adaptera zasobów na potrzeby komunikacji wychodzącej” na stronie 771
Aby skonfigurować komunikację wychodzącą, należy zdefiniować właściwości obiektu ConnectionFactory i administrowanego obiektu docelowego.

ASF i tryb bez ASF
Tryb ASF (Application Server Facilities) jest domyślną metodą, za pomocą której usługa nasłuchiwania komunikatów w produkcie WebSphere Application Server przetwarza komunikaty.

Usługa nasłuchiwania komunikatów ma dwa tryby działania: Application Server Facilities (ASF) i inne niż Application Server Facilities (inne niż ASF):

- Tryb ASF zapewnia współbieżność i obsługę transakcyjną dla aplikacji. W przypadku komponentów bean komunikatów publikowania/subskrybowania, tryb ASF zapewnia lepszą przepustowość i współbieżność, ponieważ w trybie bez ASF nasłuchiwanie jest jednowątkowe.
- Tryb bez ASF jest przeznaczony głównie do użytku z dostawcami przesyłania komunikatów innych firm, które nie obsługują ASF, co jest opcjonalnym rozszerzeniem specyfikacji JMS. Tryb bez ASF jest również transakcyjny, ale ponieważ długość ścieżki jest krótsza niż w przypadku trybu ASF, zwykle zapewnia lepszą wydajność.

Aby włączyć tryb bez ASF operacji dla wszystkich obiektów nasłuchiwania komponentów bean sterowanych komunikatami na serwerze aplikacji, należy ustawić tę właściwość na wartość niezerową.

Uwaga:

Tryb bez ASF nie może być wybrany w systemach z/OS , dlatego w tym przypadku nie można ustawić wartości niezerowej dla tej właściwości.

Przetwarzanie komunikatów w trybie ASF

W trybie ASF sesje i wątki serwera są przydzielane do pracy tylko wtedy, gdy zostanie wykryty komunikat odpowiedni dla komponentu bean sterowanego komunikatami (message-driven bean-MDB). Liczba wątków, które komponent MDB może przetwarzać współbieżnie, jest określana na podstawie wartości właściwości **Maximum Sessions** dla portu nasłuchiwania lub specyfikacji aktywowania.

Przetwarzanie komunikatów w trybie bez ASF

W trybie bez ASF wątki są aktywne od momentu uruchomienia portu nasłuchiwania lub specyfikacji aktywowania. Liczba aktywnych wątków jest podyktowana wartością określoną dla właściwości **Maximum Sessions** . Liczba wątków określonych we właściwości **Maximum Sessions** jest aktywna, niezależnie od liczby komunikatów, które są dostępne do przetworzenia. Każdy aktywny wątek jest pojedynczym fizycznym połączeniem sieciowym.

Produkt IBM WebSphere MQ w wersji 7.0 lub nowszej umożliwia dostęp do dziesięciu wątków współużytkującej pojedyncze fizyczne połączenie sieciowe.

Pojęcia pokrewne

IBM WebSphere MQ classes for JMS Application Server Facilities
W tym temacie opisano, w jaki sposób klasy WebSphere MQ classes for JMS implementują klasę ConnectionConsumer oraz zaawansowaną funkcjonalność w klasie Session. Podsumowuje on również funkcję puli sesji serwera.

Zadania pokrewne

Konfigurowanie specyfikacji aktywowania dla trybu bez ASF
Specyfikacje aktywowania to standaryzowany sposób zarządzania i konfigurowania relacji między komponentem bean sterowanym komunikatami (MDB) uruchomionym w produkcie WebSphere Application Server a miejscem docelowym w produkcie IBM WebSphere MQ. W tym zadaniu wyjaśniono, w jaki sposób można skonfigurować produkt WebSphere Application Server w taki sposób, aby używały trybu bez ASF do przetwarzania komunikatów.

Informacje pokrewne

Przetwarzanie komunikatów w trybie ASF i trybie bez ASF

Konfigurowanie specyfikacji aktywowania dla trybu bez ASF
Specyfikacje aktywowania to standaryzowany sposób zarządzania i konfigurowania relacji między komponentem bean sterowanym komunikatami (MDB) uruchomionym w produkcie WebSphere Application Server a miejscem docelowym w produkcie IBM WebSphere MQ. W tym zadaniu wyjaśniono, w jaki sposób można skonfigurować produkt WebSphere Application Server w taki sposób, aby używały trybu bez ASF do przetwarzania komunikatów.

Zanim rozpoczniesz

Sposób definiowania właściwości specyfikacji aktywowania zależy od interfejsów administracyjnych udostępnianych przez serwer aplikacji. W przypadku tego zadania założono, że jako serwer aplikacji używany jest produkt WebSphere Application Server w wersji 7 lub nowszej, a jako dostawca przesyłania komunikatów produkt IBM WebSphere MQ .

Uwaga:

Tryb bez ASF nie może być wybrany w systemach z/OS .

O tym zadaniu

Właściwości specyfikacji aktywowania określają, w jaki sposób komponent bean sterowany komunikatami (Message Drive Bean-MDB) odbiera komunikaty JMS z kolejki produktu IBM WebSphere MQ . Aby skonfigurować tryb bez ASF, należy zdefiniować właściwości jednej lub większej liczby specyfikacji aktywowania.

Istnieje kilka konfiguracji produktu IBM WebSphere MQ , których można użyć w trybie bez ASF. W następujących konfiguracjach każdy wątek korzysta z osobnego połączenia z siecią fizyczną:

- Menedżer kolejek produktu IBM WebSphere MQ w wersji 7.x , przy użyciu fabryki połączeń, która ma właściwość Wersja dostawcy ustawioną na wartość 6.
- Menedżer kolejek produktu IBM WebSphere MQ w wersji 7.x , przy użyciu fabryki połączeń, dla której właściwość wersji dostawcy jest ustawiona na wartość 7 lub nieokreślona, łącząc kanał IBM WebSphere MQ z parametrem **SHARECNV** (współużytkowanie konwersacji) ustawionym na wartość 0.

Aby skonfigurować inne niż ASF, należy ustawić właściwość ActivationSpec **NON.ASF.RECEIVE.TIMEOUT** na dodatnią liczbę całkowitą, która wskazuje, że jest używana dostawa bez ASF. Wartością jest czas (w milisekundach), przez który żądanie pobrania oczekuje na komunikaty, które mogły jeszcze nie zostać odebrane (wywołanie get z oczekiwaniem). Wartość domyślna 0 oznacza, że jest używana dostarczanie ASF. Więcej szczegółów na ten temat zawiera sekcja **Właściwości niestandardowe usługi nasłuchiwania komunikatów**.

Ten parametr jest poprawny tylko wtedy, gdy aplikacja jest uruchomiona w systemie WebSphere Application Server w wersji 7 lub nowszej.

Procedura

1. Uruchom konsolę administracyjną serwera WebSphere Application Server .
2. Wyświetl stronę ustawień usługi nasłuchiwania:
   a) W panelu nawigacyjnym wybierz opcję **Serwery > Typy serwerów > Serwery aplikacji WebSphere**.
   b) W panelu treści kliknij nazwę serwera aplikacji.
   c) W sekcji **Komunikacja**kliknij opcję **Przesyłanie komunikatów > Usługa nasłuchiwania komunikatów**.
3. Ustaw właściwość niestandardową **NON.ASF.RECEIVE.TIMEOUT** jako właściwości niestandardowe usługi nasłuchiwania komunikatów.
   a) Kliknij opcję **Właściwości niestandardowe**.
   b) Kliknij opcję **Nowy**.
   c) W polu **Nazwa** wprowadź nazwę właściwości **NON.ASF.RECEIVE.TIMEOUT**.
   d) Wprowadź wymagane wartości w polu **Wartość** .
   e) Kliknij przycisk **OK**.
4. Zapisz zmiany w konfiguracji głównej.
5. Aby aktywować zmienioną konfigurację, zatrzymaj, a następnie zrestartuj serwer aplikacji.

Wyniki

Właściwości usługi nasłuchiwania komunikatów dla produktu WebSphere Application Server zostały skonfigurowane w taki sposób, aby można było używać trybu bez ASF.

Uwaga: W przypadku korzystania z trybu bez ASF należy zapewnić odpowiednią ilość czasu na zakończenie przetwarzania, zanim zostanie osiągnięty łączny limit czasu życia transakcji, w celu uniknięcia niepożądanych przekroczeń limitu czasu transakcji. Więcej informacji na ten temat zawiera sekcja **NON.ASF.RECEIVE.TIMEOUT** w dokumentacji produktu WebSphere Application Server .

Pojęcia pokrewne

“ASF i tryb bez ASF” na stronie 789
Tryb ASF (Application Server Facilities) jest domyślną metodą, za pomocą której usługa nasłuchiwania komunikatów w produkcie WebSphere Application Server przetwarza komunikaty.

Konfigurowanie adaptera zasobów na potrzeby komunikacji przychodzącej
Aby skonfigurować komunikację przychodzącą, należy zdefiniować właściwości jednego lub większej liczby obiektów ActivationSpec .

Informacje pokrewne

Komponenty bean sterowane komunikatami

Usługa nasłuchiwania komunikatów

Przetwarzanie komunikatów w trybie ASF i trybie bez ASF

Sposób przetwarzania komunikatów w trybie bez ASF

Ograniczenia adaptera zasobów produktu IBM WebSphere MQ

Jeśli używany jest adapter zasobów IBM WebSphere MQ , niektóre funkcje produktu IBM WebSphere MQ są niedostępne lub ograniczone.

Adapter zasobów produktu IBM WebSphere MQ ma następujące ograniczenia:

- Adapter zasobów produktu IBM WebSphere MQ jest obsługiwany na wszystkich platformach IBM WebSphere MQ z wyjątkiem systemu z/OS.
- Adapter zasobów produktu IBM WebSphere MQ nie obsługuje połączeń w czasie rzeczywistym z brokerem. Obsługuje on tylko połączenia z menedżerem kolejek produktu IBM WebSphere MQ w trybie klienta lub powiązania.
- Adapter zasobów produktu IBM WebSphere MQ nie obsługuje programów obsługi wyjścia kanałów, które są napisane w językach innych niż Java.
- Gdy serwer aplikacji jest uruchomiony, wartość wymaganej właściwości sslFipsmusi mieć wartość true dla wszystkich zasobów JCA lub wartości false dla wszystkich zasobów JCA. Jest to wymaganie, nawet jeśli zasoby JCA nie są używane współbieżnie. Jeśli wymagana właściwość sslFipsma inne wartości dla różnych zasobów JCA, produkt IBM WebSphere MQ wydaje kod przyczyny MQRC_UNSUPPORTED_CIPHER_SUITE, nawet jeśli połączenie SSL nie jest używane.
- Nie można określić więcej niż jednego magazynu kluczy dla serwera aplikacji. Jeśli połączenia są nawiązane do więcej niż jednego menedżera kolejek, wszystkie połączenia muszą korzystać z tego samego magazynu kluczy. To ograniczenie nie ma zastosowania do serwera WebSphere Application Server.
- Jeśli używana jest tabela definicji kanału klienta (CCDT) z więcej niż jedną odpowiednią definicją kanału połączenia klienta, w przypadku awarii adapter zasobów może wybrać inną definicję kanału i w związku z tym inny menedżer kolejek z tabeli definicji kanału klienta, co powodowałoby problemy podczas odtwarzania transakcji. Adapter zasobów nie podejmuje żadnych działań w celu uniknięcia użycia takiej konfiguracji, a użytkownik jest odpowiedzialny za unikanie konfiguracji, które mogą powodować problemy związane z odtwarzaniem transakcji.
- Funkcje ponawiania połączenia wprowadzone w produkcie IBM WebSphere MQ Version 7.0.1 nie są obsługiwane w przypadku połączeń wychodzących podczas pracy w kontenerze JEE (EJB/Servlet). Ponowienie połączenia nie jest obsługiwane w ogóle dla wychodzących JMS, gdy adapter jest używany w kontekście kontenera JEE, bez względu na konfigurację transakcji lub w przypadku użycia nietransakcyjnego.

Zadania pokrewne

Określanie, że w czasie wykonywania w kliencie MQI są używane tylko specyfikacje CipherSpecs z certyfikatem FIPS

Odsyłacze pokrewne

Standardy FIPS (Federal Information Processing Standards) dla systemów UNIX, Linux i Windows

Konfigurowanie po instalacji dla klas produktu WebSphere MQ dla aplikacji JMS

Ten temat zawiera informacje o tym, jakie uprawnienia potrzebne są do klas WebSphere MQ dla aplikacji JMS w celu uzyskania dostępu do zasobów menedżera kolejek. Przedstawiono również tryby połączenia i opisano sposób konfigurowania menedżera kolejek w taki sposób, aby aplikacje mogły łączyć się w trybie klienta.

Pamiętaj, aby sprawdzić plik readme produktu WebSphere MQ . Może on zawierać informacje, które zastępują informacje zawarte w tym temacie.

Obiekty używane przez usługę JMS, które wymagają autoryzacji dla użytkowników nieuprzywilejowanych

Użytkownicy bez uprawnień uprzywilejowani potrzebują autoryzacji dostępu do kolejek używanych przez usługę JMS. Każda aplikacja JMS wymaga autoryzacji do menedżera kolejek, z którym działa.

Szczegółowe informacje na temat kontroli dostępu w produkcie IBM WebSphere MQmożna znaleźć w sekcji Konfigurowanie zabezpieczeń w systemach Windowsi UNIX and Linux .

Klasy produktu WebSphere MQ dla aplikacji JMS wymagają uprawnień `connect` i `inq` do menedżera kolejek. Odpowiednie autoryzacje można ustawić za pomocą komendy sterującej **setmqaut** , na przykład:

```
setmqaut -m QM1 -t qmgr -g jmsappsgroup +connect +inq
```

W przypadku domeny punkt z punktem wymagane są następujące uprawnienia:

- Kolejki, które są używane przez obiekty MessageProducer , wymagają uprawnień `put` .
- Kolejki używane przez obiekty MessageConsumer i QueueBrowser wymagają uprawnień `get`, `inq`i `browse` .
- Metoda QueueSession.createTemporaryQueue () wymaga dostępu do kolejki modelowej określonej przez właściwość TEMPMODEL obiektu fabryki QueueConnection. Domyślną kolejką modelową jest SYSTEM.TEMP.MODEL.QUEUE.

Jeśli dowolna z tych kolejek jest kolejkami aliasami, ich kolejki docelowe wymagają uprawnień do sprawdzania uprawnień. Jeśli kolejka docelowa jest kolejką klastra, wymaga to również uprawnienia do przeglądania.

W przypadku domeny publikowania/subskrypcji używane są następujące kolejki, jeśli klasy produktu WebSphere MQ dla usługi JMS łączą się z menedżerem kolejek produktu IBM WebSphere MQ w trybie migracji dostawcy przesyłania komunikatów produktu IBM WebSphere MQ :

- SYSTEM.JMS.ADMIN.QUEUE
- SYSTEM.JMS.REPORT.QUEUE
- SYSTEM.JMS.MODEL.QUEUE
- SYSTEM.JMS.PS.STATUS.QUEUE
- SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
- SYSTEM.JMS.D.SUBSCRIBER.QUEUE
- SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
- SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
- SYSTEM.BROKER.CONTROL.QUEUE

Więcej informacji na temat trybu migracji dostawcy przesyłania komunikatów produktu IBM WebSphere MQ można znaleźć w sekcji Kiedy używać produktu **PROVIDERVERSION** .

Dodatkowo, jeśli klasy produktu WebSphere MQ dla usługi JMS nawiązują połączenie z menedżerem kolejek w tym trybie, każda aplikacja publikujący komunikaty potrzebuje dostępu do kolejki strumienia określonej przez fabrykę lub obiekt tematu TopicConnection. Domyślnie ta kolejka to SYSTEM.BROKER.DEFAULT.STREAM.

Jeśli używany jest dostawca przesyłania komunikatów ConnectionConsumer, IBM WebSphere MQ Resource Adapter lub WebSphere Application Server IBM WebSphere MQ , może być potrzebna dodatkowa autoryzacja.

Kolejki, które mają być odczytane przez parametr ConnectionConsumer , muszą mieć uprawnienia `get` , `inq`i `browse` . Systemowa kolejka niedostarczonych komunikatów oraz kolejka wycofanych komunikatów lub kolejka raportów używana przez obiekt ConnectionConsumer muszą mieć uprawnienia `put` i `passall` .

Gdy aplikacja używa trybu normalnego dostawcy przesyłania komunikatów produktu WebSphere MQ do przesyłania komunikatów w trybie publikowania/subskrypcji, aplikacja korzysta ze zintegrowanych funkcji

publikowania/subskrypcji udostępnianej przez menedżer kolejek. Informacje na temat zabezpieczania tematów i kolejek, które są używane, zawiera sekcja Zabezpieczenia publikowania/subskrypcji .

Tryby połączenia dla klas produktu WebSphere MQ dla usługi JMS

Klasy produktu WebSphere MQ dla aplikacji JMS mogą łączyć się z menedżerem kolejek w trybie klienta lub powiązania. W trybie klienta klasy WebSphere MQ classes for JMS łączy się z menedżerem kolejek za pośrednictwem protokołu TCP/IP. W trybie powiązań klasy WebSphere MQ classes for JMS łączy się bezpośrednio z menedżerem kolejek przy użyciu interfejsu JNI (Java Native Interface).

Aplikacja działająca na serwerze WebSphere Application Server w systemie z/OS może łączyć się z menedżerem kolejek w powiązaniach lub trybie klienta, ale aplikacja działająca w dowolnym innym środowisku w systemie z/OS może łączyć się z menedżerem kolejek tylko w trybie powiązań. Aplikacja działająca na dowolnej innej platformie może połączyć się z menedżerem kolejek w powiązaniach lub trybie klienta.

Istnieje możliwość użycia bieżącej lub dowolnej wcześniejszej obsługiwanej wersji produktu WebSphere MQ classes for JMS z bieżącym menedżerem kolejek. Można też użyć bieżącej lub wcześniejszej obsługiwanej wersji menedżera kolejek z bieżącą wersją klas WebSphere MQ dla usługi JMS. Jeśli wymieszasz różne wersje, funkcja jest ograniczona do poziomu wcześniejszej wersji.

W poniższych sekcjach opisano szczegółowo każdy z trybów połączenia.

Tryb klienta

Aby nawiązać połączenie z menedżerem kolejek w trybie klienta, klasy WebSphere MQ dla aplikacji JMS mogą być uruchamiane w tym samym systemie, w którym jest uruchomiony menedżer kolejek, lub w innym systemie. W każdym przypadku klasy produktu WebSphere MQ dla usługi JMS łączy się z menedżerem kolejek za pośrednictwem protokołu TCP/IP.

Tryb powiązań

Aby nawiązać połączenie z menedżerem kolejek w trybie powiązań, klasy produktu WebSphere MQ dla aplikacji JMS muszą być uruchamiane w tym samym systemie, w którym działa menedżer kolejek.

Klasy produktu WebSphere MQ dla usługi JMS łączy się bezpośrednio z menedżerem kolejek przy użyciu interfejsu JNI (Java Native Interface). Aby można było używać transportu powiązań, klasy produktu WebSphere MQ dla usługi JMS muszą być uruchamiane w środowisku, które ma dostęp do bibliotek rodzimej interfejsu języka Java produktu WebSphere MQ . Dodatkowe informacje można znaleźć w sekcji “Konfigurowanie bibliotek Java Native Interface (JNI)” na stronie 743 .

Klasy produktu WebSphere MQ dla usługi JMS obsługują następujące wartości dla opcji *ConnectOption* :

- MQCNO_FASTPATH_BINDING
- MQCNO_STANDARD_BINDING
- MQCNO_SHARED_BINDING
- MQCNO_ISOLATED_BINDING
- MQCNO_SERIALIZE_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_QSG
- MQCNO_SERIALIZE_CONN_TAG_Q_MGR
- MQCNO_RESTRICT_CONN_TAG_Q_MGR

Aby zmienić opcje połączenia używane przez klasy produktu WebSphere MQ dla usługi JMS, należy zmodyfikować właściwość fabryki połączeń CONNOPT.

Więcej informacji na temat opcji połączenia zawiera sekcja “Nawiąże połączenie z menedżerem kolejek przy użyciu wywołania MQCONNX” na stronie 214

Aby można było korzystać z transportu powiązań, używane środowisko Java Runtime Environment musi obsługiwać identyfikator kodowanego zestawu znaków (Coded Character Set Identifier-CCSID) menedżera kolejek, z którym łączy się klasy WebSphere MQ dla JMS .

Szczegółowe informacje na temat sposobu określania identyfikatora CCSID obsługiwanego przez środowisko Java Runtime Environment można znaleźć w podręczniku WebSphere MQ FDC z identyfikatorem sondy 21 wygenerowanym podczas korzystania z klas WebSphere MQ V7 dla klas Java lub WebSphere MQ V7 dla usługi JMS .

Powiązania, a następnie tryb klienta

Jest to opcja domyślna. Podczas nawiązywania połączenia z menedżerem kolejek w tym trybie klasy produktu WebSphere MQ dla aplikacji JMS podejmą próbę nawiązania połączenia w trybie powiązań, który wymaga, aby menedżer kolejek rezydował na tym samym komputerze, co aplikacja. Jeśli połączenie nie powiedzie się, aplikacja podejmie próbę nawiązania połączenia w trybie klienta, umożliwiając menedżerowi kolejek rezydowanie lokalnie na tym samym komputerze co aplikacja lub zdalnie.

Konfigurowanie menedżera kolejek w taki sposób, aby klasy produktu WebSphere MQ dla aplikacji JMS mogły łączyć się w trybie klienta

Aby skonfigurować menedżer kolejek w taki sposób, aby klasy produktu WebSphere MQ dla aplikacji JMS mogły łączyć się w trybie klienta, należy utworzyć definicję kanału połączenia z serwerem i uruchomić proces nasłuchujący.

W systemie z/OSmusi być zainstalowana opcja Załącznik klienta.

Tworzenie definicji kanału połączenia z serwerem

Na wszystkich platformach można użyć komendy MQSC DEFINE CHANNEL, aby utworzyć definicję kanału połączenia z serwerem. Zapoznaj się z poniższym przykładem:

```
DEFINE CHANNEL(JAVA.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)
```

W systemie IBM imożna zamiast tego użyć komendy CL CRTMQMCHL, jak w następującym przykładzie:

```
CRTMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN)
                                TRPTYPE(*TCP)
                                MQMNAME(QMGRNAME)
```

W tej komendzie *QMGRNAME* jest nazwą menedżera kolejek.

Definicję kanału połączenia z serwerem można również utworzyć za pomocą programu IBM WebSphere MQ Explorer, który działa w systemach Linux i Windows, lub na panelach operacji i sterowania w systemie z/OS.

Nazwa kanału (JAVA.CHANNEL w poprzednich przykładach) musi być taki sam, jak nazwa kanału określona przez właściwość CHANNEL fabryki połączeń używanej przez aplikację do łączenia się z menedżerem kolejek. Wartością domyślną właściwości CHANNEL jest SYSTEM.DEF.SVRCONN.

Uruchamianie nasłuchiwania

Należy uruchomić program nasłuchujący dla menedżera kolejek, jeśli nie został on jeszcze uruchomiony.

Na wszystkich platformach można użyć komendy MQSC START LISTENER w celu uruchomienia programu nasłuchującego, ale z wyjątkiem systemu z/OS, należy najpierw utworzyć obiekt nasłuchiwania przy użyciu komendy MQSC DEFINE LISTENER. Zapoznaj się z poniższym przykładem:

```
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) PORT(1414)
START LISTENER(LISTENER.TCP)
```

W systemach UNIX, Linuxi Windows można również użyć komendy sterującej **runmqlsr** , aby uruchomić program nasłuchujący, tak jak w następującym przykładzie:

```
runmqlsr -t tcp -p 1414 -m QMgrName
```

W tej komendzie parametr *QMgrName* jest nazwą menedżera kolejek.

Program nasłuchujący można również uruchomić za pomocą programu WebSphere MQ Explorer, który działa w systemach Linux i Windows, lub na panelach operacji i sterowania w systemie z/OS.

Numer portu, na którym nasłuchuje nasłuchiwanie, musi być taki sam, jak numer portu określony przez właściwość PORT fabryki połączeń używanej przez aplikację do łączenia się z menedżerem kolejek. Wartością domyślną właściwości PORT jest 1414.

Test weryfikujący instalację punkt-punkt dla klas WebSphere MQ dla usługi JMS

Program IVT (point-to-point installation verification test) jest dostarczany z klasami WebSphere MQ dla usługi JMS. Program nawiązuje połączenie z menedżerem kolejek w trybie powiązań lub w trybie klienta i wysyła komunikat do kolejki o nazwie SYSTEM.DEFAULT.LOCAL.QUEUE, a następnie odbiera komunikat z kolejki. Program może tworzyć i konfigurować wszystkie obiekty, które są wymagane dynamicznie w czasie wykonywania, lub może używać interfejsu JNDI do pobierania obiektów administrowanych z usługi katalogowej.

Uruchom test weryfikujący instalację bez używania interfejsu JNDI jako pierwszego, ponieważ test jest samodzielny i nie wymaga użycia usługi katalogowej. Opis obiektów administrowanych zawiera sekcja “Typy obiektów JMS” na stronie 965.

Test weryfikujący instalację punkt-punkt bez użycia interfejsu JNDI

W tym teście program IVT tworzy i konfiguruje wszystkie obiekty, które są wymagane dynamicznie w czasie wykonywania i nie używa interfejsu JNDI.

Dostępny jest skrypt uruchamiający program IVT. Skrypt ma nazwę IVTRun w systemach UNIX and Linux i IVTRun.bat w systemie Windowsi znajduje się w podkatalogu `bin` katalogu instalacyjnego klas WebSphere MQ classes for JMS.

Aby uruchomić test w trybie powiązań, wprowadź następującą komendę:

```
IVTRun -nojndi [-m qmgr] [-v providerVersion] [-t]
```

Aby uruchomić test w trybie klienta, należy najpierw skonfigurować menedżer kolejek zgodnie z opisem w sekcji “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113. Należy zauważyć, że domyślnym kanałem, który ma być używany, jest **SYSTEM.DEF.SVRCONN** , a kolejką, która ma być używana, jest **SYSTEM.DEFAULT.LOCAL.QUEUE**, a następnie należy wprowadzić następującą komendę:

```
IVTRun -nojndi -client -m qmgr -host hostname [-port port] [-channel channel]
       [-v providerVersion] [-ccsid ccsid] [-t]
```

W systemach z/OS nie udostępniono równoważnego skryptu, ale można uruchomić narzędzie IVT w trybie powiązań, wywołując bezpośrednio klasę Java za pomocą następującej komendy:

```
java com.ibm.mq.jms.MQJMSIVT -nojndi [-m qmgr] [-v providerVersion] [-t]
```

Ścieżka klasy musi zawierać com.ibm.mqjms.jar.

Parametry komend mają następujące znaczenie:

-m *menedżer_kolejek*
Nazwa menedżera kolejek, z którym łączy się program IVT. Jeśli test zostanie uruchomiony w trybie powiązań i parametr ten zostanie pominięty, program IVT nawiąże połączenie z domyślnym menedżerem kolejek.

-host *nazwa_hosta*
Nazwa hosta lub adres IP systemu, w którym działa menedżer kolejek.

-port *port*
Numer portu, na którym nasłuchuje proces nasłuchujący menedżera kolejek. Wartością domyślną jest 1414.

-channel ***kanał***
Nazwa kanału MQI używanego przez program IVT do nawiązywania połączenia z menedżerem kolejek. Wartością domyślną jest SYSTEM.DEF.SVRCONN.

-v ***providerVersion***
Poziom wersji menedżera kolejek, z którym program IVT ma się połączyć.

Ten parametr służy do ustawiania właściwości PROVIDERVERSION obiektu fabryki MQQueueConnectioni ma takie same poprawne wartości jak właściwość PROVIDERVERSION. Więcej informacji na temat tego parametru, w tym jego poprawne wartości, zawiera opis właściwości PROVIDERVERSION w sekcji Właściwości obiektów IBM WebSphere MQ classes for JMS .

Wartością domyślną jest unspecified.

-ccsid ***id_ccsid***
Identyfikator (CCSID) kodowanego zestawu znaków lub strony kodowej, który ma być używany przez połączenie. Wartością domyślną jest 819.

-t
Śledzenie jest włączone. Domyślnie śledzenie jest wyłączone.

Pomyślny test generuje dane wyjściowe podobne do poniższych przykładowych danych wyjściowych:

```
5724-H72, 5655-R36, 5724-L26, 5655-L82 (c) Copyright IBM Corp. 2008, 2024. All
Rights Reserved.
Websphere MQ classes for Java(tm) Message Service 7.0
Installation Verification Test

Creating a QueueConnectionFactory
Creating a Connection
Creating a Session
Creating a Queue
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message
  JMSMessage class: jms_text
  JMSType:          null
  JMSDeliveryMode:  2
  JMSExpiration:    0
  JMSPriority:      4
  JMSMessageID:     ID:414d5120514d5f6d627720202020202001edb14620005e03
  JMSTimestamp:     1187170264000
  JMSCorrelationID: null
  JMSDestination:   queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
  JMSReplyTo:       null
  JMSRedelivered:   false
    JMSXUserID: mwhite
    JMS_IBM_Encoding: 273
    JMS_IBM_PutApplType: 28
    JMSXAppID: WebSphere MQ Client for Java
    JMSXDeliveryCount: 1
    JMS_IBM_PutDate: 20070815
    JMS_IBM_PutTime: 09310400
    JMS_IBM_Format: MQSTR
    JMS_IBM_MsgType: 8
A simple text message from the MQJMSIVT
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
```

Test weryfikujący instalację punkt-punkt przy użyciu interfejsu JNDI

W tym teście program IVT używa interfejsu JNDI do pobierania obiektów administrowanych z usługi katalogowej.

Przed uruchomieniem testu należy skonfigurować usługę katalogową opartą na serwerze LDAP (Lightweight Directory Access Protocol) lub lokalnym systemie plików. Należy również skonfigurować narzędzie administracyjne JMS WebSphere MQ , aby mogło ono używać usługi katalogowej do przechowywania obiektów administrowanych. Więcej informacji na temat tych wymagań wstępnych zawiera sekcja “Wymagania wstępne dla klas produktu WebSphere MQ dla usługi JMS” na stronie 735. Informacje na temat konfigurowania narzędzia administracyjnego WebSphere MQ JMS zawiera sekcja “Konfigurowanie narzędzia administracyjnego JMS” na stronie 961.

Program IVT musi mieć możliwość użycia interfejsu JNDI w celu pobrania obiektu fabryki MQQueueConnectioni obiektu MQQueue z usługi katalogowej. W celu utworzenia tych obiektów administrowanych udostępniono skrypt. Skrypt ma nazwę IVTSetup w systemach UNIX and Linux i IVTSetup.bat w systemie Windowsi znajduje się w podkatalogu `bin` katalogu instalacyjnego WebSphere MQ classes for JMS. Aby uruchomić skrypt, wprowadź następującą komendę:

```
IVTSetup
```

Skrypt wywołuje narzędzie administracyjne JMS produktu WebSphere MQ w celu utworzenia obiektów administrowanych.

Obiekt fabryki MQQueueConnectionjest powiązany z nazwą ivtQCF i jest tworzony z wartościami domyślnymi dla wszystkich jego właściwości, co oznacza, że program IVT działa w trybie powiązań i nawiązuje połączenie z domyślnym menedżerem kolejek. Aby uruchomić program IVT w trybie klienta lub nawiązać połączenie z menedżerem kolejek innym niż domyślny menedżer kolejek, należy użyć narzędzia administracyjnego JMS produktu WebSphere MQ lub programu WebSphere MQ Explorer w celu zmiany odpowiednich właściwości obiektu fabryki MQQueueConnection. Informacje na temat korzystania z narzędzia administracyjnego WebSphere MQ JMS zawiera sekcja “Korzystanie z narzędzia administracyjnego JMS produktu WebSphere MQ” na stronie 960. Informacje na temat korzystania z programu WebSphere MQ Explorer zawiera pomoc dostarczana z programem WebSphere MQ Explorer.

Obiekt MQQueue jest powiązany z nazwą ivtQ i jest tworzony z wartościami domyślnymi dla wszystkich jego właściwości, z wyjątkiem właściwości QUEUE, która ma wartość SYSTEM.DEFAULT.LOCAL.QUEUE.

Po utworzeniu obiektów administrowanych można uruchomić program IVT. Aby uruchomić test przy użyciu interfejsu JNDI, wprowadź następującą komendę:

```
IVTRun -url "providerURL" [-icf initCtxFact] [-t]
```

Parametry komendy mają następujące znaczenie:

-url "*providerURL*"
Adres URL (Uniform Resource Locator) usługi katalogowej. Adres URL może mieć jeden z następujących formatów:

- `ldap://`*`hostname`*`/`*`contextName`*dla usługi katalogowej opartej na serwerze LDAP
- `file:/`*`directoryPath`*dla usługi katalogowej opartej na lokalnym systemie plików

Należy pamiętać, że adres URL należy ująć w cudzysłów (").

-icf *initCtxFakt*
Nazwa klasy fabryki kontekstu początkowego, która musi mieć jedną z następujących wartości:

- `com.sun.jndi.ldap.LdapCtxFactory`dla usługi katalogowej opartej na serwerze LDAP. Jest to wartość domyślna.
- `com.sun.jndi.fscontext.RefFSContextFactory`dla usługi katalogowej opartej na lokalnym systemie plików.

-t
Śledzenie jest włączone. Domyślnie śledzenie jest wyłączone.

Test zakończony powodzeniem generuje dane wyjściowe podobne do danych wyjściowych dla pomyślnego testu bez używania interfejsu JNDI. Główna różnica polega na tym, że dane wyjściowe wskazują, że test używa interfejsu JNDI do pobierania obiektu fabryki MQQueueConnectioni obiektu MQQueue.

Chociaż nie jest to ściśle konieczne, dobrą praktyką jest porządkowanie po teście przez usunięcie obiektów administrowanych utworzonych przez skrypt IVTSetup. W tym celu udostępniono skrypt. Skrypt ma nazwę IVTTidy w systemach UNIX and Linux i IVTTidy.bat w systemie Windowsi znajduje się w podkatalogu `bin` katalogu instalacyjnego produktu WebSphere MQ classes for JMS.

Określanie problemu dla testu weryfikującego instalację punkt-punkt

Test weryfikujący instalację może zakończyć się niepowodzeniem z następujących powodów:

- Jeśli program IVT zapisze komunikat informujący, że nie może znaleźć klasy, sprawdź, czy ścieżka klasy jest ustawiona poprawnie, zgodnie z opisem w sekcji “Zmienne środowiskowe używane przez klasy produktu IBM WebSphere MQ dla usługi JMS” na stronie 741.
- Test może zakończyć się niepowodzeniem z następującym komunikatem:

  ```
  Failed to connect to queue manager 'qmgr' with connection mode 'connMode'
  and host name 'hostname'
  ```

  i powiązany kod przyczyny 2059. Zmienne w komunikacie mają następujące znaczenie:

  QMGR
  Nazwa menedżera kolejek, z którym program IVT próbuje się połączyć. Wstawienie tego komunikatu jest puste, jeśli program IVT próbuje nawiązać połączenie z domyślnym menedżerem kolejek w trybie powiązań.

  connMode
  Tryb połączenia: `Bindings` lub `Client`.

  nazwa_hosta
  Nazwa hosta lub adres IP systemu, w którym działa menedżer kolejek.

  Ten komunikat oznacza, że menedżer kolejek, z którym próbuje nawiązać połączenie program IVT, nie jest dostępny. Sprawdź, czy menedżer kolejek jest uruchomiony i jeśli program IVT próbuje nawiązać połączenie z domyślnym menedżerem kolejek, upewnij się, że menedżer kolejek jest zdefiniowany jako domyślny menedżer kolejek dla systemu.
- Test może zakończyć się niepowodzeniem z następującym komunikatem:

  ```
  Failed to open MQ queue 'SYSTEM.DEFAULT.LOCAL.QUEUE'
  ```

  Ten komunikat oznacza, że kolejka SYSTEM.DEFAULT.LOCAL.QUEUE nie istnieje w menedżerze kolejek, z którym jest połączony program IVT. Alternatywnie, jeśli kolejka istnieje, program IVT nie może otworzyć kolejki, ponieważ nie ma możliwości umieszczania i pobierania komunikatów. Sprawdź, czy kolejka istnieje i czy można w niej wstawiać i pobierać komunikaty.
- Test może zakończyć się niepowodzeniem z następującym komunikatem:

  ```
  Unable to bind to object
  ```

  Ten komunikat oznacza, że istnieje połączenie z serwerem LDAP, ale serwer LDAP nie jest poprawnie skonfigurowany. Albo serwer LDAP nie jest skonfigurowany do przechowywania obiektów Java, albo uprawnienia do obiektów lub przyrostka nie są poprawne. Więcej informacji na ten temat zawiera dokumentacja serwera LDAP.
- Test może zakończyć się niepowodzeniem z następującym komunikatem:

  ```
  The security authentication was not valid that was supplied for
  QueueManager 'qmgr' with connection mode 'Client' and host name 'hostname'
  ```

Ten komunikat oznacza, że menedżer kolejek nie jest poprawnie skonfigurowany do akceptowania połączenia klienta z systemu. Szczegółowe informacje można znaleźć w sekcji “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113.

Test weryfikujący instalację publikowania/subskrypcji dla klas WebSphere MQ dla usługi JMS

Program testu sprawdzającego instalację publikowania/subskrypcji (IVT) jest dostarczany z klasami produktu WebSphere MQ dla usługi JMS. Program nawiązuje połączenie z menedżerem kolejek w trybie powiązań lub w trybie klienta, subskrybuje temat, publikuje komunikat w temacie, a następnie odbiera komunikat, który właśnie opublikował. Program może tworzyć i konfigurować wszystkie obiekty, które są wymagane dynamicznie w czasie wykonywania, lub może używać interfejsu JNDI do pobierania obiektów administrowanych z usługi katalogowej.

Uruchom test weryfikujący instalację bez używania interfejsu JNDI jako pierwszego, ponieważ test jest samodzielny i nie wymaga użycia usługi katalogowej. Opis obiektów administrowanych zawiera sekcja “Typy obiektów JMS” na stronie 965.

Test weryfikujący instalację publikowania/subskrypcji bez użycia interfejsu JNDI

W tym teście program IVT tworzy i konfiguruje wszystkie obiekty, które są wymagane dynamicznie w czasie wykonywania i nie używa interfejsu JNDI.

Dostępny jest skrypt uruchamiający program IVT. Skrypt nosi nazwę PSIVTRun w systemach UNIX and Linux i PSIVTRun.bat w systemie Windowsi znajduje się w podkatalogu `bin` katalogu instalacyjnego WebSphere MQ classes for JMS.

Aby uruchomić test w trybie powiązań, wprowadź następującą komendę:

```
PSIVTRun -nojndi [-m qmgr] [-bqm brokerQmgr] [-v providerVersion] [-t]
```

Aby uruchomić test w trybie klienta, należy najpierw skonfigurować menedżer kolejek zgodnie z opisem w sekcji “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113 , która wskazuje, że kanał, który ma być używany, ma ustawioną wartość domyślną SYSTEM.DEF.SVRCONN, a następnie wprowadź następującą komendę:

```
PSIVTRun -nojndi -client -m qmgr -host hostname [-port port] [-channel channel]
         [-bqm brokerQmgr] [-v providerVersion] [-ccsid ccsid] [-t]
```

Parametry komend mają następujące znaczenie:

-m *menedżer_kolejek*
Nazwa menedżera kolejek, z którym łączy się program IVT. Jeśli test zostanie uruchomiony w trybie powiązań i parametr ten zostanie pominięty, program IVT nawiąże połączenie z domyślnym menedżerem kolejek.

-host *nazwa_hosta*
Nazwa hosta lub adres IP systemu, w którym działa menedżer kolejek.

-port *port*
Numer portu, na którym nasłuchuje proces nasłuchujący menedżera kolejek. Wartością domyślną jest 1414.

-channel *kanał*
Nazwa kanału MQI używanego przez program IVT do nawiązywania połączenia z menedżerem kolejek. Wartością domyślną jest `SYSTEM.DEF.SVRCONN`.

-bqm *brokerQmgr*
Nazwa menedżera kolejek, w którym działa broker. Wartością domyślną jest nazwa menedżera kolejek, z którym łączy się program IVT.

Ten parametr ma zastosowanie tylko wtedy, gdy parametr -v określa wersję menedżera kolejek mniejszą niż 7, a jako broker publikowania/subskrypcji używany jest produkt WebSphere Event Broker lub WebSphere Message Broker.

-v *providerVersion*
Poziom wersji menedżera kolejek, z którym program IVT ma się połączyć.

Ten parametr służy do ustawiania właściwości PROVIDERVERSION obiektu fabryki MQTopicConnectioni ma takie same poprawne wartości jak właściwość PROVIDERVERSION. Więcej informacji na temat tego parametru, w tym jego poprawne wartości, zawiera opis właściwości PROVIDERVERSION w sekcji Właściwości obiektów IBM WebSphere MQ classes for JMS .

Wartością domyślną jest `unspecified`.

-ccsid *id_ccsid*
Identyfikator (CCSID) kodowanego zestawu znaków lub strony kodowej, który ma być używany przez połączenie. Wartością domyślną jest 819.

-t
Śledzenie jest włączone. Domyślnie śledzenie jest wyłączone.

Pomyślny test generuje dane wyjściowe podobne do poniższych przykładowych danych wyjściowych:

```
5724-H72, 5655-R36, 5724-L26, 5655-L82 (c) Copyright IBM Corp. 2008, 2024. All
Rights Reserved.
Websphere MQ classes for Java(tm) Message Service 7.0
Publish/Subscribe Installation Verification Test

Creating a TopicConnectionFactory
Creating a Connection
Creating a Session
Creating a Topic
Creating a TopicPublisher
Creating a TopicSubscriber
Creating a TextMessage
Adding text
Publishing the message to topic://MQJMS/PSIVT/Information
Waiting for a message to arrive [5 secs max]...

Got message:
  JMSMessage class: jms_text
  JMSType:          null
  JMSDeliveryMode:  2
  JMSExpiration:    0
  JMSPriority:      4
  JMSMessageID:     ID:414d5120514d5f6d627720202020202001edb14620006706
  JMSTimestamp:     1187182520203
  JMSCorrelationID: ID:414d5120514d5f6d627720202020202001edb14620006704
  JMSDestination:   topic://MQJMS/PSIVT/Information
  JMSReplyTo:       null
  JMSRedelivered:   false
    JMSXUserID: mwhite
    JMS_IBM_Encoding: 273
    JMS_IBM_PutApplType: 26
    JMSXAppID: QM_mbw
    JMSXDeliveryCount: 1
    JMS_IBM_PutDate: 20070815
    JMS_IBM_ConnectionID: 414D5143514D5F6D627720202020202001EDB14620006601
    JMS_IBM_PutTime: 12552020
    JMS_IBM_Format: MQSTR
    JMS_IBM_MsgType: 8
A simple text message from the MQJMSPSIVT program
Reply string equals original string
Closing TopicSubscriber
Closing TopicPublisher
Closing Session
Closing Connection
PSIVT finished
```

Test weryfikujący instalację publikowania/subskrypcji przy użyciu interfejsu JNDI

W tym teście program IVT używa interfejsu JNDI do pobierania obiektów administrowanych z usługi katalogowej.

Przed uruchomieniem testu należy skonfigurować usługę katalogową opartą na serwerze LDAP (Lightweight Directory Access Protocol) lub lokalnym systemie plików. Należy również skonfigurować narzędzie administracyjne JMS WebSphere MQ , aby mogło ono używać usługi katalogowej do przechowywania obiektów administrowanych. Więcej informacji na temat tych wymagań wstępnych zawiera sekcja “Wymagania wstępne dla klas produktu WebSphere MQ dla usługi JMS” na stronie 735. Informacje na temat konfigurowania narzędzia administracyjnego WebSphere MQ JMS zawiera sekcja “Konfigurowanie narzędzia administracyjnego JMS” na stronie 961.

Program IVT musi mieć możliwość użycia interfejsu JNDI w celu pobrania obiektu fabryki MQTopicConnectioni obiektu MQTopic z usługi katalogowej. W celu utworzenia tych obiektów administrowanych udostępniono skrypt. Skrypt ma nazwę IVTSetup w systemach UNIX and Linux i IVTSetup.bat w systemie Windowsi znajduje się w podkatalogu `bin` katalogu instalacyjnego WebSphere MQ classes for JMS. Aby uruchomić skrypt, wprowadź następującą komendę:

```
IVTSetup
```

Skrypt wywołuje narzędzie administracyjne JMS produktu WebSphere MQ w celu utworzenia obiektów administrowanych.

Obiekt fabryki MQTopicConnectionjest powiązany z nazwą ivtTCF i jest tworzony z wartościami domyślnymi dla wszystkich jego właściwości, co oznacza, że program IVT działa w trybie powiązań, łączy się z domyślnym menedżerem kolejek i używa wbudowanej funkcji publikowania/subskrybowania. Jeśli program IVT ma działać w trybie klienta, nawiąż połączenie z menedżerem kolejek innym niż domyślny menedżer kolejek lub użyj funkcji WebSphere Event Broker lub WebSphere Message Broker zamiast wbudowanej funkcji publikowania/subskrypcji. należy użyć narzędzia administracyjnego WebSphere MQ JMS lub programu WebSphere MQ Explorer, aby zmienić odpowiednie właściwości obiektu fabryki MQTopicConnection. Informacje na temat korzystania z narzędzia administracyjnego WebSphere MQ JMS zawiera sekcja “Korzystanie z narzędzia administracyjnego JMS produktu WebSphere MQ” na stronie 960. Informacje na temat korzystania z programu WebSphere MQ Explorer zawiera pomoc dostarczana z programem WebSphere MQ Explorer.

Obiekt MQTopic jest powiązany z nazwą ivtT i jest tworzony z wartościami domyślnymi dla wszystkich jego właściwości, z wyjątkiem właściwości TOPIC, która ma wartość MQJMS/PSIVT/Information.

Po utworzeniu obiektów administrowanych można uruchomić program IVT. Aby uruchomić test przy użyciu interfejsu JNDI, wprowadź następującą komendę:

```
PSIVTRun -url "providerURL" [-icf initCtxFact] [-t]
```

Parametry komendy mają następujące znaczenie:

-url "*providerURL*"
Adres URL (Uniform Resource Locator) usługi katalogowej. Adres URL może mieć jeden z następujących formatów:

- `ldap://`*`hostname`*`/`*`contextName`*dla usługi katalogowej opartej na serwerze LDAP
- `file:/`*`directoryPath`*dla usługi katalogowej opartej na lokalnym systemie plików

Należy pamiętać, że adres URL należy ująć w cudzysłów (").

-icf *initCtxFakt*
Nazwa klasy fabryki kontekstu początkowego, która musi mieć jedną z następujących wartości:

- `com.sun.jndi.ldap.LdapCtxFactory`dla usługi katalogowej opartej na serwerze LDAP. Jest to wartość domyślna.
- `com.sun.jndi.fscontext.RefFSContextFactory`dla usługi katalogowej opartej na lokalnym systemie plików.

-t
Śledzenie jest włączone. Domyślnie śledzenie jest wyłączone.

Test zakończony powodzeniem generuje dane wyjściowe podobne do danych wyjściowych dla pomyślnego testu bez używania interfejsu JNDI. Główna różnica polega na tym, że dane wyjściowe

wskazują, że test używa interfejsu JNDI do pobrania obiektu fabryki MQTopicConnectioni obiektu MQTopic.

Chociaż nie jest to ściśle konieczne, dobrą praktyką jest porządkowanie po teście przez usunięcie obiektów administrowanych utworzonych przez skrypt IVTSetup. W tym celu udostępniono skrypt. Skrypt ma nazwę IVTTidy w systemach UNIX and Linux i IVTTidy.bat w systemie Windowsi znajduje się w podkatalogu `bin` katalogu instalacyjnego produktu WebSphere MQ classes for JMS.

Określanie problemu dla testu weryfikującego instalację publikowania/subskrypcji

Test weryfikujący instalację może zakończyć się niepowodzeniem z następujących powodów:

- Jeśli program IVT zapisze komunikat informujący, że nie może znaleźć klasy, sprawdź, czy ścieżka klasy jest ustawiona poprawnie, zgodnie z opisem w sekcji "Zmienne środowiskowe używane przez klasy produktu IBM WebSphere MQ dla usługi JMS" na stronie 741.
- Test może zakończyć się niepowodzeniem z następującym komunikatem:

  ```
  Failed to connect to queue manager 'qmgr' with
  connection mode 'connMode' and host name 'hostname'
  ```

  i powiązany kod przyczyny 2059. Zmienne w komunikacie mają następujące znaczenie:

  QMGR
  Nazwa menedżera kolejek, z którym program IVT próbuje się połączyć. Wstawienie tego komunikatu jest puste, jeśli program IVT próbuje nawiązać połączenie z domyślnym menedżerem kolejek w trybie powiązań.

  connMode
  Tryb połączenia: `Bindings` lub `Client`.

  nazwa_hosta
  Nazwa hosta lub adres IP systemu, w którym działa menedżer kolejek.

  Ten komunikat oznacza, że menedżer kolejek, z którym próbuje nawiązać połączenie program IVT, nie jest dostępny. Sprawdź, czy menedżer kolejek jest uruchomiony i jeśli program IVT próbuje nawiązać połączenie z domyślnym menedżerem kolejek, upewnij się, że menedżer kolejek jest zdefiniowany jako domyślny menedżer kolejek dla systemu.
- Test może zakończyć się niepowodzeniem z następującym komunikatem:

  ```
  Unable to bind to object
  ```

  Ten komunikat oznacza, że istnieje połączenie z serwerem LDAP, ale serwer LDAP nie jest poprawnie skonfigurowany. Albo serwer LDAP nie jest skonfigurowany do przechowywania obiektów Java, albo uprawnienia do obiektów lub przyrostka nie są poprawne. Więcej informacji na ten temat zawiera dokumentacja serwera LDAP.
- Test może zakończyć się niepowodzeniem z następującym komunikatem:

  ```
  The security authentication was not valid that was supplied for
  QueueManager 'qmgr' with connection mode 'Client' and host name 'hostname'
  ```

  Ten komunikat oznacza, że menedżer kolejek nie jest poprawnie skonfigurowany do akceptowania połączenia klienta z systemu. Więcej informacji na ten temat zawiera "Przygotowywanie i uruchamianie przykładowych programów" na stronie 113.

Program testowy weryfikujący instalację dla adaptera zasobów produktu WebSphere MQ

Program IVT jest dostarczany jako plik EAR. Aby korzystać z programu, należy go wdrożyć i zdefiniować niektóre obiekty jako zasoby JCA.

Program do sprawdzania poprawności instalacji (IVT) jest dostarczany jako plik archiwum korporacyjnego (EAR) o nazwie wmq.jmsra.ivt.ear. Ten plik jest instalowany z klasami produktu WebSphere MQ dla usługi JMS w tym samym katalogu, w którym znajduje się plik RAR adaptera zasobów produktu WebSphere

MQ (wmq.jmsra.rar). Informacje o tym, gdzie te pliki są zainstalowane, zawiera sekcja “Instalowanie adaptera zasobów produktu WebSphere MQ” na stronie 753.

Należy wdrożyć program IVT na serwerze aplikacji. Program IVT zawiera serwlet oraz komponent MDB, który testuje, że komunikat może zostać wysłany do kolejki produktu WebSphere MQ i odebrany z niej. Opcjonalnie można użyć programu IVT, aby sprawdzić, czy adapter zasobów WebSphere MQ został poprawnie skonfigurowany do obsługi transakcji rozproszonych.

Przed uruchomieniem programu IVT należy zdefiniować obiekt ConnectionFactory , obiekt kolejki i być może obiekt specyfikacji aktywowania jako zasoby JCA, a także upewnić się, że serwer aplikacji tworzy obiekty JMS z tych definicji i wiąże je z przestrzenią nazw JNDI. Użytkownik może wybrać właściwości obiektów, ale następujący zestaw właściwości jest prostym przykładem:

Obiekt ConnectionFactory

```
channel:              SYSTEM.DEF.SVRCONN
hostName:             localhost
port:                 1414
queueManager:         ExampleQM
transportType:        CLIENT
```

Obiekt kolejki

```
baseQueueManagerName:  ExampleQM
baseQueueName:         TEST.QUEUE
```

Domyślnie program IVT oczekuje, że obiekt ConnectionFactory będzie powiązany w przestrzeni nazw JNDI z nazwą jms/ivt/IVTCF i obiektem kolejki, który ma być powiązany z nazwą jms/ivt/IVTQueue. Można użyć różnych nazw, ale jeśli to zrobisz, musisz wprowadzić nazwy obiektów na początkowej stronie programu IVT i odpowiednio zmodyfikować plik EAR.

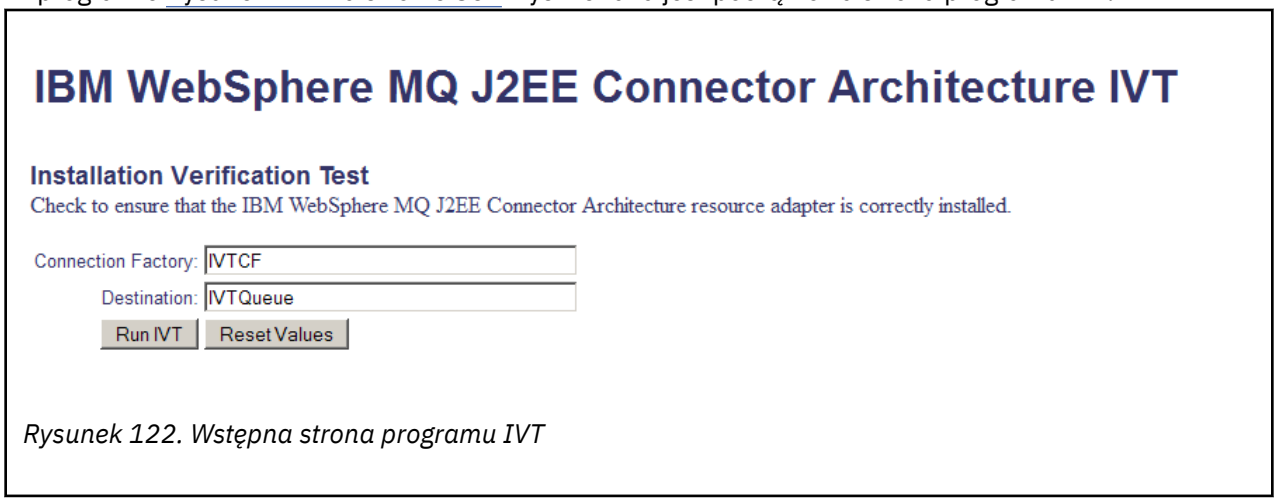
Po wdrożeniu programu IVT, gdy serwer aplikacji utworzył obiekty JMS i powiąże je z przestrzenią nazw JNDI, można uruchomić program IVT, wprowadzając adres URL w następującym formacie w przeglądarce WWW:

```
http://app_server_host:port/WMQ_IVT/
```

gdzie *host_serwera_aplikacji* to adres IP lub nazwa hosta systemu, na którym działa serwer aplikacji, a *port* jest numerem portu TCP, na którym nasłuchuje serwer aplikacji. Oto przykład:

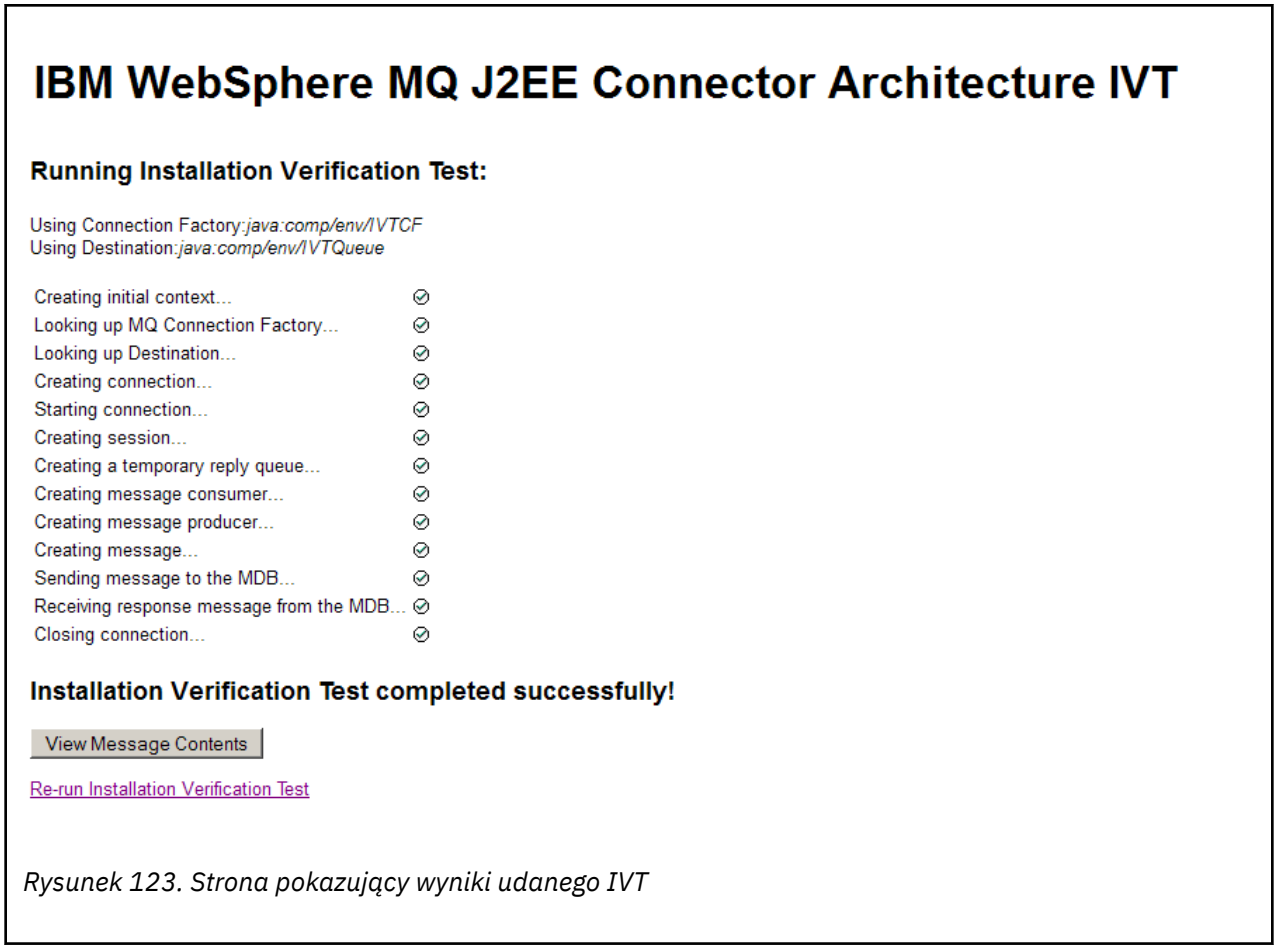
```
http://localhost:9080/WMQ_IVT/
```

W programie Rysunek 122 na stronie 804 wyświetlana jest początkowa strona programu IVT.



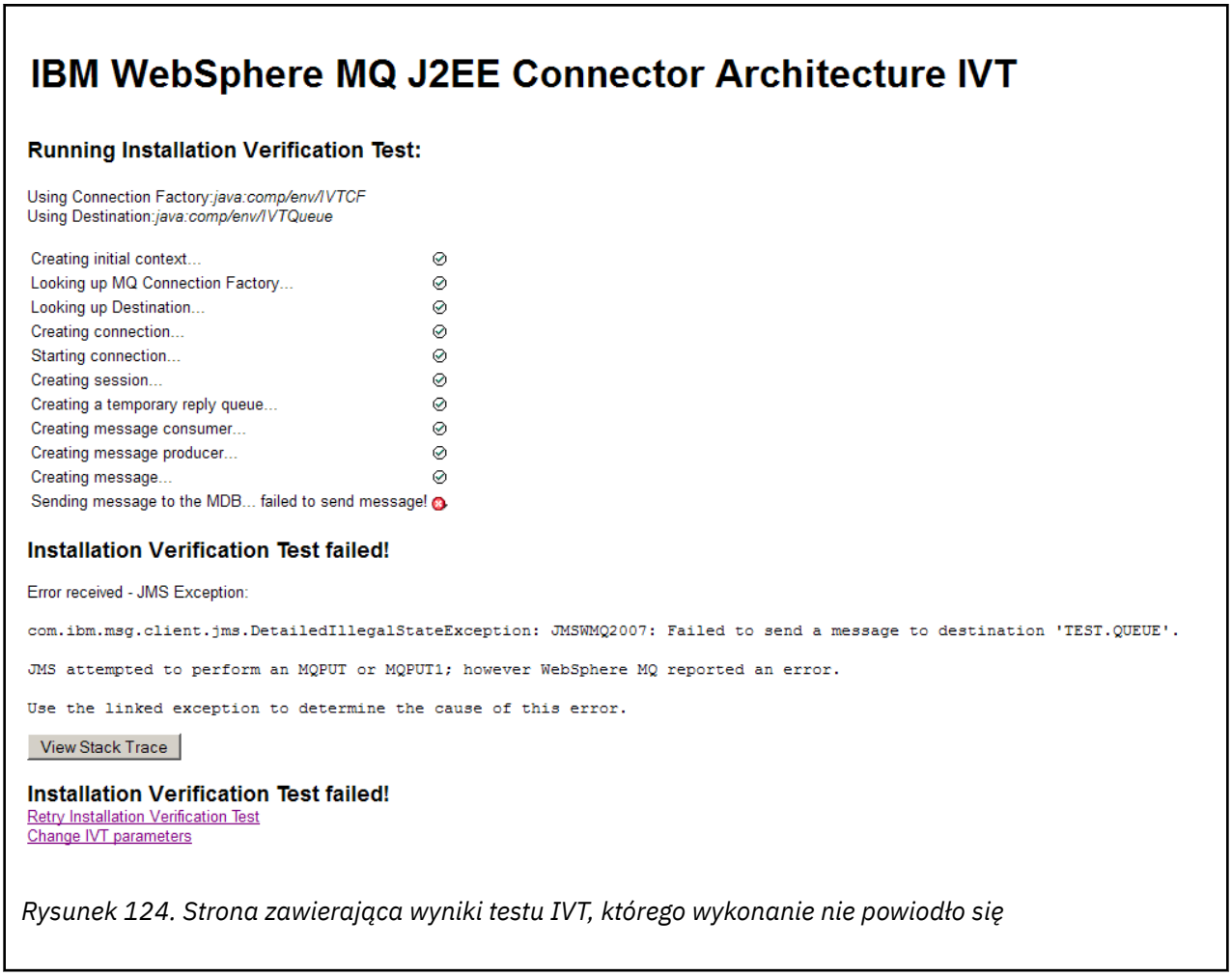
Rysunek 122. Wstępna strona programu IVT

Aby uruchomić test, kliknij opcję **Uruchom IVT**. Rysunek 123 na stronie 805 Wyświetla stronę, która jest wyświetlana, jeśli program IVT zakończy się pomyślnie.



Rysunek 123. Strona pokazujący wyniki udanego IVT

Jeśli test IVT nie powiedzie się, zostanie wyświetlona strona taka jak wyświetlana w sekcji Rysunek 124 na stronie 806 . Aby uzyskać dalsze informacje na temat przyczyny niepowodzenia, kliknij opcję **Wyświetl stos wywołań**.



Rysunek 124. Strona zawierająca wyniki testu IVT, którego wykonanie nie powiodło się

Szczegółowe instrukcje i informacje na temat skryptów programu narzędziowego udostępnionych w celu wdrożenia aplikacji IVT na serwerach aplikacji JBoss i WAS CE można znaleźć pod adresem:

Zadania pokrewne

“Instalowanie i testowanie adaptera zasobów produktu MQ w produkcie WAS CE” na stronie 806
Instalowanie adaptera zasobów produktu IBM WebSphere MQ i uruchomienie testu sprawdzania poprawności instalacji (IVT) w produkcie WebSphere Application Server CE.

“Instalowanie i testowanie adaptera zasobów w produkcie JBoss AS 5.1 i 6” na stronie 809
Po zainstalowaniu adaptera zasobów IBM WebSphere MQ w systemie JBoss AS 5.1 lub 6 można przetestować instalację adaptera zasobów, instalując i uruchamiając aplikację testową weryfikacyjną instalacji (IVT).

Instalowanie i testowanie adaptera zasobów produktu MQ w produkcie WAS CE

Instalowanie adaptera zasobów produktu IBM WebSphere MQ i uruchomienie testu sprawdzania poprawności instalacji (IVT) w produkcie WebSphere Application Server CE.

Zanim rozpoczniesz

W przypadku tego zadania założono, że użytkownik korzysta z działającego serwera WebSphere Application Server CE i że użytkownik jest zaznajomiony z zadaniami administrowania standardowymi. W tym zadaniu założono również, że w systemie lokalnym jest dostępna instalacja produktu IBM WebSphere MQ , a użytkownik jest zaznajomiony ze standardowymi zadaniami administracyjnymi.

Jeśli do nawiązywania połączenia z klientem IBM WebSphere MQ używany jest adapter zasobów i konieczne jest przeprowadzenie rozproszonych transakcji XA, należy wykonać dodatkowe kroki oznaczone jako **Tylko klient XA**.

1. Utwórz menedżer kolejek o nazwie `ExampleQM`i skonfiguruj go zgodnie z opisem w sekcji “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113 , zwracając uwagę, że nasłuchiwanie powinno być uruchomione na porcie 1414, kanał, który ma być używany, nosi nazwę `SYSTEM.DEF.SVRCONN` , a kolejka używana przez aplikację IVT ma nazwę `TEST.QUEUE`. Kolejka modelowa `SYSTEM.DEFAULT.MODEL.QUEUE` również musi mieć nadane uprawnienia DSP i PUT, aby aplikacja mogła utworzyć tymczasową kolejkę odpowiedzi. Aby użyć innego menedżera kolejek, innych szczegółów połączenia lub innej kolejki, należy zapoznać się z informacjami w sekcji “Wdrażanie aplikacji IVT na serwerze WAS CE z niestandardowym środowiskiem MQ” na stronie 808.
2. Uzyskaj plik adaptera zasobów (`wmq.jmsra.rar`), aplikację IVT (`wmq.jmsra.ivt.ear`), a także `WAS_CE_jmsra_deployment_plan.xml` i `WAS_CE_jmsra_ivt_deployment_plan.xml` `deployment plan files`. Szczegółowe informacje na temat położenia tych plików zawiera sekcja “Instalowanie adaptera zasobów produktu WebSphere MQ” na stronie 753.

Opis powiązań i połączeń w trybie klienta można znaleźć w sekcji “Tryby połączenia dla klas produktu WebSphere MQ dla usługi JMS” na stronie 794.

Jeśli zamiast trybu klienta ma być używana inna kolejka, menedżer kolejek, port, host, kanał lub tryb powiązań, należy zapoznać się z “Wdrażanie aplikacji IVT na serwerze WAS CE z niestandardowym środowiskiem MQ” na stronie 808.

Procedura

1. **Tylko klient XA**: służy do edytowania kopii pliku `WAS_CE_jmsra_deployment_plan.xml` .
   a) Znajdź definicję połączenia jms/ivt/IVTCF i zmodyfikuj ją w taki sposób, aby fabryka połączeń była włączona w transakcji XA.
      i) Przekształć w komentarz sekcję NonXA :

      ```
      <conn:xa-transaction>
      ```

      ii) Usuń znak komentarza z sekcji konfiguracji XA:

      ```
      <conn:xa-transaction>
          <conn:transaction-caching/>
      </conn:xa-transaction>
      ```

   b) Zapisz zmiany.
2. Opcjonalne: **Tylko klient XA klienta**: Zmodyfikuj deskryptor zespołu komponentu MDB, aby wymagać transakcji. Wymusza to działanie komponentu MDB w tabeli IVT w celu uczestniczenia w transakcji XA, mimo że aplikacja IVT nadal działa bez tej modyfikacji.
   a) Otwórz plik `wmq.jmsra.ivt.ear`.
   b) Otwórz w nim plik `WMQ_IVT_MDB.jar` .
   c) Edytuj `META-INF/ejb-jar.xml`.
      i) Przekształć w komentarz lub usuń wiersz w deskryptorze zespołu:

      ```
      <trans-attribute>NotSupported</trans-attribute>
      ```

      ii) Usuń znak komentarza z wiersza w deskryptorze zespołu:

      ```
      <trans-attribute>Required</trans-attribute>
      ```

      iii) Zapisz zmiany i zaktualizuj plik w pliku `WMQ_IVT_MDB.jar` .
      iv) Zaktualizuj plik wmq.jmsra.ivt.ear ze zmodyfikowanym plikiem `WMQ_IVT_MDB.jar` .
3. Przeprowadź wdrożenie adaptera zasobów na serwerze przy użyciu zmodyfikowanego pliku planu wdrożenia.

a) Aby wykonać tę komendę w wierszu komend, wpisz następującą komendę WAS CE:

```
deploy -u system -p manager deploy wmq.jmsra.rar WAS_CE_jmsra_deployment_plan.xml
```

b) Za pomocą interfejsu administrowania WWW przejdź do sekcji **Aplikacje > Deployer:**

i) Ustaw archiwum jako plik wmq.jmsra.rar .

ii) Ustaw plan na plik WAS_CE_jmsra_deployment_plan.xml .

iii) Upewnij się, że opcja 'Uruchom aplikację po instalacji' jest wybrana.

iv) Kliknij opcję **Install**.

4. Wdróż aplikację IVT na serwerze, korzystając z udostępnionego planu wdrożenia.

a) W wierszu komend można to zrobić za pomocą następującej komendy WAS CE:

```
deploy -u system -p manager deploy wmq.jmsra.ivt.ear WAS_CE_jmsra_ivt_deployment_plan.xml
```

b) Za pomocą interfejsu administrowania WWW przejdź do sekcji **Aplikacje > Deployer:**

i) Ustaw w polu **Archiwum** wartość w pliku `wmq.jmsra.ivt.ear` .

ii) Ustaw wartość **Plan** na plik `WAS_CE_jmsra_ivt_deployment_plan.xml` .

iii) Upewnij się, że wybrana jest opcja **Uruchom aplikację po instalacji** .

iv) Kliknij opcję **Install**.

5. Uruchom aplikację IVT. Szczegółowe informacje na ten temat zawiera sekcja “Program testowy weryfikujący instalację dla adaptera zasobów produktu WebSphere MQ” na stronie 803. Domyślny adres URL serwera WAS CE to `http://localhost:8080/WMQ_IVT/`.

Wdrażanie aplikacji IVT na serwerze WAS CE z niestandardowym środowiskiem MQ

Jeśli zamiast trybu klienta ma być używana inna kolejka, menedżer kolejek, port, host, kanał lub tryb powiązań, należy zmodyfikować aplikację IVT i powiązane skrypty w produkcie WebSphere Application Server CE przed wdrożeniem adaptera zasobów lub aplikacji IVT.

O tym zadaniu

Aby wdrożyć w innej konfiguracji niż określono w programie “Instalowanie i testowanie adaptera zasobów produktu MQ w produkcie WAS CE” na stronie 806, to znaczy, jeśli zamiast trybu klienta ma być używana inna kolejka, menedżer kolejek, port, host, kanał lub tryb powiązań, należy wykonać następujące kroki przed wdrożeniem adaptera zasobów lub aplikacji IVT.

Procedura

1. Aby określić inny menedżer kolejek i kolejkę, która ma być używana dla aplikacji IVT, należy ustawić wartości dla menedżera kolejek i kolejki w produkcie `WAS_CE_jmsra_deployment_plan.xml`, aby uzyskać szczegółowe informacje na ten temat. Więcej informacji na ten temat zawiera sekcja “Ustawianie wartości dla menedżera kolejek i kolejki” na stronie 809.
2. Aby określić inny menedżer kolejek i kolejkę w konfiguracji komponentu bean sterowanego komunikatami (message-driven bean-MDB), ustaw wartości dla menedżera kolejek i kolejki, które są używane w produkcie `WAS_CE_jmsra_ivt_deployment_plan.xml`, aby uzyskać szczegółowe informacje na ten temat, patrz sekcja “Ustawianie wartości dla konfiguracji komponentu MDB” na stronie 809.
3. W przypadku konfigurowania adaptera zasobów w celu nawiązania połączenia z produktem IBM WebSphere MQ w trybie powiązań należy upewnić się, że biblioteki JNI znajdują się w ścieżce systemowej lub w ścieżce dla serwera WAS CE. Więcej informacji na ten temat zawiera sekcja “Instalowanie i testowanie adaptera zasobów produktu MQ w produkcie WAS CE” na stronie 806.
4. Jeśli wdrożono już adapter zasobów, można go ponownie wdrożyć za pomocą zmodyfikowanego planu wdrożenia w celu zmiany ustawień za pomocą następującej komendy:

```
deploy --user system --password manager redeploy wmq.jmsra.rar
WAS_CE_jmsra_deployment_plan.xml
```

Co dalej

Kontynuuj wdrażanie adaptera zasobów zgodnie z opisem w sekcji "Instalowanie i testowanie adaptera zasobów produktu MQ w produkcie WAS CE" na stronie 806.

Ustawianie wartości dla menedżera kolejek i kolejki

Wyjaśnia, w jaki sposób ustawić wartości dla menedżera kolejek i kolejki, które są używane w produkcie `WAS_CE_jmsra_deployment_plan.xml`.

Procedura

W programie `WAS_CE_jmsra_deployment_plan.xml`ustaw wartości dla menedżera kolejek i kolejki, które są używane dla aplikacji IVT.

Dla definicji połączenia jms/ivt/IVTCF:

1. Ustaw wartość elementu queueManager tak, aby była nazwą menedżera kolejek.
2. Jeśli używane jest połączenie klienckie, należy ustawić wartość różnych elementów połączenia klienta, aby były one odpowiednie dla połączenia z menedżerem kolejek.
3. W przypadku korzystania z połączenia powiązań:
   a. Ustaw wartość elementu transportType na wartość BINDINGS.
   b. Przekształć w komentarz lub usuń różne elementy połączenia klienta.
4. W przypadku miejsca docelowego komunikatów jms/ivt/IVTQueue ustaw wartość elementu nazwy baseQueuena nazwę kolejki utworzonej dla aplikacji IVT.
5. Zapisz zmiany.

Ustawianie wartości dla konfiguracji komponentu MDB

Zawiera informacje na temat ustawiania wartości dla konfiguracji komponentu MDB w produkcie `WAS_CE_jmsra_deployment_plan.xml`.

Procedura

W programie `WAS_CE_jmsra_ivt_deployment_plan.xml` ustaw wartości dla menedżera kolejek i kolejki, które są używane w konfiguracji dla komponentu MDB.

Dla sterowanego komunikatami komponentu bean sterowanego komunikatami WMQ_IVT_MDB:

1. Ustaw wartość elementu queueManager tak, aby była nazwą menedżera kolejek.
2. Jeśli używane jest połączenie klienckie, należy ustawić wartość różnych elementów połączenia klienta, aby były one odpowiednie dla połączenia z menedżerem kolejek.
3. W przypadku korzystania z połączenia powiązań:
   a. Ustaw wartość elementu transportType na wartość BINDINGS.
   b. Przekształć w komentarz lub usuń różne elementy połączenia klienta.
4. Zapisz zmiany.

Instalowanie i testowanie adaptera zasobów w produkcie JBoss AS 5.1 i 6

Po zainstalowaniu adaptera zasobów IBM WebSphere MQ w systemie JBoss AS 5.1 lub 6 można przetestować instalację adaptera zasobów, instalując i uruchamiając aplikację testową weryfikacyjną instalacji (IVT).

Zanim rozpoczniesz

Ważne: Te instrukcje dotyczą produktu JBoss AS 5.1 i 6, nie są one poprawne dla produktu JBoss AS 7.

Informacje na temat instalowania adaptera zasobów w produkcie JBoss EAP 6.3można znaleźć w sekcji “Instalowanie i testowanie adaptera zasobów w produkcie JBoss EAP 6.3” na stronie 812.

W przypadku tego zadania założono, że użytkownik korzysta z działającego serwera JBoss i jest zaznajomiony z zadaniami administrowania standardowymi. W tym zadaniu założono również, że w systemie lokalnym jest dostępna instalacja produktu IBM WebSphere MQ , a użytkownik jest zaznajomiony ze standardowymi zadaniami administracyjnymi.

Jeśli do nawiązywania połączenia z klientem IBM WebSphere MQ jest używany adapter zasobów i konieczne jest przeprowadzenie rozproszonych transakcji XA, należy wykonać dodatkowe kroki oznaczone jako **Tylko klient XA klienta**. Opis powiązań i połączeń w trybie klienta można znaleźć w sekcji “Tryby połączenia dla klas produktu WebSphere MQ dla usługi JMS” na stronie 794.

Procedura

1. Utwórz menedżer kolejek o nazwie ExampleQMi ustaw go zgodnie z opisem w sekcji “Przygotowywanie i uruchamianie przykładowych programów” na stronie 113.

   Podczas konfigurowania menedżera kolejek należy zwrócić uwagę na następujące punkty:

   - Nasłuchiwanie musi być uruchomione na porcie 1414.
   - Kanał, który ma być używany, nosi nazwę SYSTEM.DEF.SVRCONN.
   - Nazwa kolejki używana przez aplikację IVT nosi nazwę TEST.QUEUE.

   Kolejka modelowa SYSTEM.DEFAULT.MODEL.QUEUE musi również mieć nadane uprawnienia DSP i PUT, aby aplikacja mogła utworzyć tymczasową kolejkę odpowiedzi.

   Aby użyć innego menedżera kolejek, innych szczegółów połączenia lub innej kolejki, należy zapoznać się z informacjami w sekcji “Wdrażanie aplikacji IVT na serwerze WAS CE z niestandardowym środowiskiem MQ” na stronie 808.
2. Uzyskaj plik adaptera zasobów (`wmq.jmsra.rar`), aplikację IVT (`wmq.jmsra.ivt.ear`) i plik `jboss-jmsra-ds.xml` .

   Informacje na temat położenia tych plików zawiera sekcja “Instalowanie adaptera zasobów produktu WebSphere MQ” na stronie 753.
3. **Tylko klient XA klienta**: zmodyfikuj plik `jboss-jmsra-ds.xml` , aby włączyć transakcje XA w fabryce połączeń.
   a) Przekształć w komentarz lub usuń wiersz w definicji fabryki połączeń `<local-transaction/>`.
   b) Usuń znak komentarza z wiersza w definicji fabryki połączeń `<xa-transaction/>`.
   c) Zapisz zmiany.
4. **Tylko klient XA klienta**: (opcjonalnie) Zmodyfikuj deskryptor składania komponentu MDB, aby wymagać transakcji. Wymusza to działanie komponentu MDB w tabeli IVT w celu uczestniczenia w transakcji XA, mimo że aplikacja IVT nadal działa bez tej modyfikacji.
   a) Otwórz plik `wmq.jmsra.ivt.ear`.
   b) Otwórz w nim plik `WMQ_IVT_MDB.jar` .
   c) Edytuj `META-INF/ejb-jar.xml`:
      i) Przekształć w komentarz lub usuń wiersz w deskryptorze zespołu:

      ```
      <trans-attribute>NotSupported</trans-attribute>
      ```

      ii) Usuń znak komentarza z wiersza w deskryptorze zespołu:

      ```
      <trans-attribute>Required</trans-attribute>
      ```

      iii) Zapisz zmiany i zaktualizuj plik w pliku `WMQ_IVT_MDB.jar` .
      iv) Zaktualizuj plik `wmq.jmsra.ivt.ear` ze zmodyfikowaną `WMQ_IVT_MDB.jar`.

5. Przeprowadź wdrożenie adaptera zasobów na serwerze, kopiując plik `wmq.jmsra.rar` do katalogu `jboss/server/default/deploy`.
6. Utwórz zasoby JMS wymagane dla aplikacji IVT, kopiując plik `jboss-jmsra-ds.xml` do katalogu `jboss/server/default/deploy`.
7. Wdróż aplikację IVT, kopiując plik `wmq.jmsra.ivt.ear` do katalogu `jboss/server/default/deploy`.
8. Uruchom aplikację IVT. Szczegółowe informacje na ten temat zawiera sekcja “Program testowy weryfikujący instalację dla adaptera zasobów produktu WebSphere MQ” na stronie 803. W przypadku produktu JBossdomyślnym adresem URL jest `http://localhost:8080/WMQ_IVT/`.

Wdrażanie aplikacji IVT w produkcie JBoss przy użyciu niestandardowego środowiska produktu IBM WebSphere MQ

Jeśli adapter zasobów produktu IBM WebSphere MQ jest instalowany w produkcie JBoss, jeśli zamiast trybu klienta ma być używany inny tryb kolejki, menedżer kolejek, port, host, kanał lub powiązania, należy najpierw zmodyfikować aplikację IVT i powiązane skrypty w produkcie JBoss przed wdrożeniem adaptera zasobów lub aplikacji IVT.

O tym zadaniu

Ważne: Te instrukcje mają zastosowanie tylko w przypadku środowiska Java EE w wersjach 6 i 5, a nie dla środowiska Java EE 7. Użycie tych instrukcji dla produktu JBoss w wersji 8 (WildFly) nie jest zatem obsługiwane.

Aby wdrożyć w innej konfiguracji niż określono w programie “Instalowanie i testowanie adaptera zasobów w produkcie JBoss AS 5.1 i 6” na stronie 809, to znaczy, jeśli zamiast trybu klienta ma być używany inny menedżer kolejek, kolejka, port, host, kanał lub tryb powiązań, należy wykonać następujące kroki przed wdrożeniem adaptera zasobów lub aplikacji IVT.

Procedura

1. Aby określić inny menedżer kolejek i kolejkę, która ma być używana dla aplikacji IVT, należy ustawić wartości dla menedżera kolejek i kolejki.
   a) Dla definicji połączenia jms/ivt/IVTCF:
      i) Ustaw wartość właściwości queueManager , tak aby była ona nazwą menedżera kolejek.
      ii) Jeśli używane jest połączenie klienckie, należy ustawić wartość różnych elementów połączenia klienta, aby były one odpowiednie dla połączenia z menedżerem kolejek.
      iii) Jeśli używane jest połączenie powiązań, ustaw wartość elementu transportType na wartość BINDINGS, a następnie wykomentuj lub usuń różne elementy połączenia klienta.
   b) W przypadku komponentu mbean jms/ivt/IVTQueue ustaw wartość elementu nazwy baseQueuena nazwę kolejki utworzonej dla aplikacji IVT.
   c) Zapisz zmiany.
2. Aby określić inny menedżer kolejek i kolejkę w konfiguracji komponentu bean sterowanego komunikatami (message-driven bean-MDB), należy zmodyfikować konfigurację komponentu MDB w celu nawiązania połączenia z menedżerem kolejek i kolejką.
   a) Otwórz plik `wmq.jmsra.ivt.ear`.
   b) Otwórz w nim `WMQ_IVT_MDB.jar` .
   c) Edytuj `META-INF/ejb-jar.xml`:
      i) Ustaw wartość opcji queueManager activation-config-property, aby była nazwą menedżera kolejek.
      ii) W przypadku korzystania z połączenia klienckiego należy ustawić wartość różnych właściwości aktywowania-config-config-config-config, aby było to odpowiednie dla połączenia z menedżerem kolejek.

iii) Jeśli używane jest połączenie powiązań, ustaw wartość właściwości transportType activation-config-property na wartość BINDINGS, a następnie wykomentuj lub usuń różne elementy połączenia z klientem.

d) Zapisz zmiany i zaktualizuj plik w pliku `WMQ_IVT_MDB.jar` .

e) Zaktualizuj plik `wmq.jmsra.ivt.ear` ze zmodyfikowaną `WMQ_IVT_MDB.jar`.

3. Jeśli adapter zasobów jest konfigurowany w celu nawiązania połączenia z produktem IBM WebSphere MQ w trybie powiązań, należy upewnić się, że biblioteki JNI znajdują się w ścieżce systemowej lub w ścieżce JBoss. Szczegółowe informacje na ten temat zawiera sekcja "Konfigurowanie bibliotek Java Native Interface (JNI)" na stronie 743.

Co dalej

Kontynuuj wdrażanie adaptera zasobów zgodnie z opisem w sekcji "Instalowanie i testowanie adaptera zasobów w produkcie JBoss AS 5.1 i 6" na stronie 809.

.

Instalowanie i testowanie adaptera zasobów w produkcie JBoss EAP 6.3

Po zainstalowaniu adaptera zasobów produktu IBM WebSphere MQ w produkcie JBoss Enterprise Application Platform (EAP) 6.3na serwerze autonomicznym albo na serwerze działającym w domenie zarządzanej można przetestować instalację adaptera zasobów, instalując i uruchamiając aplikację testową weryfikacyjną instalacji (IVT).

O tym zadaniu

Ważne: Te instrukcje dotyczą tylko produktu JBoss EAP 6.3 . Informacje na temat instalowania adaptera zasobów w systemie JBoss AS 5.1 i 6 znajdują się w sekcji "Instalowanie i testowanie adaptera zasobów w produkcie JBoss AS 5.1 i 6" na stronie 809.

W przypadku tego zadania założono, że użytkownik korzysta z działającego serwera JBoss i jest zaznajomiony z zadaniami administrowania standardowymi. W tym zadaniu założono również, że w systemie lokalnym jest dostępna instalacja produktu IBM WebSphere MQ , a użytkownik jest zaznajomiony ze standardowymi administrowaniem.

Procedura

1. Utwórz menedżer kolejek o nazwie ExampleQMi ustaw go zgodnie z opisem w sekcji "Przygotowywanie i uruchamianie przykładowych programów" na stronie 113.

   Podczas konfigurowania menedżera kolejek należy zwrócić uwagę na następujące punkty:

   - Nasłuchiwanie musi być uruchomione na porcie 1414.
   - Kanał, który ma być używany, nosi nazwę SYSTEM.DEF.SVRCONN.
   - Nazwa kolejki używana przez aplikację IVT nosi nazwę TEST.QUEUE.

   Kolejka modelowa SYSTEM.DEFAULT.MODEL.QUEUE musi również mieć nadane uprawnienia DSP i PUT, aby aplikacja mogła utworzyć tymczasową kolejkę odpowiedzi.

   Aby użyć innego menedżera kolejek, innych szczegółów połączenia lub innej kolejki, należy zapoznać się z informacjami w sekcji "Wdrażanie aplikacji IVT na serwerze WAS CE z niestandardowym środowiskiem MQ" na stronie 808.
2. Uzyskaj plik adaptera zasobów (`wmq.jmsra.rar`) i aplikację IVT (`wmq.jmsra.ivt.ear`).

   Informacje na temat położenia tych plików zawiera sekcja "Instalowanie adaptera zasobów produktu WebSphere MQ" na stronie 753.
3. Zainstaluj adapter zasobów, a następnie przetestuj instalację, uruchamiając test sprawdzający instalację (IVT):

- Jeśli adapter zasobów jest instalowany na serwerze autonomicznym, należy zapoznać się z “Instalowanie i testowanie na serwerze autonomicznym” na stronie 813.
- Jeśli adapter zasobów jest instalowany na serwerze działającym w domenie zarządzanej, należy zapoznać się z “Instalowanie i testowanie na serwerze działającym w domenie zarządzanej” na stronie 814.

Instalowanie i testowanie na serwerze autonomicznym

Po zainstalowaniu adaptera zasobów produktu IBM WebSphere MQ w systemie JBoss EAP 6.3 na serwerze autonomicznym można przetestować instalację adaptera zasobów, instalując i uruchamiając aplikację testową weryfikacyjną instalacji (IVT).

O tym zadaniu

Informacje zawarte w tym zadaniu są przeznaczone do instalowania i testowania adaptera zasobów na serwerze autonomicznym. Jeśli adapter zasobów jest instalowany na serwerze działającym w domenie zarządzanej, należy zapoznać się z “Instalowanie i testowanie na serwerze działającym w domenie zarządzanej” na stronie 814.

Ważne: Te instrukcje dotyczą tylko produktu JBoss EAP 6.3 . Informacje na temat instalowania adaptera zasobów w systemie JBoss AS 5.1 i 6 znajdują się w sekcji “Instalowanie i testowanie adaptera zasobów w produkcie JBoss AS 5.1 i 6” na stronie 809.

Procedura

1. Przeprowadź wdrożenie adaptera zasobów na serwerze, kopiując plik `wmq.jmsra.rar` do katalogu <*EAP_HOME*>`/standalone/deployments`.
2. Utwórz zasoby JMS wymagane dla aplikacji IVT, dodając następujące wpisy do sekcji `<resource-adapters>` w pliku <*EAP_HOME*>`/standalone/configuration/standalone-full.xml` :

```
<resource-adapter id="wmq.jmsra">
  <archive>
      wmq.jmsra.rar
  </archive>
  <transaction-support>NoTransaction</transaction-support>
  <connection-definitions>
     <connection-definition
          class-name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
          jndi-name="java:jboss/jms/ivt/IVTCF"
          enabled="true"
          use-java-context="true"
          pool-name="IVTCF">
       <config-property name="port">
           1414
       </config-property>
       <config-property name="hostName">
           localhost
       </config-property>
       <config-property name="channel">
           SYSTEM.DEF.SVRCONN
       </config-property>
       <config-property name="transportType">
           CLIENT
       </config-property>
       <config-property name="queueManager">
           ExampleQM
       </config-property>
     </connection-definition>
  </connection-definitions>
  <admin-objects>
    <admin-object class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
                  jndi-name="java:jboss/jms/ivt/IVTQueue"
                  pool-name="IVTQueue">
      <config-property name="baseQueueName">
          TEST.QUEUE
      </config-property>
    </admin-object>
  </admin-objects>
</resource-adapter>
```

3. Dodaj następujące informacje do parametrów uruchamiania serwera aplikacji:

```
-Dcom.ibm.mq.connector.IVTMDBCFJNDIName=java:jboss/jms/ivt/IVTCF
```

4. Wdróż aplikację IVT, kopiując plik `wmq.jmsra.ivt.ear` do katalogu <*EAP_HOME*>`/standalone/deployments`.
5. Uruchom serwer aplikacji.
6. Uruchom aplikację IVT.

   Więcej informacji na ten temat zawiera sekcja “Program testowy weryfikujący instalację dla adaptera zasobów produktu WebSphere MQ” na stronie 803. W przypadku produktu JBossdomyślnym adresem URL jest `http://localhost:8080/WMQ_IVT/`.

   Uwaga: Nazwy JNDI używane na potrzeby zasobów JMS wymaganych do uruchamiania aplikacji IVT są następujące:

```
Connection Factory : IVTCF
JNDI name          : java:jboss/jms/ivt/IVTCF

Destination        : IVTQueue
JNDI name          : java:jboss/jms/ivt/IVTQueue
```

   Po uruchomieniu aplikacji IVT z użyciem podanego powyżej adresu URL wprowadź nazwy JNDI tych zasobów w odpowiednich polach, a następnie uruchom aplikację, klikając opcję **Uruchom program IVT**.

Instalowanie i testowanie na serwerze działającym w domenie zarządzanej

Po zainstalowaniu adaptera zasobów IBM WebSphere MQ w systemie JBoss EAP 6.3 na serwerze działającym w domenie zarządzanej można przetestować instalację adaptera zasobów, instalując i uruchamiając aplikację testową weryfikacyjną instalacji (IVT).

O tym zadaniu

Informacje zawarte w tym zadaniu są przeznaczone do instalowania i testowania adaptera zasobów na serwerze, który działa w domenie zarządzanej. Jeśli adapter zasobów jest instalowany na serwerze autonomicznym, należy zapoznać się z “Instalowanie i testowanie na serwerze autonomicznym” na stronie 813.

Ważne: Te instrukcje dotyczą tylko produktu JBoss EAP 6.3 . Informacje na temat instalowania adaptera zasobów w systemie JBoss AS 5.1 i 6 znajdują się w sekcji “Instalowanie i testowanie adaptera zasobów w produkcie JBoss AS 5.1 i 6” na stronie 809.

Procedura

1. Adapter zasobów należy wdrożyć na serwerze przy użyciu konsoli zarządzania JBoss lub interfejsu wiersza komend zarządzania.
2. Utwórz zasoby JMS wymagane dla aplikacji IVT, dodając następujące wpisy do sekcji `<resource-adapters>` w `<EAP_HOME>/domain/configuration/domain.xml file`:

```
<resource-adapter id="wmq.jmsra">
  <archive>
      wmq.jmsra.rar
  </archive>
  <transaction-support>NoTransaction</transaction-support>
  <connection-definitions>
    <connection-definition
         class-name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
         jndi-name="java:jboss/jms/ivt/IVTCF"
         enabled="true"
         use-java-context="true"
         pool-name="IVTCF">
      <config-property name="port">
          1414
      </config-property>
      <config-property name="hostName">
          localhost
```

```
      </config-property>
      <config-property name="channel">
          SYSTEM.DEF.SVRCONN
      </config-property>
      <config-property name="transportType">
          CLIENT
      </config-property>
      <config-property name="queueManager">
          ExampleQM
      </config-property>
    </connection-definition>
  </connection-definitions>
  <admin-objects>
    <admin-object class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
                  jndi-name="java:jboss/jms/ivt/IVTQueue"
                  pool-name="IVTQueue">
      <config-property name="baseQueueName">
          TEST.QUEUE
      </config-property>
    </admin-object>
  </admin-objects>
</resource-adapter>
```

3. Dodaj następujące informacje do parametrów uruchamiania serwera aplikacji:

```
-Dcom.ibm.mq.connector.IVTMDBCFJNDIName=java:jboss/jms/ivt/IVTCF
```

4. Zatrzymaj i zrestartuj serwer aplikacji.
5. Wdróż aplikację IVT za pomocą konsoli zarządzania JBoss lub interfejsu wiersza komend zarządzania.
6. Uruchom aplikację IVT.

   Więcej informacji na ten temat zawiera sekcja “Program testowy weryfikujący instalację dla adaptera zasobów produktu WebSphere MQ” na stronie 803. W przypadku produktu JBossdomyślnym adresem URL jest `http://localhost:8080/WMQ_IVT/`.

   Uwaga: Nazwy JNDI używane na potrzeby zasobów JMS wymaganych do uruchamiania aplikacji IVT są następujące:

```
Connection Factory : IVTCF
JNDI name          : java:jboss/jms/ivt/IVTCF

Destination        : IVTQueue
JNDI name          : java:jboss/jms/ivt/IVTQueue
```

   Wprowadź nazwy JNDI tych zasobów w odpowiednich polach, gdy aplikacja IVT zostanie uruchomiona przy użyciu podanego powyżej adresu URL, a następnie uruchom aplikację, klikając opcję **Uruchom program IVT**.

Skrypty dostarczane z klasami produktu WebSphere MQ dla usługi JMS

Udostępniono wiele skryptów ułatwiających wykonywanie typowych zadań, które należy wykonać podczas korzystania z klas WebSphere MQ dla usługi JMS.

Tabela 102 na stronie 815 zawiera listę wszystkich skryptów i ich zastosowania. Skrypty znajdują się w podkatalogu `bin` katalogu instalacyjnego produktu WebSphere MQ dla katalogu instalacyjnego JMS.

Tabela 102. Skrypty dostarczane z klasami produktu WebSphere MQ dla usługi JMS

Użyteczność	**Użyj**
Czyszczenie[1]	Ten skrypt jest obsługiwany w celu zachowania zgodności z poprzednimi wersjami, ale nie wykonuje żadnej funkcji. Ręczne czyszczenie informacji o subskrypcji nie jest już potrzebne
DefaultConfiguration	Służy do uruchamiania domyślnej aplikacji konfiguracyjnej na platformach innych niż Windows.
formatLog[1]	Ten skrypt jest obsługiwany w celu zachowania zgodności z poprzednimi wersjami, ale nie wykonuje żadnej funkcji. Dane wyjściowe dziennika są teraz tworzone w postaci czytelnego tekstu.

Tabela 102. Skrypty dostarczane z klasami produktu WebSphere MQ dla usługi JMS (kontynuacja)	
Użyteczność	**Użyj**
IVTRun[1] IVTSetup[1] IVTTidy[1]	Używany w teście weryfikowania instalacji punkt z punktem, zgodnie z opisem w sekcji “Test weryfikujący instalację punkt-punkt dla klas WebSphere MQ dla usługi JMS” na stronie 796.
JMSAdmin[1]	Uruchamia narzędzie administracyjne JMS produktu WebSphere MQ , zgodnie z opisem w sekcji “Wywoływanie narzędzia administracyjnego IBM WebSphere MQ classes for JMS” na stronie 960.
JMSAdmin.config	Plik konfiguracyjny dla narzędzia administracyjnego WebSphere MQ JMS, zgodnie z opisem w sekcji “Konfigurowanie narzędzia administracyjnego JMS” na stronie 961.
PSIVTRun[1]	Uruchamia program testowy sprawdzający instalację publikowania/subskrypcji zgodnie z opisem w sekcji “Test weryfikujący instalację publikowania/subskrypcji dla klas WebSphere MQ dla usługi JMS” na stronie 800.
PSReportDump.class	Ta klasa jest obsługiwana w celu zachowania kompatybilności z poprzednimi wersjami, ale nie wykonuje żadnej funkcji.
setjmsenv	Ustawia zmienne środowiskowe na potrzeby uruchamiania klas produktu WebSphere MQ dla aplikacji JMS w 32-bitowej wirtualnej maszynie języka Java (JVM) w systemach UNIX and Linux , zgodnie z opisem w sekcji “Zmienne środowiskowe używane przez klasy produktu IBM WebSphere MQ dla usługi JMS” na stronie 741.
setjmsenv64	Ustawia zmienne środowiskowe na potrzeby uruchamiania klas WebSphere MQ dla aplikacji JMS w 64-bitową maszynę JVM w systemach UNIX and Linux , zgodnie z opisem w sekcji “Zmienne środowiskowe używane przez klasy produktu IBM WebSphere MQ dla usługi JMS” na stronie 741.
Uwaga: 1. W systemie Windowsnazwa pliku zawiera rozszerzenie .bat.	

Obsługa środowiska OSGi

Środowisko OSGi udostępnia środowisko, które obsługuje wdrażanie aplikacji w postaci pakunków. W ramach produktu IBM WebSphere MQ classes for JMS dostarczane są dziewięć pakunków OSGi.

Środowisko OSGi udostępnia ogólne, bezpieczne i zarządzane środowisko produktuJava , które obsługuje wdrażanie aplikacji, które wchodzą w postaci pakunków. Urządzenia zgodne ze środowiskiem OSGi mogą pobierać i instalować pakunki, a także usuwać je, gdy nie są już wymagane. Środowisko zarządza instalacją i aktualizacją pakunków w sposób dynamiczny i skalowalny.

IBM WebSphere MQ classes for JMS. zawiera następujące pakunki OSGi.

com.ibm.msg.client.osgi.jms< numer wersji > .jar
Wspólna warstwa kodu w IBM WebSphere MQ classes for JMS. Informacje na temat architektury warstwowej klas produktu WebSphere MQ dla usługi JMS zawiera sekcja “Architektura warstwowa” na stronie 820.

com.ibm.msg.client.osgi.jms.prereq_< numer wersji > .jar
Wstępnie wymagane pliki archiwum produktu Java (JAR) dla wspólnej warstwy.

com.ibm.msg.client.osgi.commonservices.j2se_<version numer > .jar
Wspólne usługi dla aplikacji Java Platform, Standard Edition (Java SE).

com.ibm.msg.client.osgi.nls_< numer wersji > .jar
Komunikaty dla warstwy wspólnej.

com.ibm.msg.client.osgi.wmq_< numer wersji > .jar
Dostawca przesyłania komunikatów produktu IBM WebSphere MQ w produkcie IBM WebSphere MQ classes for JMS. Informacje na temat architektury warstwowej produktu IBM WebSphere MQ classes for JMS można znaleźć w sekcji “Architektura warstwowa” na stronie 820.

com.ibm.msg.client.osgi.wmq.prereq_< numer wersji > .jar
Wstępnie wymagane pliki JAR dla dostawcy przesyłania komunikatów produktu IBM WebSphere MQ .

com.ibm.msg.client.osgi.wmq.nls_< numer wersji > .jar
Komunikaty dla dostawcy przesyłania komunikatów produktu IBM WebSphere MQ .

com.ibm.mq.osgi.directip_< numer wersji > .jar
Pliki JAR, które umożliwiają dostawcy przesyłania komunikatów produktu IBM WebSphere MQ tworzenie połączenia w czasie rzeczywistym z brokerem.

gdzie < numer wersji > jest numerem wersji produktu WebSphere MQ , który został zainstalowany.

Pakunki są instalowane w podkatalogu `java/lib/OSGi` instalacji produktu WebSphere MQ lub w folderze `java\lib\OSGi` w systemie Windows.

Pakunek com.ibm.mq.osgi.java < numer wersji > .jar, który jest również instalowany w podkatalogu `java/lib/OSGi` instalacji produktu WebSphere MQ lub w folderze `java\lib\OSGi` w systemie Windows, jest częścią klas produktu WebSphere MQ dla języka Java. Ten pakunek nie może być ładowany do środowiska wykonawczego OSGi, które zawiera klasy WebSphere MQ dla ładowania JMS.

Pakunki OSGi dla klas WebSphere MQ classes for JMS zostały zapisane w specyfikacji OSGi Release 4. Nie działają one w środowisku OSGi Release 3.

Należy poprawnie ustawić ścieżkę systemową lub ścieżkę do biblioteki, aby środowisko wykonawcze OSGi było w stanie znaleźć wszystkie wymagane pliki DLL lub współużytkowane biblioteki.

Jeśli pakunki OSGi są używane na potrzeby produktu IBM WebSphere MQ classes for JMS, tematy tymczasowe nie działają. Ponadto klasy wyjścia kanału zapisane w produkcie Java nie są obsługiwane z powodu nieodłącznego problemu związanego z ładowaniem klas w środowisku programu ładującego wiele klas, takim jak środowisko OSGi. Pakunek użytkownika może być świadomy klas IBM WebSphere MQ dla pakunków JMS, ale pakunki IBM WebSphere MQ classes for JMS nie są znane w żadnym pakunku użytkownika. W wyniku tego program ładujący klasy używany w pakunku IBM WebSphere MQ classes for JMS nie może załadować klasy wyjścia kanału, która znajduje się w pakunku użytkownika.

Więcej informacji na temat środowiska OSGi można znaleźć w serwisie WWW OSGi Alliance .

Rozwiązywanie problemów z klasami produktu IBM WebSphere MQ dla usługi JMS

Problemy można śledzić, uruchamiając programy weryfikujące instalację i korzystając z narzędzi śledzenia i dziennika.

Jeśli program nie zakończy się pomyślnie, uruchom jeden z programów do weryfikacji instalacji zgodnie z opisem w sekcji “Test weryfikujący instalację punkt-punkt dla klas WebSphere MQ dla usługi JMS” na stronie 796 i “Test weryfikujący instalację publikowania/subskrypcji dla klas WebSphere MQ dla usługi JMS” na stronie 800, a następnie postępuj zgodnie z zaleceniami podanymi w komunikatach diagnostycznych.

Rejestrowanie i IBM WebSphere MQ classes for JMS

Domyślnie dane wyjściowe dziennika są wysyłane do pliku `mqjms.log` . Można ją przekierować do określonego pliku lub katalogu.

Narzędzie rejestrowania produktu IBM WebSphere MQ classes for JMS jest udostępniane do zgłaszania poważnych problemów, szczególnie problemów, które mogą wskazywać na błędy konfiguracji, a nie błędy programowania. Domyślnie dane wyjściowe dziennika są wysyłane do pliku `mqjms.log` w katalogu roboczym maszyny JVM.

Dane wyjściowe dziennika można przekierować do innego pliku, ustawiając właściwość com.ibm.msg.client.commonservices.log.outputName. Wartość tej właściwości może być następująca:

- Pojedyncza nazwa ścieżki.
- Rozdzielana przecinkami lista nazw ścieżek (wszystkie dane są rejestrowane we wszystkich plikach).

Każda nazwa ścieżki może być następująca:

- Wartość bezwzględna lub względna.
- `stderr` lub `System.err` do reprezentowania standardowego strumienia błędów.
- `sttdout` lub `System.out` do reprezentowania standardowego strumienia wyjściowego.

Jeśli wartość właściwości identyfikuje katalog, dane wyjściowe dziennika są zapisywane w katalogu `mqjms.log` w tym katalogu. Jeśli wartość właściwości identyfikuje konkretny plik, dane wyjściowe dziennika są zapisywane do tego pliku.

Tę właściwość można ustawić w pliku konfiguracyjnym IBM WebSphere MQ classes for JMS lub jako właściwość systemową w komendzie **java** . W poniższym przykładzie właściwość jest ustawiana jako właściwość systemowa i identyfikuje konkretny plik:

```
java -Djava.library.path=library_path
     -Dcom.ibm.msg.client.commonservices.log.outputName=/mydir/mylog.txt
      MyAppClass
```

W komendzie *ścieżka_biblioteki* jest ścieżką do katalogu zawierającego biblioteki produktu IBM WebSphere MQ classes for JMS (patrz sekcja “Konfigurowanie bibliotek Java Native Interface (JNI)” na stronie 743).

Dane wyjściowe dziennika można wyłączyć, ustawiając wartość właściwości com.ibm.msg.client.commonservices.log.status na OFF. Wartość domyślna tej właściwości to ON.

Wartości `System.err` i `System.out` mogą być ustawione tak, aby dane wyjściowe dziennika były wysyłane do strumieni `System.err` i `System.out` .

Wprowadzenie do klas WebSphere MQ dla usługi JMS, dla programistów

Klasy WebSphere MQ classes for JMS są dostawcą JMS dostarczonym z produktem WebSphere MQ. Klasy WebSphere MQ classes for JMS implementują interfejsy zdefiniowane w pakiecie javax.jms , a także udostępniają dwa zestawy rozszerzeń do interfejsu API JMS. Aplikacje Java Platform, Standard Edition (Java SE) i Java Platform, Enterprise Edition (Java EE) mogą używać klas produktu WebSphere MQ dla usługi JMS.

Specyfikacja JMS definiuje zestaw interfejsów, których aplikacje mogą używać do wykonywania operacji przesyłania komunikatów. Najnowsza wersja specyfikacji to wersja 1.1. Pakiet javax.jms określa szczegółowe informacje na temat interfejsów JMS, a dostawca JMS implementuje te interfejsy dla konkretnego produktu przesyłania komunikatów. Klasy WebSphere MQ classes for JMS są dostawcą JMS, który implementuje interfejsy JMS dla produktu WebSphere MQ.

Przepływ logiki w ramach aplikacji JMS rozpoczyna się od obiektu ConnectionFactory i obiektów docelowych. Aplikacja korzysta z obiektu ConnectionFactory w celu utworzenia obiektu połączenia, który reprezentuje aktywne połączenie z aplikacji na serwer przesyłania komunikatów. Aplikacja korzysta z obiektu połączenia w celu utworzenia obiektu sesji, który jest pojedynczym kontekstem wielowątkowym służącym do tworzenia i konsumowania komunikatów. Aplikacja może następnie użyć obiektu Session i obiektu docelowego w celu utworzenia obiektu MessageProducer , którego aplikacja używa do wysyłania komunikatów do określonego miejsca docelowego. Miejsce docelowe jest kolejką lub tematem w systemie przesyłania komunikatów i jest hermetyzowane przez obiekt docelowy. Aplikacja może również użyć obiektu Session i obiektu docelowego w celu utworzenia obiektu MessageConsumer , którego aplikacja używa do odbierania komunikatów, które zostały wysłane do określonego miejsca docelowego.

Specyfikacja JMS oczekuje, że obiekty ConnectionFactory i Destination mają być administrowane obiektami. Administrator tworzy i obsługuje administrowane obiekty w centralnym repozytorium, a aplikacja JMS pobiera te obiekty przy użyciu interfejsu JNDI (Java Naming and Directory Interface).

Repozytorium administrowanych obiektów może być używane z prostego pliku do katalogu LDAP (Lightweight Directory Access Protocol).

Klasy WebSphere MQ classes for JMS obsługują korzystanie z administrowanych obiektów. Aplikacja może korzystać ze wszystkich funkcji produktu WebSphere MQ , które są ujawniane za pośrednictwem klas WebSphere MQ dla usługi JMS bez konieczności posiadania jakichkolwiek informacji specyficznych dla produktu WebSphere MQzakodowanych w samej aplikacji. Ten układ udostępnia aplikację o stopniu niezależności od bazowej konfiguracji produktu WebSphere MQ . Aby uzyskać tę niezależność, aplikacja może używać interfejsu JNDI do pobierania fabryk połączeń i miejsc docelowych, które są przechowywane jako obiekty administrowane, i używać tylko interfejsów zdefiniowanych w pakiecie javax.jms w celu wykonywania operacji przesyłania komunikatów. Administrator może za pomocą narzędzia administracyjnego WebSphere MQ JMS lub programu IBM WebSphere MQ Explorer utworzyć i obsługiwać administrowane obiekty w centralnym repozytorium. Jednak serwer aplikacji zwykle udostępnia własne repozytorium dla administrowanych obiektów oraz własnych narzędzi do tworzenia i obsługi obiektów. Aplikacja Java EE może zatem używać interfejsu JNDI do pobierania administrowanych obiektów z repozytorium serwera aplikacji lub z repozytorium centralnego.

Klasy WebSphere MQ classes for JMS udostępniają również rozszerzenia interfejsu API JMS. Poprzednie wersje klas produktu WebSphere MQ dla usługi JMS zawierają rozszerzenia implementowane w obiektach MQConnectionFactory, MQQueue i MQTopic. Te obiekty mają właściwości i metody specyficzne dla produktu WebSphere MQ. Obiektami mogą być obiekty administrowane, a aplikacja może tworzyć obiekty dynamicznie w czasie wykonywania. Ta wersja klas produktu WebSphere MQ dla usługi JMS utrzymuje te rozszerzenia, a użytkownik może nadal używać, bez zmian, żadnych aplikacji, które używają tych rozszerzeń. Rozszerzenia te są znane jako *Rozszerzenia WebSphere MQ JMS*. Należy pamiętać, że w tym zestawie dokumentacji obiekty tworzone dynamicznie przez aplikację w czasie wykonywania *nie* są traktowane jako obiekty administrowane.

Oprócz rozszerzeń JMS produktu WebSphere MQ ta wersja klas WebSphere MQ dla usługi JMS udostępnia bardziej ogólny zestaw rozszerzeń do interfejsu API JMS. Rozszerzenia te są znane jako *Rozszerzenia IBM JMS*i mają następujące ogólne cele:

- Zapewnienie większego poziomu spójności między dostawcami JMS IBM
- Aby ułatwić zapis aplikacji pomostowej między dwoma systemami przesyłania komunikatów IBM
- Aby ułatwić port aplikacji od jednego dostawcy IBM JMS do innego

Główne uwagi dotyczące tych rozszerzeń dotyczą tworzenia i konfigurowania fabryk połączeń i miejsc docelowych dynamicznie w czasie wykonywania, ale rozszerzenia udostępniają również funkcje, które nie są bezpośrednio związane z przesyłaniem komunikatów, takie jak funkcja określania problemu.

Fabryka połączeń, kolejka lub obiekt tematu utworzony przy użyciu interfejsu javax.jms lub zestawu rozszerzeń JMS można adresować przy użyciu dowolnego z tych interfejsów APIs;, które można rzutować na dowolny z interfejsów. Aby zachować przenośność aplikacji na najwyższym poziomie, należy użyć najbardziej ogólnego interfejsu API, który jest odpowiedni dla wymagań użytkownika.

Zarówno aplikacje Java SE, jak i aplikacje Java EE mogą używać klas WebSphere MQ dla usługi JMS. Na platformie Java EE klasy produktu WebSphere MQ classes for JMS obsługują dwa typy komunikacji między komponentem aplikacji a menedżerem kolejek produktu WebSphere MQ :

Komunikacja wychodząca
Bezpośrednio za pomocą interfejsu API JMS komponent aplikacji tworzy połączenie z menedżerem kolejek, a następnie wysyła i odbiera komunikaty.

Komponentem aplikacji może być na przykład klient aplikacji, serwlet, strona JSP (JavaServer Page), komponent EJB (Enterprise Java Bean) lub komponent bean sterowany komunikatami (message driven bean-MDB). W tym typie komunikacji kontener serwera aplikacji udostępnia tylko funkcje niskiego poziomu wspierające operacje przesyłania komunikatów, takie jak zestawianie połączeń i zarządzanie wątkami.

Komunikacja przychodząca
Komunikat przylot do miejsca docelowego jest dostarczany do komponentu MDB, który następnie przetwarza ten komunikat.

Aplikacje Java EE używają komponentów MDB do asynchronicznej przetwarzania komunikatów. Komponent MDB działa jako obiekt nasłuchiwania komunikatów JMS i jest implementowany za pomocą metody onMessage(), która definiuje sposób przetwarzania komunikatu. Komponent MDB jest wdrażany w kontenerze EJB serwera aplikacji. Dokładny sposób konfigurowania komponentu MDB zależy od używanego serwera aplikacji, ale informacje o konfiguracji muszą określać, który menedżer kolejek ma nawiązać połączenie, jak nawiązać połączenie z menedżerem kolejek, którego miejsce docelowe ma być monitorowane pod kątem komunikatów, a także zachowanie transakcyjne komponentu MDB. Informacje te są następnie używane przez kontener EJB. Gdy komunikat spełniający kryteria wyboru komponentu MDB nadchodzi do określonego miejsca docelowego, kontener EJB korzysta z klas WebSphere MQ classes for JMS w celu pobrania komunikatu z menedżera kolejek, a następnie dostarcza komunikat do komponentu MDB, wywołując jego metodę onMessage().

Klasy produktu IBM WebSphere MQ dla architektury JMS

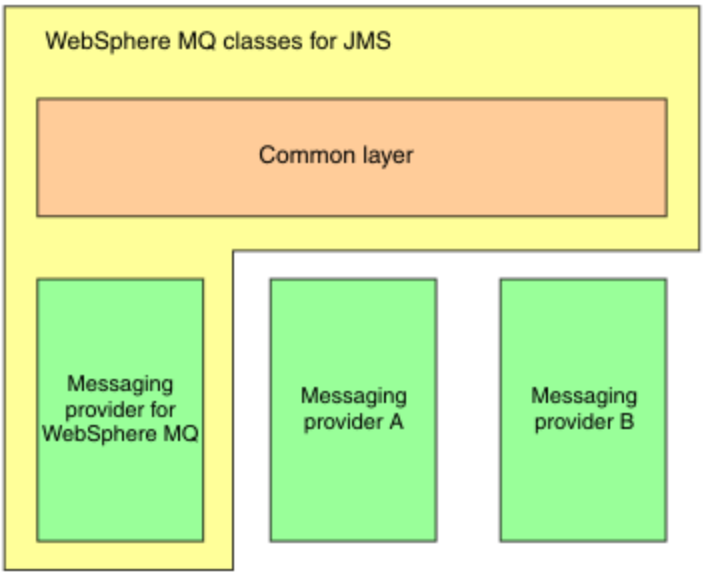
Klasy produktu IBM WebSphere MQ dla usługi JMS dostarczane w produkcie IBM WebSphere MQ w wersji 7.0, a następnie w kolejnych wersjach, zawierają szereg udoskonaleń w porównaniu z poprzednimi wersjami. Niektóre z tych udoskonaleń są wynikiem zmian w implementacji klas IBM WebSphere MQ dla usługi JMS, a niektóre z nich są wynikiem zmian w klasach IBM WebSphere MQ dla JMS wykorzystujących zmiany w bazowej funkcji IBM WebSphere MQ .

W poniższych sekcjach zostały podsumowane kluczowe udoskonalenia.

Architektura warstwowa

We wcześniejszych wersjach produktu WebSphere MQimplementacja klas produktu WebSphere MQ dla usługi JMS jest całkowicie specyficzna dla produktu WebSphere MQ. Inne produkty IBM , które udostępniają systemy przesyłania komunikatów, zawierają również dostawców JMS, ale te dostawcy JMS mają bardzo niewiele lub nic wspólnego z implementacją klas produktu WebSphere MQ dla usługi JMS.

W przypadku produktu WebSphere MQ V7.0, klasy produktu WebSphere MQ dla usługi JMS mają architekturę warstwową. Górna warstwa kodu jest wspólną warstwą, która może być używana przez dowolny dostawca IBM JMS. Gdy aplikacja wywołuje metodę JMS, każde przetwarzanie wywołania, które nie jest specyficzne dla systemu przesyłania komunikatów, jest wykonywane przez wspólną warstwę, która również zapewnia spójną odpowiedź na wywołanie. Każde przetwarzanie wywołania, które jest specyficzne dla systemu przesyłania komunikatów, jest delegowane do niższej warstwy. Rysunek 125 na stronie 820 przedstawia architekturę warstwową.



Rysunek 125. Architektura warstwowa dla dostawców JMS IBM

Przejście do architektury warstwowej ma następujące cele:

- W celu zwiększenia spójności działania różnych dostawców JMS IBM
- Aby ułatwić zapis aplikacji pomostowej między dwoma systemami przesyłania komunikatów IBM
- Aby ułatwić port aplikacji od jednego dostawcy IBM JMS do innego

Ta implementacja klas produktu WebSphere MQ dla usługi JMS wprowadza również nowy zestaw rozszerzeń do interfejsu API JMS. Rozszerzenia te są znane jako *Rozszerzenia IBM JMS*. Główne uwagi dotyczące tych rozszerzeń dotyczą tworzenia i konfigurowania fabryk połączeń i miejsc docelowych dynamicznie w czasie wykonywania.

Aplikacja używała rozszerzenia IBM JMS rozpoczyna się od utworzenia obiektu fabryki JmsFactory, określając jako parametr stałą, która identyfikuje wybrany system przesyłania komunikatów. Aplikacja korzysta z obiektu fabryki JmsFactoryw celu utworzenia fabryk połączeń i miejsc docelowych, które mają poprawne klasy specjalne dla wybranego systemu przesyłania komunikatów.

Aplikacja może następnie skonfigurować fabryki połączeń i miejsca docelowe, ustawiając ich właściwości. Rozszerzenia JMS produktu IBM udostępniają zestaw metod do ustawiania właściwości. Te metody są niezależne od dowolnego systemu przesyłania komunikatów. Każdy typ danych ma własną metodę set, a każda właściwość jest identyfikowana za pomocą nazwy, która jest zdefiniowana jako statyczny element końcowy klasy WMQConstants. Gdy aplikacja wywołuje jedną z tych metod, jednym z parametrów w wywołaniu jest nazwa właściwości, a drugi parametr jest wartością właściwości.

Na przykład, jeśli produkt WebSphere MQ jest systemem przesyłania komunikatów, jedną z właściwości fabryki połączeń jest nazwa menedżera kolejek, z którym ma zostać nawiązane połączenie. Za pomocą rozszerzeń JMS IBM aplikacja ustawia nazwę menedżera kolejek na JUPITER, wywołując następującą metodę:

```
JmsConnectionFactory myCF;
...
myCF.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "JUPITER");
```

Natomiast aplikacja może wykonać tę samą funkcję, wywołując następującą metodę:

```
MQConnectionFactory myCF;
...
myCF.setQueueManager("JUPITER");
```

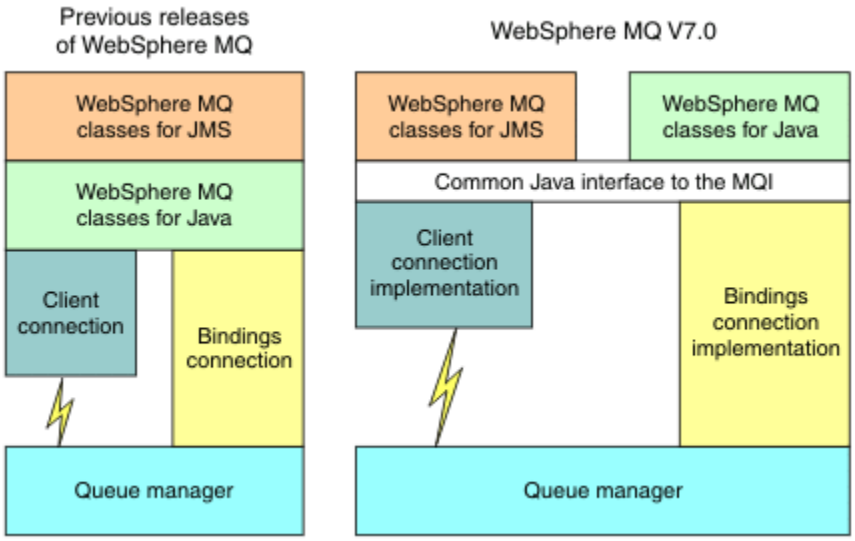
Ta metoda jest rozszerzeniem JMS produktu WebSphere MQ i jest specyficzne dla produktu WebSphere MQ jako systemu przesyłania komunikatów. Użycie tej metody sprawia, że aplikacja jest potencjalnie mniej łatwa do portu dla innego dostawcy JMS IBM .

Relacja między klasami produktu WebSphere MQ classes for JMS a klasami produktu WebSphere MQ dla języka Java

W wersjach produktu WebSphere MQ, wcześniejszych niż wersja 7.0, klasy produktu WebSphere MQ dla usługi JMS zostały zaimplementowane niemal w całości jako warstwa kodu w klasach produktu WebSphere MQ dla języka Java. To rozwiązanie spowodowało pewne zamieszanie wśród twórców aplikacji, ponieważ ustawienie pól lub metod wywoływania w klasie MQEnvironment może spowodować niepożądane i nieoczekiwane skutki działania kodu napisanego przy użyciu klas WebSphere MQ dla usługi JMS. Ponadto implementacja klas produktu WebSphere MQ dla usługi JMS miała pewne ograniczenia w obszarach, w których interfejs API JMS nie jest naturalnym pasem w górnej części klas produktu WebSphere MQ dla języka Java, a te ograniczenia doprowadziły do niektórych problemów dotyczących wydajności środowiska wykonawczego.

W produkcie WebSphere MQ V7.0implementacja klas produktu WebSphere MQ dla usługi JMS nie jest już zależna od klas produktu WebSphere MQ dla języka Java. Klasy WebSphere MQ classes for Java i klasy WebSphere MQ classes for JMS są teraz równorzędnymi interfejsami, które używają wspólnego interfejsu Java do interfejsu MQI. Takie rozwiązanie pozwala na większą optymalizację wydajności i oznacza, że ustawienie pól lub metod wywoływania w klasie MQEnvironment nie ma wpływu na zachowanie kodu w czasie wykonywania, które jest napisane przy użyciu klas WebSphere MQ dla usługi JMS. Rysunek 126 na stronie 822 Pokazuje relację między klasami produktu WebSphere MQ dla klas JMS i klas WebSphere

MQ dla języka Java w poprzednich wersjach produktu WebSphere MQ i w produkcie WebSphere MQ V7.0 i kolejnych wersjach.



Rysunek 126. Relacja między klasami produktu WebSphere MQ classes for JMS a klasami produktu WebSphere MQ dla języka Java

Aby zachować kompatybilność z wcześniejszymi wersjami, klasy wyjścia kanału napisane w języku Java mogą nadal używać klas produktu WebSphere MQ dla interfejsów Java, nawet jeśli klasy wyjścia kanału są wywoływane z klas produktu WebSphere MQ dla usługi JMS. Jednak użycie klas WebSphere MQ dla interfejsów Java oznacza, że aplikacje nadal są zależne od klas produktu WebSphere MQ dla pliku JAR Java (com.ibm.mq.jar). Jeśli plik com.ibm.mq.jar nie ma być w ścieżce klasy, można zamiast niego użyć nowego zestawu interfejsów w pakiecie com.ibm.mq.exits .

Teraz można tworzyć i konfigurować obiekty administrowane JMS przy użyciu programu WebSphere MQ Explorer.

Przesyłanie komunikatów publikowania/subskrypcji

Produkt WebSphere MQ V7.0i kolejne wydania zawierają wbudowaną funkcję publikowania/subskrypcji. Ta funkcja zastępuje produkt WebSphere MQ Publish/Subscribe, który został dostarczony z produktem WebSphere MQ V6.0.

Klasy produktu WebSphere MQ dla aplikacji JMS mogą korzystać z wbudowanej funkcji publikowania/subskrypcji i mogą z niej korzystać zamiast WebSphere Event Broker lub WebSphere Message Broker do przesyłania komunikatów w trybie publikowania/subskrypcji z produktem WebSphere MQ jako transportem. Skonfigurowanie klas WebSphere MQ dla usługi JMS pod kątem używania nowej funkcji jest prostsze niż konfigurowanie klas WebSphere MQ dla usługi JMS w celu używania produktu WebSphere MQ Publish/Subscribe, WebSphere Event Broker lub WebSphere Message Broker. Administratorzy i programiści aplikacji nie muszą już zarządzać kolejkami publikacji, kolejkami subskrybentów, składnicami subskrypcji i czyszczeniem subskrybenta. Ponadto obiekt ConnectionFactory i obiekty tematów mają mniejszą liczbę właściwości.

Wbudowana funkcja publikowania/subskrypcji udostępnia również dodatkowe funkcje, takie jak zachowane publikacje i wybór dwóch schematów wieloznacznych w celu określenia zakresu tematów, do których aplikacja chce się subskrybować.

Aplikacja nadal może korzystać z połączenia w czasie rzeczywistym z brokerem produktu WebSphere Event Broker lub WebSphere Message Broker na potrzeby przesyłania komunikatów w trybie publikowania/subskrypcji. Ta obsługa jest niezmieniona.

Aplikacje korzystające z produktu WebSphere MQ Publish/Subscribe mogą korzystać z wbudowanej funkcji publikowania/subskrypcji bez zmiany, gdy menedżer kolejek, do którego są połączone, jest

aktualizowany. Właściwości, które są ustawiane przez aplikację, ale nie są wymagane przez wbudowaną funkcję publikowania/subskrypcji, są ignorowane.

Dostawca przesyłania komunikatów produktu WebSphere MQ

Dostawca przesyłania komunikatów produktu WebSphere MQ ma dwa tryby działania:

- *Tryb normalny dostawcy przesyłania komunikatów produktu WebSphere MQ*
- *Tryb migracji dostawcy przesyłania komunikatów produktu WebSphere MQ*

Tryb normalny dostawcy przesyłania komunikatów produktu WebSphere MQ używa wszystkich funkcji produktu WebSphere MQ w wersji 7.0, a następnie menedżery kolejek wersji w celu zaimplementowania usługi JMS. Ten tryb jest używany tylko do łączenia się z menedżerem kolejek produktu WebSphere MQ i może łączyć się z produktem WebSphere MQ w wersji 7.0, a następnie menedżerami kolejek wersji w trybie klienta lub powiązania. Ten tryb jest zoptymalizowany pod kątem używania nowej wersji produktu WebSphere MQ w wersji 7.0 i późniejszej wersji.

Tryb migracji dostawcy przesyłania komunikatów produktu WebSphere MQ jest oparty na funkcji produktu WebSphere MQ w wersji 6.0 i korzysta tylko z funkcji dostępnych w menedżerze kolejek produktu WebSphere MQ w wersji 6.0 w celu zaimplementowania usługi JMS. You can connect to a WebSphere MQ Version 7.0 and subsequent release queue managers using WebSphere MQ messaging provider migration mode but you cannot use any of the Version 7.0 optimizations. Ten tryb umożliwia nawiązanie połączeń z jedną z następujących wersji menedżera kolejek:

1. WebSphere MQ , wersja 7.0, i następnie menedżer kolejek w powiązaniach lub trybie klienta, ale w tym trybie są używane tylko te funkcje, które były dostępne dla menedżera kolejek produktu WebSphere MQ w wersji 6.0 .
2. Produkt WebSphere MQ w wersji 6.0 lub starszy menedżer kolejek w trybie klienta

Aby nawiązać połączenie z produktem WebSphere Event Broker lub WebSphere Message Broker przy użyciu produktu WebSphere MQ Enterprise Transport, należy użyć trybu migracji dostawcy przesyłania komunikatów produktu WebSphere MQ . Jeśli używany jest produkt WebSphere MQ Real-Time Transport, automatycznie wybierany jest tryb migracji dostawcy przesyłania komunikatów produktu WebSphere MQ , ponieważ w obiekcie fabryki połączeń zostały jawnie wybrane właściwości. Połączenie z brokerem zdarzeń WebSphere lub produktem WebSphere Message Broker przy użyciu produktu WebSphere MQ Enterprise Transport jest zgodne z ogólnymi regułami wyboru trybu opisanymi w sekcji Reguły wyboru trybu dostawcy przesyłania komunikatów produktu WebSphere MQ .

Asynchroniczne wykorzystanie komunikatów

Produkt WebSphere MQ V7.0 i wszystkie kolejne wydania obsługują asynchroniczne wykorzystanie komunikatów. Aplikacja może zarejestrować funkcję zwrotną dla miejsca docelowego. Gdy do miejsca docelowego zostanie wysłany odpowiedni komunikat, program WebSphere MQ wywołuje funkcję i przekazuje komunikat jako parametr. Funkcja następnie przetwarza komunikat asynchronicznie. W poprzednich wersjach produktu WebSphere MQten składnik był dostępny tylko wtedy, gdy używane są klasy produktu WebSphere MQ dla usługi JMS.

Klasy produktu WebSphere MQ dla usługi JMS zostały zmienione w celu wykorzystania tej nowej funkcji w produkcie WebSphere MQ V7.0 i w kolejnych wydaniach. Implementacja obiektów nasłuchiwania komunikatów JMS jest teraz bardziej naturalna w przypadku klas WebSphere MQ, a klasy WebSphere MQ dla usługi JMS nie muszą już odpytywać miejsca docelowego w celu sprawdzenia, czy do miejsca docelowego został wysłany odpowiedni komunikat. Wydajność programów nasłuchujących komunikatów JMS została poprawiona w wyniku, szczególnie jeśli aplikacja korzysta z wielu obiektów nasłuchiwania komunikatów w sesji w celu monitorowania wielu miejsc docelowych. Przepustowość komunikatów jest zwiększana, a czas dostarczenia komunikatu do obiektu nasłuchiwania komunikatów po jego dotarciu do miejsca docelowego jest zmniejszony.

Komponenty bean sterowane komunikatami (MDB) mają podobną poprawę wydajności. Dodatkowo, z powodu innego rozszerzenia funkcji produktu WebSphere MQ , wiele komponentów MDB, które

konsumują komunikaty z tego samego miejsca docelowego, teraz doświadcza zredukowanej rywalizacji w komunikatach.

Wybór komunikatów

Z wyjątkiem wybierania komunikatów według identyfikatora komunikatu lub identyfikatora korelacji, wszystkie opcje wyboru komunikatów w wersjach produktu WebSphere MQ wcześniejszych niż wersja 7.0 były wykonywane przez klasy produktu WebSphere MQ dla usługi JMS. W produkcie WebSphere MQ V7.0, a także w kolejnych wydaniach, wszystkie opcje wyboru komunikatów są wykonywane przez menedżer kolejek.

W rezultacie przepustowość komunikatów jest zwiększana dla aplikacji, które korzystają z komunikatów przy użyciu wyboru komunikatów. Poprawa wydajności jest większa dla aplikacji, która łączy się w trybie klienta, ponieważ tylko te komunikaty, które spełniają kryteria wyboru, są transportowane przez sieć, a klasy WebSphere MQ dla usługi JMS obsługują tylko te komunikaty, które dostarcza do aplikacji.

Współużytkowanie połączenia komunikacyjnego

W poprzednich wersjach produktu WebSphere MQ, jeśli aplikacja kliencka WebSphere MQ jest połączona z menedżerem kolejek więcej niż jeden raz przy użyciu tego samego kanału MQI, każda instancja kanału MQI wymagała oddzielnego połączenia TCP. W produkcie WebSphere MQ V7.0 i w dowolnej kolejnej wersji każde połączenie z menedżerem kolejek przy użyciu tego samego kanału MQI może współużytkować pojedyncze połączenie TCP. Takie rozwiązanie oznacza, że wymagane jest zmniejszenie liczby zasobów sieciowych, a łączny czas potrzebny na utworzenie wielu połączeń z menedżerem kolejek jest zmniejszony, szczególnie w przypadku korzystania z protokołu SSL, ponieważ uzgadnianie SSL odbywa się tylko raz na początku połączenia TCP.

Klasy WebSphere MQ classes for JMS wykorzystują to udoskonalenie. W przypadku aplikacji, która łączy się z menedżerem kolejek w trybie klienta, klasy WebSphere MQ classes for JMS mogą tworzyć więcej niż jedno połączenie z menedżerem kolejek przy użyciu kanału MQI o nazwie określonej jako właściwość obiektu ConnectionFactory . Każdy z tych połączeń z menedżerem kolejek może teraz współużytkować pojedyncze połączenie TCP.

Odczyt z wyprzedzeniem na połączeniach klienckich

Jeśli aplikacja korzysta z połączenia klienckiego w celu konsumowania komunikatów nietrwałych z miejsca docelowego, miejsce docelowe można skonfigurować w taki sposób, aby klasy WebSphere MQ classes for JMS używały buforu do przechowywania wiadomości będących przedmiotem zainteresowania przed dostarczeniem ich do aplikacji. Ta optymalizacja jest nazywana *odczytu z wyprzedzeniem* i może być używana przez aplikacje, które odbierają komunikaty synchronicznie przez wywołanie metody receive (), a także przez programy nasłuchujące komunikatów i komponenty MDB, które wykorzystują komunikaty asynchronicznie. Odczyt z wyprzedzeniem jest szczególnie skuteczny w przypadku miejsc docelowych z dużą liczbą komunikatów, które muszą zostać szybko zużyte.

Odczyt z wyprzedzeniem nie ma zastosowania do komunikatów trwałych, ponieważ jeśli komunikaty trwałe zostały odczytane w buforze, menedżer kolejek nie będzie w stanie odzyskać komunikatów po wystąpieniu awarii. Jednak aplikacja, która konsumuje komunikaty z miejsca docelowego, z mieszaniną komunikatów trwałych i nietrwałych, nadal może używać odczytu z wyprzedzeniem. Kolejność komunikatów jest zachowywana, ale korzyści ze środowiska wykonawczego dla odczytu z wyprzedzeniem mają zastosowanie tylko do nietrwałych komunikatów.

Podejmując decyzję o zastosowaniu odczytu z wyprzedzeniem, należy wziąć pod uwagę następujące kwestie:

- Jeśli aplikacja konsumuje komunikaty z miejsca docelowego, które jest skonfigurowane do odczytu z wyprzedzeniem, a aplikacja kończy działanie z dowolnego powodu, wszystkie nietrwałe komunikaty zapisane w buforze są usuwane.
- Jeśli wszystkie poniższe warunki są spełnione, komunikaty wysłane do kolejki w sesji mogą nie zostać odebrane w kolejności, w jakiej zostały wysłane:

- Aplikacja używa dwóch konsumentów komunikatów w tej samej sesji do korzystania z komunikatów z kolejki.
- Każdy konsument komunikatów korzysta z innego obiektu docelowego dla kolejki.
- Każdy lub oba obiekty docelowe są skonfigurowane do odczytu z wyprzedzeniem.

Wysyłanie komunikatów

Gdy aplikacja wysyła komunikaty do miejsca docelowego, miejsce docelowe można skonfigurować w taki sposób, aby podczas wysyłania wywołań aplikacji (), WebSphere MQ classes for JMS przekazało komunikat do menedżera kolejek i zwracał sterowanie z powrotem do aplikacji bez określania, czy menedżer kolejek odebrał komunikat w sposób bezpieczny. Klasy WebSphere MQ classes for JMS mogą pracować w ten sposób tylko w przypadku komunikatów nietrwałych i dla komunikatów trwałych wysyłanych w sesji transakcyjnych.

W przypadku trwałych komunikatów wysyłanych w sesji transakcyjnych aplikacja ostatecznie określa, czy menedżer kolejek odebrał komunikaty w sposób bezpieczny, gdy wywołuje zatwierdzenie (). W przypadku wszystkich komunikatów wysłanych w sesji, które nie są transakcyjne, właściwość SENDCHECKCOUNT obiektu ConnectionFactory określa liczbę komunikatów, które mają zostać wysłane przed sprawdzeniami JMS przez klasy WebSphere MQ sprawdzające, czy menedżer kolejek odebrał komunikaty w sposób bezpieczny.

Ta optymalizacja jest najbardziej korzystna dla aplikacji, która łączy się z menedżerem kolejek w trybie klienta i wymaga wysyłania sekwencji komunikatów w szybkim dziedziczeniu, ale nie wymaga natychmiastowych informacji zwrotnych od menedżera kolejek dla każdego wysłanego komunikatu.

Wyjścia kanału

W przypadku wywołania z klas WebSphere MQ dla usługi JMS programy obsługi wyjścia kanału napisane w języku C lub C++ zachowują się w taki sam sposób, jak w przypadku wywołania z klienta MQI produktu Websphere MQ . Wydajność klas wyjścia kanału napisanych w języku Java została poprawiona i można teraz zapisywać klasy wyjścia kanału za pomocą nowego zestawu interfejsów w pakiecie com.ibm.mq.exits zamiast używać interfejsów w klasach produktu WebSphere MQ dla języka Java.

Właściwości komunikatu

Komunikat JMS składa się z zestawu pól nagłówka, zestawu właściwości oraz treści, które zawierają dane aplikacji. Komunikat WebSphere MQ (co najmniej) składa się z deskryptora komunikatu i danych aplikacji.

Gdy klasy produktu WebSphere MQ dla aplikacji JMS wysyła komunikat JMS, klasy WebSphere MQ dla JMS odwzorowują komunikat JMS na komunikat WebSphere MQ . Niektóre pola nagłówka i właściwości JMS są odwzorowywane na pola w deskryptorze komunikatu, a niektóre z nich są odwzorowywane na pola w dodatkowym nagłówku WebSphere MQ o nazwie nagłówka MQRFH2 . Gdy klasy produktu WebSphere MQ dla aplikacji JMS odbierają komunikat JMS, klasy WebSphere MQ dla usługi JMS wykonuje odwzorowanie odwrotne.

Aplikacja korzystała z interfejsu MQI w celu odbierania komunikatów z klas produktu WebSphere MQ dla aplikacji JMS musi więc obsługiwać nagłówek MQRFH2 . If the application cannot handle an MQRFH2 header, the TARGCLIENT property of the Destination object can be set to tell WebSphere MQ classes for JMS not to include an MQRFH2 header in the WebSphere MQ messages. Jednak, wykluczając nagłówek MQRFH2 , informacje przechowywane w niektórych polach nagłówka JMS i właściwościach są tracone.

Podobnie aplikacja, która używa interfejsu MQI w celu wysyłania komunikatów do klas WebSphere MQ dla aplikacji JMS, musi zawierać nagłówek MQRFH2 w każdym komunikacie. Jeśli nagłówek MQRFH2 nie jest uwzględniony, klasy WebSphere MQ dla usługi JMS mogą ustawiać tylko te pola nagłówka JMS i właściwości, które można uzyskać na podstawie pól w deskryptorze komunikatu.

Produkt WebSphere MQ V7.0 udostępnia dodatkowe wsparcie dla aplikacji, które korzystają z interfejsu MQI w celu odbierania komunikatów z klas WebSphere MQ i wysyłania komunikatów do nich w przypadku aplikacji JMS.

Gdy aplikacja wywołuje wywołanie MQGET w celu odebrania komunikatu z klas produktu WebSphere MQ dla aplikacji JMS, aplikacja może wybrać opcję odebrania komunikatu w jeden z następujących sposobów:

1. Komunikat jest dostarczany z deskryptorem komunikatu, nagłówkiem MQRFH2 , który zawiera dane pochodzące z pól nagłówka i właściwości JMS oraz dane aplikacji.
2. Komunikat jest dostarczany z deskryptorem komunikatu, danymi aplikacji i zestawem właściwości komunikatu.

W opcji 2 każda właściwość komunikatu reprezentuje pole nagłówka JMS lub właściwość, która została pierwotnie odwzorowana przez klasy WebSphere MQ dla usługi JMS na pole w nagłówku MQRFH2 . Po wywołaniu wywołania MQGET aplikacja może użyć wywołania MQINQMP w celu pobrania wartości właściwości komunikatu. Użycie opcji 2 zamiast opcji 1 w celu odebrania komunikatu upraszcza logikę aplikacji w następujący sposób:

- Aplikacja nie musi analizować części zmiennej nagłówka MQRFH2 , która zawiera pole nagłówka JMS i dane właściwości zakodowane w formacie XML.
- Aplikacja nie musi konwertować danych znakowych w zmiennej części nagłówka MQRFH2 .

Odpowiednio, zanim aplikacja wywoła wywołanie MQPUT w celu wysłania komunikatu do klas WebSphere MQ dla aplikacji JMS, aplikacja może użyć wywołania MQSETMP w celu ustawienia wartości właściwości komunikatu zamiast konstruowania nagłówka MQRFH2 .

Łatwość serwisowania

Klasy WebSphere MQ classes for JMS zawierają szereg udoskonaleń związanych z serwisowalnością:

- Śledzenie.

  Klasy produktu WebSphere MQ classes for JMS zawierają klasę, której aplikacja może używać do sterowania śledzeniem. Aplikacja może uruchamiać i zatrzymują śledzenie, określać wymagany poziom szczegółowości w śledzeniu oraz dostosowywać dane wyjściowe śledzenia na różne sposoby.

- Rejestrowanie.

  Klasy WebSphere MQ classes for JMS przechowują plik dziennika, który zawiera komunikaty o błędach, które są potrzebne do poprawienia. Komunikaty są zapisywane w postaci jawnego tekstu. Klasy WebSphere MQ classes for JMS zawierają klasę, której aplikacja może używać do określania położenia pliku dziennika i jego maksymalnej wielkości.

- Technologia obsługi pierwszej awarii (FFST).

  Jeśli wystąpi poważna awaria, klasy WebSphere MQ classes for JMS generują raport FFST w pliku FDC. Raport FFST zawiera informacje, których usługa IBM służy do szybszego diagnozowania problemu.

- Informacje o wersji.

  Klasy WebSphere MQ classes for JMS zawierają klasę, której aplikacja może używać do wysyłania zapytań do wersji klas WebSphere MQ dla usługi JMS.

- Komunikaty o wyjątkach.

  Komunikaty o wyjątkach zostały rozszerzone, aby udostępnić więcej informacji na temat przyczyn błędów oraz działań wymaganych do poprawienia błędów.

- Serwery aplikacji.

  Poprawiono integrację funkcji serwisowych klas WebSphere MQ dla usługi JMS z elementami serwera WebSphere Application Server.

MQC jest zastępowane przez tabele MQConstants

Nowy pakiet, com.ibm.mq.constants, jest dostarczany z produktem IBM WebSphere MQ w wersji 7.0. Ten pakiet zawiera klasę MQConstants, która implementuje pewną liczbę interfejsów. Tabela MQConstants zawiera definicje wszystkich stałych, które znajdowały się w interfejsie MQC, oraz wiele nowych stałych. Interfejsy znajdujące się w tym pakiecie są ściśle zgodne z nazwami plików nagłówkowych stałych używanych w produkcie IBM WebSphere MQ.

Na przykład interfejs CMQC zawiera stałą MQOO_INPUT_SHARED; ten interfejs i stałą odpowiadają plikowej `cmqc.h` pliku nagłówkowego i stałej MQOO_INPUT_SHARED.

Klasa com.ibm.mq.constants może być używana z klasami produktu IBM WebSphere MQ dla klas Java i IBM WebSphere MQ dla usługi JMS.

Zmaterializowana tabela zapytania (MQC) jest nadal obecna i zawiera stałe, które wcześniej posiadała. Jednak w przypadku wszystkich nowych aplikacji należy użyć pakietu com.ibm.mq.constants .

Zapisywanie klas produktu WebSphere MQ dla aplikacji JMS

Po krótkim wprowadzeniu do modelu JMS ten temat zawiera szczegółowe informacje na temat pisania klas produktu WebSphere MQ dla aplikacji JMS.

Model JMS

Model JMS definiuje zestaw interfejsów, które mogą być używane przez aplikacje Java do wykonywania operacji przesyłania komunikatów. Klasy produktu WebSphere MQ dla usługi JMS, jako dostawcy JMS, definiują sposób, w jaki obiekty JMS są powiązane z pojęciami WebSphere MQ . Specyfikacja JMS oczekuje, że niektóre obiekty JMS będą administrowane obiektami.

Specyfikacja JMS oraz pakiet javax.jms definiują zestaw interfejsów, które mogą być używane przez aplikacje Java do wykonywania operacji przesyłania komunikatów. Poniższa lista zawiera podsumowanie głównych interfejsów JMS:

Miejsce docelowe
Miejsce docelowe to miejsce, w którym aplikacja wysyła komunikaty lub jest to źródło, z którego aplikacja odbiera komunikaty, lub oba te komunikaty.

ConnectionFactory
Obiekt ConnectionFactory hermetyzuje zestaw właściwości konfiguracyjnych dla połączenia. Aplikacja korzysta z fabryki połączeń w celu utworzenia połączenia.

Połączenie
Obiekt połączenia hermetyzuje aktywne połączenie aplikacji z serwerem przesyłania komunikatów. Aplikacja korzysta z połączenia w celu utworzenia sesji.

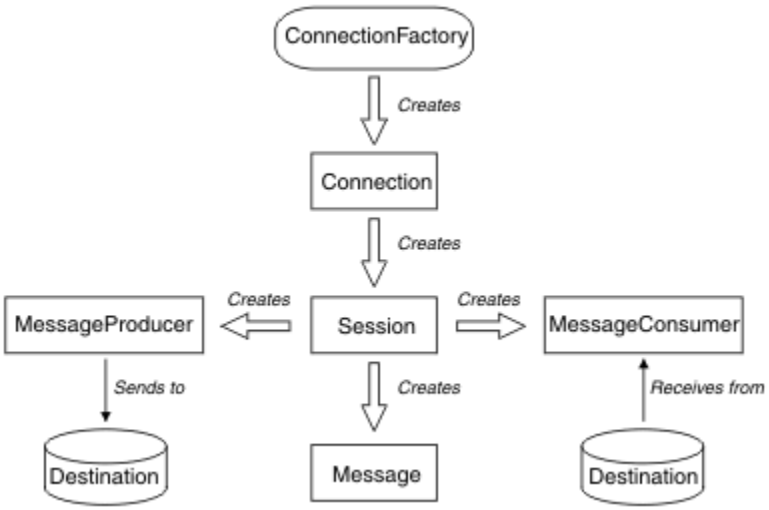
Sesja
Sesja jest jednowątkowym kontekstem do wysyłania i odbierania komunikatów. Aplikacja korzysta z sesji w celu utworzenia komunikatów, producentów komunikatów i konsumentów komunikatów. Sesja jest transakcowana lub nie jest transakowana.

Komunikat
Obiekt komunikatu hermetyzuje komunikat, który aplikacja wysyła lub odbiera.

MessageProducer
Aplikacja korzysta z producenta komunikatów w celu wysyłania komunikatów do miejsca docelowego.

MessageConsumer
Aplikacja używa konsumenta komunikatów do odbierania komunikatów wysyłanych do miejsca docelowego.

Rysunek 127 na stronie 828 przedstawia te obiekty i ich relacje.



Rysunek 127. Obiekty JMS i ich relacje

Obiekt docelowy, obiekt ConnectionFactorylub obiekt połączenia może być używany współbieżnie przez różne wątki aplikacji wielowątkowej, ale nie można jednocześnie używać obiektu Session, MessageProducerlub obiektu MessageConsumer przez różne wątki. Najprostszym sposobem zapewnienia, że obiekt Session, MessageProducerlub obiekt MessageConsumer nie jest używany współbieżnie, jest utworzenie osobnego obiektu sesji dla każdego wątku.

Usługa JMS obsługuje dwa style przesyłania komunikatów:

- Przesyłanie komunikatów w modelu punkt-punkt
- Przesyłanie komunikatów publikowania/subskrypcji

Te style przesyłania komunikatów są również nazywane *domenami przesyłania komunikatów*, a użytkownik może łączyć oba style przesyłania komunikatów w aplikacji. W domenie typu punkt z punktem miejsce docelowe jest kolejką, a w domenie publikowania/subskrybowania-miejscem docelowym jest temat.

W przypadku wersji JMS przed JMS 1.1programowanie w domenie typu punkt z punktem używa jednego zestawu interfejsów i metod, a programowanie dla domeny publikowania/subskrybowania używa innego zestawu. Oba zestawy są podobne, ale oddzielne. Za pomocą interfejsu JMS 1.1można użyć wspólnego zestawu interfejsów i metod obsługujących zarówno domeny przesyłania komunikatów. Wspólne interfejsy udostępniają niezależny widok domeny dla każdej domeny przesyłania komunikatów. Tabela 103 na stronie 828 zawiera listę niezależnych interfejsów domeny JMS i odpowiadających im interfejsów specyficznych dla domeny.

Tabela 103. Niezależne i specyficzne dla domeny interfejsy domeny JMS

Niezależne interfejsy domeny	**Interfejsy specyficzne dla domeny dla domeny punkt z punktem**	**Interfejsy specyficzne dla domeny dla domeny publikowania/subskrybowania**
ConnectionFactory	Fabryka QueueConnection	Fabryka TopicConnection
Połączenie	QueueConnection	TopicConnection
Miejsce docelowe	Kolejka	Temat
Sesja	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

Usługa JMS 1.1 zachowuje wszystkie interfejsy specyficzne dla domeny, a więc istniejące aplikacje nadal mogą używać tych interfejsów. Jednak w przypadku nowych aplikacji należy rozważyć użycie interfejsów niezależnych od domeny.

W klasach produktu WebSphere MQ dla usługi JMS obiekty JMS są powiązane z pojęciami WebSphere MQ w następujący sposób:

- Obiekt połączenia ma właściwości, które są pobierane z właściwości fabryki połączeń, która została użyta do utworzenia połączenia. Te właściwości sterują sposobem łączenia się aplikacji z menedżerem kolejek. Przykładami tych właściwości są nazwa menedżera kolejek oraz, dla aplikacji, która łączy się z menedżerem kolejek w trybie klienta, nazwa hosta lub adres IP systemu, na którym jest uruchomiony menedżer kolejek.
- Obiekt Session obudowuje uchwyt połączenia WebSphere MQ , który w związku z tym definiuje zasięg transakcyjny sesji.
- Obiekt MessageProducer i obiekt MessageConsumer każdego obudowuje uchwyt obiektu WebSphere MQ .

W przypadku używania klas WebSphere MQ dla usługi JMS stosowane są wszystkie normalne reguły produktu WebSphere MQ . Należy zwrócić uwagę, że aplikacja może wysłać komunikat do kolejki zdalnej, ale może odebrać komunikat tylko z kolejki, której właścicielem jest menedżer kolejek, z którym połączona jest aplikacja.

Specyfikacja JMS oczekuje, że obiekty ConnectionFactory i Destination mają być administrowane obiektami. Administrator tworzy i obsługuje administrowane obiekty w centralnym repozytorium, a aplikacja JMS pobiera te obiekty przy użyciu interfejsu JNDI (Java Naming and Directory Interface).

W klasach produktu WebSphere MQ dla usługi JMS implementacja interfejsu docelowego jest abstrakcyjną nadklasą kolejki i tematu, a więc instancją miejsca docelowego jest obiekt kolejki lub obiekt tematu. Niezależne interfejsy domeny traktują kolejkę lub temat jako miejsce docelowe. Domena przesyłania komunikatów dla obiektu MessageProducer lub MessageConsumer jest określana na podstawie tego, czy miejscem docelowym jest kolejka, czy temat.

W związku z tym w klasach produktu WebSphere MQ dla usługi JMS obiekty następujących typów mogą być administrowane obiektami:

- ConnectionFactory
- Fabryka QueueConnection
- Fabryka TopicConnection
- Kolejka
- Temat
- XAConnectionFactory
- Fabryka XAQueueConnection
- Fabryka XATopicConnection

Komunikaty JMS

Komunikaty JMS składają się z nagłówka, właściwości i treści. Usługa JMS definiuje pięć typów treści komunikatu.

Komunikaty JMS składają się z następujących części:

Nagłówek
Wszystkie komunikaty obsługują ten sam zestaw pól nagłówka. Pola nagłówka zawierają wartości, które są używane przez klientów i dostawców do identyfikowania i kierowania komunikatów.

Właściwości
Każdy komunikat zawiera wbudowane narzędzie do obsługi wartości właściwości zdefiniowanych przez aplikację. Właściwości udostępniają efektywny mechanizm filtrowania komunikatów zdefiniowanych przez aplikację.

Treść
Usługa JMS definiuje kilka typów treści komunikatu, które obejmują większość stylów przesyłania komunikatów, które są obecnie używane.

Usługa JMS definiuje pięć typów treści komunikatu:

Strumień
Strumień wartości podstawowych Java. Jest on wypełniany i odczytywaniany sekwencyjnie.

Odwzoruj
Zestaw par nazwa-wartość, gdzie nazwy są łańcuchami, a wartości są typami podstawowymi Java. Dostęp do pozycji można uzyskać w sposób sekwencyjny lub losowy według nazwy. Kolejność pozycji jest niezdefiniowana.

Tekstowy
Komunikat zawierający obiekt java.lang.String.

Obiekt
Komunikat, który zawiera obiekt Java z możliwością serializacji

Bajty
Strumień niewysłanych bajtów. Ten typ komunikatu dotyczy dosłownie kodowania treści w celu dopasowania do istniejącego formatu komunikatu.

Pole nagłówka JMSCorrelationID służy do połączenia jednego komunikatu z innym komunikatem. Zwykle łączy ona komunikat odpowiedzi z żądaniem komunikatu. Atrybut JMSCorrelationID może zawierać identyfikator komunikatu specyficznego dla dostawcy, łańcuch specyficzny dla aplikacji lub wartość bajtu dostawcy [] w formacie dostawcy.

Selektory komunikatów w usłudze JMS

Komunikaty mogą zawierać wartości właściwości zdefiniowane przez aplikację. Aplikacja może używać selektorów komunikatów do posiadania komunikatów filtru dostawcy JMS.

Komunikat zawiera wbudowane narzędzie do obsługi wartości właściwości zdefiniowanych przez aplikację. Ten mechanizm udostępnia mechanizm dodawania pól nagłówka specyficznego dla aplikacji do komunikatu. Właściwości umożliwiają aplikacji, przy użyciu selektorów komunikatów, możliwość wyboru lub filtrowania komunikatów dostawcy JMS w jego imieniu przy użyciu kryteriów specyficznych dla aplikacji. Właściwości zdefiniowane przez aplikację muszą być przestrzegane następujących reguł:

- Nazwy właściwości muszą być przestrzegane w regułach dla identyfikatora selektora komunikatów.
- Wartości właściwości mogą być typu Boolean, byte, short, int, long, float, double i String.
- Przedrostki JMSX i JMS_ name są zastrzeżone.

Wartości właściwości są ustawiane przed wysłaniem komunikatu. Gdy klient otrzymuje komunikat, właściwości komunikatu są tylko do odczytu. Jeśli klient podejmie próbę ustawienia właściwości w tym momencie, zgłaszany jest wyjątek MessageNotWriteableException . Jeśli wywoływane jest pole clearProperties , wówczas właściwości mogą być odczytywanych zarówno z, jak i zapisywane w.

Wartość właściwości może zduplikować wartość w treści komunikatu. Usługa JMS nie definiuje strategii dla elementów, które mogą zostać wprowadzone do właściwości. Programiści aplikacji muszą jednak mieć świadomość, że dostawcy JMS prawdopodobnie obsługują dane w treści komunikatu bardziej wydajnie niż dane we właściwościach komunikatu. Aby uzyskać najlepszą wydajność, aplikacje muszą używać właściwości komunikatu tylko wtedy, gdy muszą dostosować nagłówek komunikatu. Podstawową przyczyną tego działania jest obsługa dostosowanej selekcji komunikatów.

Selektor komunikatów JMS umożliwia klientowi określenie komunikatów, które są zainteresowane przy użyciu nagłówka komunikatu. Dostarczane są tylko komunikaty z nagłówkami, które są zgodne z selektorem.

Selektory komunikatów nie mogą odwoływać się do wartości treści komunikatu.

Selektor komunikatów jest zgodny z komunikatem, gdy selektor ma wartość true, gdy pole nagłówka komunikatu i wartości właściwości są podstawiane dla odpowiadających im identyfikatorów w selektorze.

Selektor komunikatów to łańcuch, którego składnia jest oparta na podzbiorze składni wyrażeń warunkowych SQL92 . Kolejność, w jakiej selektor komunikatów jest wartościowany, jest od lewej do prawej w obrębie poziomu priorytetu. Aby zmienić to zamówienie, można użyć nawiasów. Predefiniowane literały selektora i nazwy operatorów są zapisywane w tym miejscu wielkimi literami, jednak nie są one w nich rozróżniane.

Selektor może zawierać:

- Literały
  - Literał łańcuchowy jest ujęty w znaki cudzysłowu. Podwojony znak cudzysłowu reprezentuje cudzysłów. Przykładami są literały i literał. Podobnie jak literały łańcuchowe języka Java, używają one kodowania znaków Unicode.
  - Dokładny literał liczbowy jest wartością liczbową bez przecinka dziesiętnego, np. 57, -957 i +62. Obsługiwane są liczby z zakresu długości języka Java.
  - Przybliżony literał liczbowy to wartość liczbowa w notacji naukowej, taka jak 7E3 lub -57.9E2, lub wartość liczbowa z dziesiętnym, takim jak 7., -95.7lub +6.2. Obsługiwane są liczby z zakresu dwukrotnego języka Java.
  - Literały boolowskie TRUE i FALSE.
- Identyfikatory:
  - Identyfikator jest nieograniczoną sekwencją znaków języka Java i cyfr Java, z których pierwsza musi być literą języka Java. Litera to dowolny znak, dla którego metoda Character.isJavaLetter zwraca wartość true. Obejmuje to _ oraz $. Litera lub cyfra to dowolny znak, dla którego metoda Character.isJavaLetterOrDigit zwraca wartość true.
  - Identyfikatory nie mogą być nazwami NULL, TRUE ani FALSE.
  - Identyfikatory nie mogą być identyfikatorami NOT, AND, OR, BETWEEN, LIKE, IN lub IS.
  - Identyfikatory są odwołaniami do pól nagłówka lub odwołaniami do właściwości.
  - W identyfikatorach rozróżniana jest wielkość liter.
  - Odwołania do pól nagłówka komunikatu są ograniczone do:
    - JMSDeliveryMode
    - JMSPriority
    - JMSMessageID
    - JMSTimestamp
    - JMSCorrelationID
    - JMSType

    Wartości JMSMessageID, JMSTimestamp, JMSCorrelationIDi JMSType mogą mieć wartość NULL, a jeśli tak, są traktowane jako wartość NULL.
  - Każda nazwa rozpoczynający się od JMSX jest nazwą właściwości zdefiniowaną przez JMS.
  - Każda nazwa rozpoczynający się od JMS_ jest nazwą właściwości specyficzną dla dostawcy.
  - Każda nazwa, która nie zaczyna się od usługi JMS, jest nazwą właściwości specyficznej dla aplikacji. Jeśli istnieje odwołanie do właściwości, która nie istnieje w komunikacie, jej wartość jest równa NULL. Jeśli ta wartość istnieje, jej wartością jest odpowiadająca jej wartość właściwości.
- Biały znak jest taki sam, jak w przypadku języka Java: obszar, karta pozioma, kanał informacyjny formularza i terminator linii.
- Wyrażenia:
  - Selektor jest wyrażeniem warunkowym. Selektor wartościowany do prawdziwych dopasowań; selektor, którego wartościowanie ma wartość false lub nieznany, nie jest zgodny.
  - Wyrażenia arytmetyczne składają się z samych siebie, operacji arytmetycznych, identyfikatorów (z wartością, która jest traktowana jako literał liczbowy), oraz literałów liczbowych.
  - Wyrażenia warunkowe składają się z samych siebie, operacji porównania i operacji logicznych.

- Standard bracketing (), aby ustawić kolejność, w jakiej wyrażenia są wartościowane, jest obsługiwane.
- Operatory logiczne w kolejności kolejności wykonywania: NOT, AND, OR.
- Operatory porównania: =, >, > =, <, < =, < > (nie równe).
  - Porównywane są tylko wartości tego samego typu. Jedynym wyjątkiem jest to, że poprawne jest porównanie dokładnych wartości liczbowych i przybliżonych wartości liczbowych. (Wymagana konwersja typów jest zdefiniowana przez reguły promocji numerycznej języka Java). Jeśli istnieje próba porównania różnych typów, selektor zawsze ma wartość false.
  - Porównywanie łańcuchowe i boolowskie jest ograniczone do = i < >. Dwa łańcuchy są równe tylko wtedy, gdy zawierają taką samą sekwencję znaków.
- Operatory arytmetyczne w kolejności kolejności wykonywania zadań:
  - +,-unary.
  - *,/, mnożenie i dzielenie.
  - +,-,-, dodawanie i odejmowanie.
  - Operacje arytmetyczne na wartości NULL nie są obsługiwane. Jeśli są one uruchomione, pełny selektor zawsze ma wartość false.
  - Operacje arytmetyczne muszą używać promocji numerycznej języka Java.
- Operator porównania arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 i arithmetic-expr3 :
  - Wiek od 15 do 19 lat jest odpowiednikiem wieku > = 15 AND wiek < = 19.
  - Wiek NIE MIĘDZY 15 a 19 jest odpowiednikiem wieku < 15 LUB wieku > 19.
  - Jeśli którekolwiek z wyrażeń operacji BETWEEN ma wartość NULL, wartością tej operacji jest false (fałsz). Jeśli którekolwiek z wyrażeń operacji NOT BETWEEN ma wartość NULL, wartość tej operacji jest równa true.
- identyfikator [NOT] IN (string-literal1, string-literal2, ...) operator porównania, gdzie identyfikator ma wartość String lub NULL.
  - Kraj IN ("Wielka Brytania", "USA", "Francja") ma wartość "true" dla "UK" i "false" dla "Peru". Jest on równoważny z wyrażeniem (Country = 'UK ') OR (Country = 'US') OR (Country = 'Francja ').
  - Kraj NOT IN ("UK", "US", "Francja") ma wartość "false" dla "UK" i "true" w odniesieniu do "Peru". Jest ono równoznaczne z wyrażeniem NOT ((Country = 'UK ') OR (Country = 'US') OR (Country = 'France ')).
  - Jeśli identyfikator operacji IN lub NOT IN ma wartość NULL, wartość tej operacji jest nieznana.
- identifier [NOT] LIKE wzorzec-wartość [ESCAPE escape-character] operator porównania, gdzie identyfikator ma wartość łańcuchową. wzorzec-wartość jest literałem łańcuchowym, gdzie _ oznacza dowolny pojedynczy znak, a% oznacza dowolną sekwencję znaków (włącznie z pustą sekwencją). Wszystkie inne postacie stoją dla siebie. Opcjonalny znak zmiany znaczenia jest jednoznakowym literałem łańcuchowym o znaku, który jest używany do zmiany znaczenia specjalnego znaczenia _ i% w wartości wzorca.
  - Telefon LIKE '12%3' ma wartość true dla 123 i 12993, a wartość false dla 1234.
  - Słowo LIKE 'l_se' ma wartość true dla "lose" i false dla "loose".
  - podkreślono wartość LIKE '_ %' ESCAPE' \ ' jest wartością true dla opcji "_foo" i false dla "bar".
  - Telefon NOT LIKE '12%3' ma wartość false dla 123 i 12993, a wartość true dla 1234.
  - Jeśli identyfikator operacji LIKE lub NOT LIKE ma wartość NULL, wartość tej operacji jest nieznana.
- identyfikator IS NULL operator porównania testów dla wartości pola nagłówka o wartości NULL lub brakującej wartości właściwości.
  - prop_name ma wartość NULL.
- Identyfikator IS NOT NULL operator porównania testów dla istnienia wartości pola nagłówka innej niż NULL lub wartości właściwości.
  - prop_name IS NOT NULL.

Poniższy selektor komunikatów wybiera komunikaty z typem komunikatu samochodu, koloru niebieskiego i wagą większą niż 2500 funtów:

```
"JMSType = 'car' AND color = 'blue' AND weight > 2500"
```

Jak zauważono na poprzedniej liście, wartości właściwości mogą mieć wartość NULL. Wartościowanie wyrażeń selektora, które zawierają wartości NULL, jest definiowane przez semantykę NULL języka SQL 92. Poniższa lista zawiera krótki opis tych semantyki:

- SQL traktuje wartość NULL jako nieznaną.
- Porównanie lub arytmetyka z nieznaną wartością zawsze daje nieznaną wartość.
- Operator IS NULL przekształca nieznaną wartość w wartość PRAWDA.
- Operator IS NOT NULL przekształca nieznaną wartość w wartość FALSE.

Chociaż SQL obsługuje stałe porównania dziesiętne i arytmetyczne, selektory komunikatów JMS nie są dostępne. Dlatego dokładne literały liczbowe są ograniczone do tych bez liczby dziesiętnej. Dlatego też istnieją cyfry z separatą dziesiętną jako alternatywna reprezentacja dla przybliżonej wartości liczbowej.

Komentarze SQL nie są obsługiwane.

Odwzorowywanie komunikatów JMS na komunikaty produktu WebSphere MQ

Komunikaty produktu WebSphere MQ składają się z deskryptora komunikatu, opcjonalnego nagłówka MQRFH2 i treści. Treść komunikatu JMS jest częściowo odwzorowywana i częściowo kopiowana do komunikatu produktu WebSphere MQ .

W tym temacie opisano, w jaki sposób struktura komunikatu JMS opisana w pierwszej części tej sekcji jest odwzorowywana na komunikat WebSphere MQ . Interesują go programiści, którzy chcą przesyłać komunikaty między usługą JMS i tradycyjnymi aplikacjami produktu WebSphere MQ . Interesują go także osoby, które chcą manipulować komunikatami przesyłanych między dwiema aplikacjami JMS, na przykład w implementacji produktu WebSphere Message Broker.

Ta sekcja nie ma zastosowania, jeśli aplikacja korzysta z połączenia w czasie rzeczywistym z brokerem. Jeśli aplikacja korzysta z połączenia w czasie rzeczywistym, cała komunikacja jest wykonywana bezpośrednio za pośrednictwem protokołu TCP/IP; nie są w nim związane żadne kolejki ani komunikaty produktu WebSphere MQ .

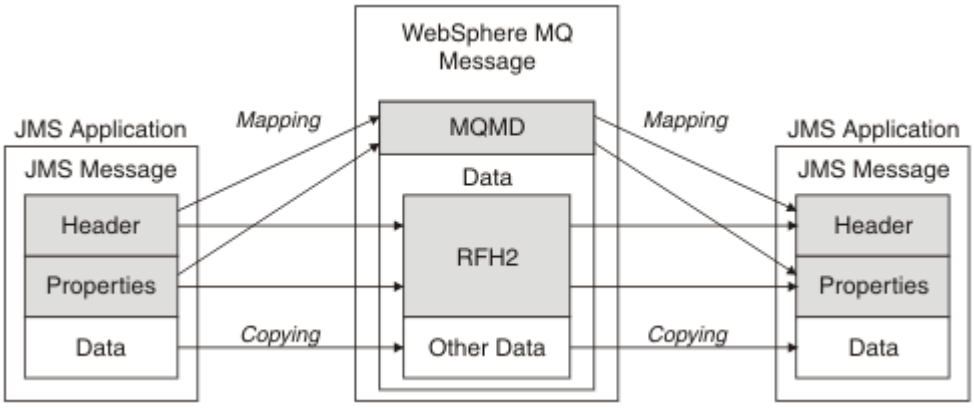
Komunikaty produktu WebSphere MQ składają się z trzech komponentów:

- Deskryptor komunikatu produktu WebSphere MQ (MQMD)
- A WebSphere MQ MQRFH2 header
- Treść komunikatu.

Obiekt MQRFH2 jest opcjonalny, a jego uwzględnienie w komunikacie wychodzącym jest zarządzane przez flagę w klasie miejsca docelowego JMS. Opcję tę można ustawić przy użyciu narzędzia administracyjnego WebSphere MQ JMS. Ponieważ MQRFH2 zawiera informacje specyficzne dla usługi JMS, zawsze należy je uwzględnić w komunikacie, gdy nadawca wie, że odbierającym miejscem docelowym jest aplikacja JMS. W normalnych warunkach pomiń MQRFH2 podczas wysyłania komunikatu bezpośrednio do aplikacji innej niż JMS. Jest to spowodowane tym, że taka aplikacja nie oczekuje MQRFH2 w swoim komunikacie WebSphere MQ .

Jeśli komunikat przychodzący nie ma nagłówka MQRFH2 , to obiekt kolejki lub tematu uzyskany z pola nagłówka JMSReplyTo komunikatu domyślnie ma ten zestaw flag, dzięki czemu komunikat odpowiedzi wysłany do kolejki lub tematu również nie ma nagłówka MQRFH2 . Można wyłączyć to zachowanie, w tym nagłówek MQRFH2 w komunikacie odpowiedzi tylko wtedy, gdy oryginalny komunikat ma nagłówek MQRFH2 , ustawiając właściwość TARGCLIENTMATCHING dla fabryki połączeń na wartość NO.

Rysunek 128 na stronie 834 pokazuje, w jaki sposób struktura komunikatu JMS jest transformowana do komunikatu produktu WebSphere MQ i ponownie:



Rysunek 128. Sposób transformowania komunikatów między JMS i WebSphere MQ przy użyciu nagłówka MQRFH2

Struktury są transformowane na dwa sposoby:

Odwzorowanie
W przypadku, gdy deskryptor MQMD zawiera pole równoważne z polem JMS, pole JMS jest odwzorowywane na pole MQMD. Dodatkowe pola MQMD są prezentowane jako właściwości JMS, ponieważ może być konieczne pobranie lub ustawienie tych pól przez aplikację JMS podczas komunikowania się z aplikacją inną niż JMS.

Kopiowanie
Jeśli nie ma odpowiednika deskryptora MQMD, to pole lub właściwość nagłówka JMS są przekazywane, prawdopodobnie transformowane, jako pole w MQRFH2.

Nagłówek MQRFH2 i JMS
W tej kolekcji tematów opisano nagłówek MQRFH, wersja 2, który zawiera dane specyficzne dla usługi JMS, które są powiązane z treścią komunikatu. MQRFH2 , wersja 2, jest rozszerzalnym nagłówkiem i może również zawierać dodatkowe informacje, które nie są bezpośrednio powiązane z usługą JMS. Ta sekcja obejmuje jednak tylko jej użycie przez usługę JMS.Pełny opis znajduje się w sekcji MQRFH2 -Reguły i formatowanie nagłówka 2.

Znajdują się tam dwie części nagłówka, część stała i część zmiennej.

Stała część
The fixed portion is modeled on the *standardowe* WebSphere MQ header pattern and consists of the following fields:

StrucId (MQCHAR4)
Identyfikator struktury.

Musi to być wartość MQRFH_STRUC_ID (wartość: "RFH ") (wartość początkowa).

Zdefiniowano także wartość MQRFH_STRUC_ID_ARRAY (wartość: "R","F","H"," ").

Wersja (MQLONG)
Numer wersji struktury.

Musi to być wartość MQRFH_VERSION_2 (wartość: 2) (wartość początkowa).

StrucLength (MQLONG)
Łączna długość MQRFH2, w tym pola danych NameValue.

Wartość ustawiona w polu StrucLength musi być wielokrotnością 4 (dane w polach danych NameValuemogą być dopełniane znakami spacji, aby to osiągnąć).

Kodowanie (MQLONG)
Kodowanie danych.

Kodowanie dowolnych danych liczbowych w części komunikatu po wykonaniu komendy MQRFH2 (następny nagłówek lub dane komunikatu po tym nagłówku).

CodedCharSetId (MQLONG)
Identyfikator kodowanego zestawu znaków.

Reprezentacja dowolnych danych znakowych w części komunikatu po MQRFH2 (następny nagłówek lub dane komunikatu po tym nagłówku).

Format (MQCHAR8)
Nazwa formatu.

Nazwa formatu dla części komunikatu, która jest następująca po MQRFH2.

Flagi (MQLONG)
Flagi.

MQRFH_NO_FLAGS = 0. Nie ustawiono flag.

NameValueCCSID (MQLONG)
Identyfikator kodowanego zestawu znaków (CCSID) dla łańcuchów znaków danych NameValuezawartych w tym nagłówku. Dane NameValuemogą być kodowane w zestawie znaków, który różni się od innych łańcuchów znaków, które są zawarte w nagłówku (StrucID i Format).

Jeśli identyfikator CCSID NameValueto 2-bajtowy identyfikator CCSID Unicode (1200, 13488 lub 17584), kolejność bajtów w kodzie Unicode jest taka sama, jak kolejność bajtów w polach liczbowych w tabeli MQRFH2. (Na przykład sam identyfikator CCSID wersji, StrucLengthi NameValue).

Tabela 104. Możliwe wartości dla pola identyfikatora CCSID NameValue

Wartość	Znaczenie
1200	Otwarto UCS2
1208	UTF8
13488	Podzbiór UCS2 2.0
17584	Podzbiór UCS2 2.1 (zawiera symbol Euro)

Część zmiennej
Część zmiennej jest zgodna ze stałą porcją. Część zmiennej zawiera zmienną liczbę folderów MQRFH2 . Każdy folder zawiera zmienną liczbę elementów lub właściwości. Właściwości powiązane z grupą folderów. Nagłówki MQRFH2 utworzone przez usługę JMS mogą zawierać dowolne z następujących folderów:

Folder < mcd>

`mcd` zawiera właściwości opisujące format komunikatu. Na przykład właściwość domeny usługi komunikatu `Msd` identyfikuje komunikat JMS jako `JMSTextMessage`, `JMSBytesMessage`, `JMSStreamMessage`, `JMSMapMessage`, `JMSObjectMessage` lub wartość NULL.

Folder `mcd` jest zawsze obecny w komunikacie usługi Java Message Service zawierającym `MQRFH2`.

Jest ona zawsze obecna w komunikacie zawierającym `MQRFH2` wysłanym z produktu WebSphere Message Broker. Opisuje on domenę, format, typ i zestaw komunikatu.

Tabela 105. mcd - nazwa właściwości, synonim, typ danych i folder

Synonim właściwości	Nazwa właściwości	Typ danych	Folder
	`mcd.Msd`	`string`	`<mcd><Msd>`*`messageDomain`*`</Msd></mcd>`
	`mcd.Set`	`string`	`<mcd><Set>`*`messageDomain`*`</Set></mcd>`

Tabela 105. mcd - nazwa właściwości, synonim, typ danych i folder (kontynuacja)			
Synonim właściwości	**Nazwa właściwości**	**Typ danych**	**Folder**
	mcd.Type	string	<mcd><Type>*messageDomain*</Type></mcd>
	mcd.Fmt	string	<mcd><Fmt>*messageDomain*</Fmt></mcd>

Nie należy dodawać własnych właściwości w folderze mcd.

Folder < jms>

jms zawiera pola nagłówka JMS oraz właściwości JMSX, które nie mogą być w pełni wyrażone w produkcie MQMD. Folder jms zawsze znajduje się w nagłówku MQRFH2 JMS.

Folder < usr>

usr zawiera zdefiniowane przez aplikację właściwości JMS powiązane z komunikatem. Folder usr jest obecny tylko wtedy, gdy w aplikacji ustawiono właściwość definiowaną przez aplikację.

Folder < mqext>

Produkt mqext zawiera właściwości używane tylko przez serwer WebSphere Application Server. Folder jest obecny tylko wtedy, gdy aplikacja ustawiła co najmniej jedną z zdefiniowanych właściwości IBM .

Tabela 106. mqext - nazwa właściwości, synonim, typ danych i folder			
Synonim właściwości	**Nazwa właściwości**	**Typ danych**	**Folder**
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>*armCorrelator*</Arm></mqext>
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>*wrmCorrelator*</Wrm></mqext>

Nie należy dodawać własnych właściwości w folderze mqext.

Folder < mqps>

mqps zawiera właściwości, które są używane tylko przez proces publikowania/subskrybowania produktu IBM WebSphere MQ. Folder jest obecny tylko wtedy, gdy w przypadku aplikacji ustawiono co najmniej jedną zintegrowaną właściwość publikowania/subskrybowania.

Tabela 107. mqps - nazwa właściwości, synonim, typ danych i folder			
Synonim właściwości	**Nazwa właściwości**	**Typ danych**	**Folder**
MQTopicString	mqps.Top	string	<mqps><Top>*topicString*</Top></mqps>
MQSubUserData	mqps.Sud	string	<mqps><Sud>*subscriberUserData*</Sud></mqps>
MQIsRetained	mqps.Ret	boolean	<mqps><Ret>*isRetained*</Ret></mqps>
MQPubOptions	mqps.Pub	i8	<mqps><Pub>*publicationOptions*</Pub></mqps>

Tabela 107. mqps - nazwa właściwości, synonim, typ danych i folder (kontynuacja)			
Synonim właściwości	**Nazwa właściwości**	**Typ danych**	**Folder**
MQPubLevel	mqps.Pbl	i8	<mqps><Pbl>*publicationLevel*</Pbl></mqps>
MQPubTime	mqpse.Pts	string	<mqps><Pts>*publicationTime*</Pts></mqps>
MQPubSeqNum	mqpse.Seq	i8	<mqps><Seq>*publicationSequenceNumber*</Seq></mqps>
MQPubStrIntData	mqpse.Sid	string	<mqps><Sid>*publicationData*</Sid></mqps>
MQPubFormat	mqpse.Pfmt	i8	<mqps><Pfmt>*messageFormat*</Pfmt></mqps>

Nie należy dodawać własnych właściwości w folderze mqps.

W programie Tabela 108 na stronie 837 jest wyświetlana pełna lista nazw właściwości.

Tabela 108. Foldery MQRFH2 i właściwości używane przez JMS				
Nazwa pola JMS	**Typ Java**	**Nazwa folderu MQRFH2**	**Nazwa właściwości**	**Typ/wartości**
JMSDestination	Miejsce docelowe	jms	Dst	string (łańcuch)
JMSExpiration	long	jms	Wyg.	i8
JMSPriority	int	jms	PRI	i4
JMSDeliveryMode	int	jms	Dlv	i4
JMSCorrelationID	Łańcuch	jms	CID	string (łańcuch)
JMSReplyTo	Miejsce docelowe	jms	Rto	string (łańcuch)
JMSTimestamp	long	jms	tms	i8
JMSType	Łańcuch	MCD	Typ, Ustaw, Fmt	string (łańcuch)
JMSXGroupID	Łańcuch	jms	Identyfikator grupy	string (łańcuch)
JMSXGroupSeq	int	jms	Sekw	i4
xxx (zdefiniowana przez użytkownika)	Dowolna	USR	xxx	dowolne
		MCD	MSD	jms_none tekst_jms jms_bytes jms_map Strumień jms_ obiekt jms_object

NameValue(długość nazwy) (MQLONG)
Długość (w bajtach) łańcucha danych NameValuewystępujący bezpośrednio po tej zmiennej długości (nie obejmuje własnej długości).

Dane NameValue(MQCHARn)
Pojedynczy łańcuch znaków, którego długość w bajtach jest podana w poprzedzającym polu NameValueLength. Zawiera on folder zawierający sekwencję właściwości. Każda właściwość jest tercetu typu nazwa/typ/wartość, zawartego w elemencie XML, którego nazwa jest nazwą folderu, w następujący sposób:

```
 <foldername>
triplet1 triplet2 .....   tripletn </foldername>
```

Po zamykaniu znacznika `</foldername>` mogą występować spacje jako znaki dopełniania. Każdy triplet jest zakodowany przy użyciu składni typu XML:

```
   <name dt='datatype'>value</name>
```

Element `dt='datatype'` jest opcjonalny i jest pomijany dla wielu właściwości, ponieważ typ danych jest predefiniowany. Jeśli ta opcja jest włączona, przed znacznikiem `dt=` musi zostać umieszczony jeden lub więcej znaków spacji.

name
jest nazwą właściwości; patrz Tabela 108 na stronie 837.

datatype
musi być zgodne, po złożeniu, jeden z typów danych wymienionych w sekcji Tabela 109 na stronie 838.

value
jest łańcuchową reprezentacją wartości, która ma być transportowana, przy użyciu definicji w produkcie Tabela 109 na stronie 838.

Wartość NULL jest zakodowana przy użyciu następującej składni:

```
<name dt='datatype' xsi:nil='true'></name>
```

Nie należy używać produktu `xsi:nil='false'`.

Tabela 109. Typy danych właściwości

Typ danych	Definicja
string (łańcuch)	Dowolna sekwencja znaków z wyjątkiem < i &
boolean (boolowskie)	Znak `0` lub `1` (`0` = false, `1` = true)
bin.hex	Cyfry szesnastkowe reprezentujące oktety
i1	Liczba, wyrażona za pomocą cyfr `0..9`, z opcjonalnym znakiem (bez ułamków lub wykładników). Musi leżeć w zakresie od -128 do 127 włącznie
i2	Liczba, wyrażona za pomocą cyfr `0..9`, z opcjonalnym znakiem (bez ułamków lub wykładników). Musi leżeć w zakresie od -32768 do 32767 włącznie
i4	Liczba, wyrażona za pomocą cyfr `0..9`, z opcjonalnym znakiem (bez ułamków lub wykładników). Musi leżeć w zakresie od -2147483648 do 2147483647 włącznie
i8	Liczba, wyrażona za pomocą cyfr `0..9`, z opcjonalnym znakiem (bez ułamków lub wykładników). Musi leżeć w zakresie od -9223372036854775808 do 9223372036854775807 włącznie

Tabela 109. Typy danych właściwości (kontynuacja)	
Typ danych	**Definicja**
int	Liczba, wyrażona za pomocą cyfr `0..9`, z opcjonalnym znakiem (bez ułamków lub wykładników). Musi znajdować się w tym samym zakresie co i8. Może być używany w miejsce jednego z typów i *, jeśli nadawca nie chce skojarzyć konkretnej precyzji z właściwością
r4	Liczba zmiennopozycyjna, wielkość < = 3.40282347E+38, > = 1.175E-37 wyrażona przy użyciu cyfr `0..9`, opcjonalny znak, opcjonalne cyfry ułamkowe, opcjonalny wykładnik
r8	Liczba zmiennopozycyjna, wielkość < = 1.7976931348623E+308, > = 2.225E-307 wyrażona przy użyciu cyfr `0..9`, opcjonalny znak, opcjonalne cyfry ułamkowe, opcjonalny wykładnik

Wartość łańcuchowa może zawierać spacje. W wartości łańcuchowej należy użyć następujących sekwencji o zmienionym znaczeniu:

- `&` dla znaku &
- < dla znaku <

Można użyć następujących sekwencji o zmienionym znaczeniu, ale nie są one wymagane:

- `>` dla znaku >
- `&apos;` dla znaku '
- `&quot;` dla znaku "

Pola i właściwości JMS z odpowiednimi polami MQMD
W tych tabelach są wyświetlane pola MQMD równoważne z polami nagłówka JMS, właściwościami JMS i właściwościami specyficznymi dla dostawcy JMS.

Tabela 110 na stronie 839 zawiera listę pól nagłówka JMS, a Tabela 111 na stronie 840 zawiera listę właściwości JMS, które są odwzorowywane bezpośrednio na pola MQMD. Tabela 112 na stronie 840 Zawiera listę właściwości specyficznych dla dostawcy oraz pól MQMD, do których są one odwzorowane.

Tabela 110. Odwzorowywanie pól nagłówka JMS na pola MQMD			
Pole nagłówka JMS	**Typ Java**	**Pole MQMD**	**Typ C**
JMSDeliveryMode	int	Trwałość	MQLONG
JMSExpiration	long	Utrata ważności	MQLONG
JMSPriority	int	Priorytet	MQLONG
JMSMessageID	Łańcuch	MsgID	MQBYTE24
JMSTimestamp	long	PutDate PutTime	MQCHAR8 MQCHAR8
JMSCorrelationID	Łańcuch	CorrelId	MQBYTE24

Tabela 111. Odwzorowanie właściwości JMS na pola MQMD			
właściwość JMS	**Typ Java**	**Pole MQMD**	**Typ C**
JMSXUserID	Łańcuch	UserIdentifier	MQCHAR12
JMSXAppID	Łańcuch	Nazwa_aplikacji_wstawiającej	MQCHAR28
JMSXDeliveryCount	int	BackoutCount	MQLONG
JMSXGroupID	Łańcuch	GroupId	MQBYTE24
JMSXGroupSeq	int	Numer_kolejny_komunikatu	MQLONG

Tabela 112. Odwzorowywanie właściwości specyficznych dla dostawcy JMS na pola MQMD			
Właściwość specyficzna dla dostawcy JMS	**Typ Java**	**Pole MQMD**	**Typ C**
Wyjątek JMS_IBM_Report_Exception	int	Raport	MQLONG
Wygaśnięcie JMS_IBM_Report_Expiration	int	Raport	MQLONG
Plik JMS_IBM_Report_COA	int	Raport	MQLONG
JMS_IBM_Report_COD	int	Raport	MQLONG
Plik JMS_IBM_Report_PAN	int	Raport	MQLONG
Plik JMS_IBM_Report_NAN	int	Raport	MQLONG
JMS_IBM_Report_Pass_Msg_ID	int	Raport	MQLONG
JMS_IBM_Report_Pass_Correl_ID	int	Raport	MQLONG
JMS_IBM_Report_Discard_Msg	int	Raport	MQLONG
JMS_IBM_MsgType	int	MsgType	MQLONG
Opinie JMS_IBM_Feedback	int	Opinie	MQLONG
Format JMS_IBM_Format	Łańcuch	Format"1" na stronie 841	MQCHAR8
Typ JMS_IBM_PutAppl	int	Typ_aplikacji_wstawiającej	MQLONG
JMS_IBM_Encoding	int	Kodowanie	MQLONG
Zestaw znaków JMS_IBM_Character_Set	Łańcuch	CodedCharacterSetId"2" na stronie 841	MQLONG
JMS_IBM_PutDate	Łańcuch	PutDate	MQCHAR8
JMS_IBM_PutTime	Łańcuch	PutTime	MQCHAR8
Grupa JMS_IBM_Last_Msg_In_Group	boolean (boolowskie)	MsgFlags	MQLONG

Tabela 112. Odwzorowywanie właściwości specyficznych dla dostawcy JMS na pola MQMD (kontynuacja)			
Właściwość specyficzna dla dostawcy JMS	**Typ Java**	**Pole MQMD**	**Typ C**
Uwaga: 1. Format JMS_IBM_Format reprezentuje format treści komunikatu. Może to być zdefiniowane przez aplikację ustawiające właściwość JMS_IBM_Format komunikatu (należy pamiętać o tym, że istnieje limit 8 znaków) lub może być domyślnie ustawiona na format WebSphere MQ treści komunikatu odpowiedniej dla typu komunikatu JMS. Pliki JMS_IBM_Format są odwzorowywać na pole Format MQMD tylko wtedy, gdy komunikat nie zawiera sekcji RFH lub RFH2 . W typowym komunikacie jest on odwzorowyany na pole Format w RFH2 bezpośrednio poprzedzające treść komunikatu. 2. Wartość właściwości JMS_IBM_Character_Set jest wartością typu String, która zawiera zestaw znaków Java odpowiadający wartości liczbowej CodedCharacterSetId . Pole MQMD CodedCharacterSetId jest wartością liczbową, która zawiera odpowiednik łańcucha zestawu znaków Java określonego we właściwości JMS_IBM_Character_Set.			

Odwzorowywanie pól JMS na pola produktu WebSphere MQ (komunikaty wychodzące)
W tych tabelach przedstawiono sposób, w jaki pola nagłówka i właściwości JMS są odwzorowywane na pola MQMD i MQRFH2 w czasie wysyłania () lub publikowania ().

Tabela 113 na stronie 841 pokazuje, w jaki sposób pola nagłówka JMS są odwzorowywane na pola MQMD/RFH2 w czasie wysyłania () lub publikowania (). Tabela 114 na stronie 842 Pokazuje, w jaki sposób właściwości JMS są odwzorowywane na pola MQMD/RFH2 w czasie wysyłania () lub publikowania (). Tabela 115 na stronie 842 pokazuje, w jaki sposób właściwości specyficzne dla dostawcy JMS są odwzorowywane na pola MQMD w czasie wysyłania () lub publikowania (),

W przypadku pól oznaczonych przez obiekt Message Object nadawana wartość jest wartością przechowaną w komunikacie JMS bezpośrednio przed operacją send () lub publish (). Wartość w komunikacie JMS pozostaje niezmieniona przez operację.

W przypadku pól oznaczonych metodą Send Method (Set by Send Method) wartość jest przypisywany po wykonaniu funkcji send () lub publikowania (dowolna wartość przechowana w komunikacie JMS jest ignorowana). Wartość w komunikacie JMS zostanie zaktualizowana w celu wyświetlenia użytej wartości.

Pola oznaczone jako Odbierz-tylko nie są przesyłane i pozostają niezmienione w wiadomości przez funkcję send () lub publish ().

Tabela 113. Odwzorowanie pola komunikatu wychodzącego

Nazwa pola nagłówka JMS	**Pole MQMD używane do transmisji**	**Nagłówek**	**Ustawione przez**
JMSDestination		MQRFH2	Metoda wysyłania
JMSDeliveryMode	Trwałość	MQRFH2	Metoda wysyłania
JMSExpiration	Utrata ważności	MQRFH2	Metoda wysyłania
JMSPriority	Priorytet	MQRFH2	Metoda wysyłania
JMSMessageID	MsgID		Metoda wysyłania
JMSTimestamp	PutDate/PutTime		Metoda wysyłania

Tabela 113. Odwzorowanie pola komunikatu wychodzącego (kontynuacja)

Nazwa pola nagłówka JMS	Pole MQMD używane do transmisji	Nagłówek	Ustawione przez
JMSCorrelationID	CorrelId	MQRFH2	Obiekt komunikatu
JMSReplyTo	ReplyToQ/ReplyToQMgr	MQRFH2	Obiekt komunikatu
JMSType		MQRFH2	Obiekt komunikatu
JMSRedelivered			Tylko odbiór

Uwaga:

1. Pole MQMD CodedCharacterSetId jest wartością liczbową, która zawiera odpowiednik łańcucha zestawu znaków Java określonego we właściwości JMS_IBM_Character_Set.

Tabela 114. Odwzorowanie właściwości JMS komunikatu wychodzącego

Nazwa właściwości JMS	Pole MQMD używane do transmisji	Nagłówek	Ustawione przez
JMSXUserID	UserIdentifier		Metoda wysyłania
JMSXAppID	Nazwa_aplikacji_wstawiająca j		Metoda wysyłania
JMSXDeliveryCount			Tylko odbiór
JMSXGroupID	GroupId	MQRFH2	Obiekt komunikatu
JMSXGroupSeq	Numer_kolejny_komunikatu	MQRFH2	Obiekt komunikatu

Tabela 115. Odwzorowanie właściwości specyficznej dla dostawcy JMS komunikatów wychodzących

Nazwa właściwości specyficznej dla dostawcy JMS	Pole MQMD używane do transmisji	Nagłówek	Ustawione przez
Wyjątek JMS_IBM_Report_Exception	Raport		Obiekt komunikatu
Wygaśnięcie JMS_IBM_Report_Expiration	Raport		Obiekt komunikatu
JMS_IBM_Report_COA/COD	Raport		Obiekt komunikatu
Plik JMS_IBM_Report_NAN/PAN	Raport		Obiekt komunikatu
JMS_IBM_Report_Pass_Msg_ID	Raport		Obiekt komunikatu
JMS_IBM_Report_Pass_Correl_ID	Raport		Obiekt komunikatu

Tabela 115. Odwzorowanie właściwości specyficznej dla dostawcy JMS komunikatów wychodzących (kontynuacja)

Nazwa właściwości specyficznej dla dostawcy JMS	Pole MQMD używane do transmisji	Nagłówek	Ustawione przez
JMS_IBM_Report_Discard_Msg	Raport		Obiekt komunikatu
JMS_IBM_MsgType	MsgType		Obiekt komunikatu
Opinie JMS_IBM_Feedback	Opinie		Obiekt komunikatu
Format JMS_IBM_Format	Format		Obiekt komunikatu
Typ JMS_IBM_PutAppl	Typ_aplikacji_wstawiającej		Metoda wysyłania
JMS_IBM_Encoding	Kodowanie		Obiekt komunikatu
Zestaw znaków JMS_IBM_Character_Set	CodedCharacterSetId		Obiekt komunikatu
JMS_IBM_PutDate	PutDate		Metoda wysyłania
JMS_IBM_PutTime	PutTime		Metoda wysyłania
Grupa JMS_IBM_Last_Msg_In_Group	MsgFlags		Obiekt komunikatu

Odwzorowywanie pól nagłówka JMS przy użyciu funkcji send () lub publikowania ()
Te uwagi odnoszą się do odwzorowania pól JMS przy użyciu funkcji send () lub publikowania ().

Miejsce JMSDestination do MQRFH2
Jest ona przechowywana jako łańcuch, który przekształca do postaci szeregowej parametry salient obiektu docelowego, tak aby odbierający JMS mógł odtworzyć równoważny obiekt docelowy. Pole MQRFH2 jest zakodowane jako identyfikator URI (szczegółowe informacje na temat notacji URI zawiera sekcja “Jednolite identyfikatory zasobów (URI)” na stronie 904).

JMSReplyTo do MQMD.ReplyToQ, ReplyToQMgr, MQRFH2
Nazwa kolejki jest kopiowana do MQMD.ReplyToQ , a nazwa menedżera kolejek jest kopiowana do pól ReplyToQMgr. Informacje o rozszerzeniu miejsca docelowego (inne przydatne szczegóły, które są przechowywane w obiekcie docelowym), są kopiowane do pola MQRFH2 . Pole MQRFH2 jest zakodowane jako identyfikator URI (patrz sekcja “Jednolite identyfikatory zasobów (URI)” na stronie 904 , aby uzyskać szczegółowe informacje na temat notacji identyfikatora URI).

JMSDeliveryMode do MQMD.Persistence
Wartość JMSDeliveryMode jest ustawiana przez metodę send () lub publish () lub MessageProducer, chyba że obiekt docelowy przesłoni ją. Wartość JMSDeliveryMode jest odwzorowana na wartość MQMD.Persistence w następujący sposób:

- Wartość JMS PERSISTENT jest równoważna wartości MQPER_PERSISTENT
- Wartość JMS NON_PERSISTENT jest równoważna wartości MQPER_NOT_PERSISTENT

Jeśli właściwość trwałości MQQueue nie jest ustawiona na wartość WMQConstants.WMQ_PER_QDEF, wartość trybu dostarczania jest również zakodowana w tabeli MQRFH2.

JMSExpiration do/z MQMD.Expiry, MQRFH2
JMSExpiration przechowuje czas do utraty ważności (suma czasu bieżącego i czasu życia), podczas gdy menedżer MQMD zapisuje czas życia. Ponadto wartość JMSExpiration jest podana w milisekundach, ale MQMD.Expiry jest w dziesiątych częściach sekundy.

- Jeśli metoda send () ustawia nieograniczony czas na życie, MQMD.Expiry jest ustawiony na wartość MQEI_UNLIMITED, a wartość JMSExpiration nie jest zakodowana w tabeli MQRFH2.
- Jeśli metoda send () ustawia czas na życie, który jest mniejszy niż 214748364.7 sekund (około 7 lat), czas życia jest zapisywany w strukturze MQMD.Expiry, a czas utraty ważności (w milisekundach) jest kodowany jako wartość i8 w MQRFH2.
- Jeśli metoda send () ustawia czas życia większy niż 214748364.7 sekund, MQMD.Expiry jest ustawiony na wartość MQEI_UNLIMITED. Rzeczywisty czas utraty ważności (w milisekundach) jest kodowany jako wartość i8 w MQRFH2.

JMSPriority dla MQMD.Priority
Bezpośrednio odwzoruj wartość właściwości JMSPriority (0-9) na wartość priorytetu MQMD (0-9). Jeśli właściwość JMSPriority jest ustawiona na wartość inną niż domyślna, poziom priorytetu jest również zakodowany w tabeli MQRFH2.

JMSMessageID z MQMD.MessageID
Wszystkie komunikaty wysłane z usługi JMS mają unikalne identyfikatory komunikatów przypisane przez produkt WebSphere MQ. Przypisana wartość jest zwracana w strukturze MQMD.MessageId po wywołaniu MQPUT i jest przekazywany z powrotem do aplikacji w polu JMSMessageID . WebSphere MQ messageId jest 24-bajtową wartością binarną, a JMSMessageID jest łańcuchem. Wartość JMSMessageID składa się z binarnej wartości messageId przekształconej w sekwencję o długości 48 znaków szesnastkowych, poprzedzonej przedrostkiem ID:. Usługa JMS udostępnia wskazówkę, która może zostać ustawiona w celu wyłączenia tworzenia identyfikatorów komunikatów. Ta wskazówka jest ignorowana, a we wszystkich przypadkach przypisany jest unikalny identyfikator. Każda wartość, która jest ustawiana w polu JMSMessageId przed nadpisaniem () jest nadpisywana.

Jeśli wymagane jest określenie wartości MQMD.MessageID, można to zrobić przy użyciu jednego z rozszerzeń JMS produktu WebSphere MQ opisanych w sekcji "Odczytywanie i zapisywanie deskryptora komunikatu z klas produktu WebSphere MQ dla aplikacji JMS" na stronie 921.

JMSTimestamp do MQRFH2
Podczas wysyłania pole JMSTimestamp jest ustawiane zgodnie z zegarem maszyny JVM. Ta wartość jest ustawiana w MQRFH2. Każda wartość, która jest ustawiona w polu JMSTimestamp przed nadpisaniem () jest nadpisywana. Patrz także właściwości JMS_IBM_PutDate i JMS_IBM_PutTime .

JMSType do MQRFH2
Ten łańcuch jest ustawiany w polu MQRFH2 mcd.Type . Jeśli jest on w formacie identyfikatora URI, może mieć również wpływ na pola mcd.Set i mcd.Fmt . Patrz także "Korzystanie z połączenia w czasie rzeczywistym z brokerem produktu WebSphere Event Broker lub WebSphere Message Broker" na stronie 949.

JMSCorrelationID z MQMD.CorrelId, MQRFH2
Element JMSCorrelationID może zawierać jedną z następujących wartości:

Identyfikator komunikatu specyficznego dla dostawcy
Jest to identyfikator komunikatu z wcześniej wysłanego lub odebranego komunikatu, który powinien być łańcuchem o długości 48 małych cyfr szesnastkowych, poprzedzonym przedrostkiem *ID:* . Przedrostek jest usuwany, pozostałe znaki są przekształcane w plik binarny, a następnie są ustawiane w strukturze MQMD.CorrelId , pole. Wartość CorrelId nie jest zakodowana w tabeli MQRFH2.

Dostawca-wartość bajtu rodzimego []
Wartość jest kopiowana do MQMD.CorrelId -dopełnione wartościami pustymi lub obcięte do 24 bajtów, jeśli jest to konieczne. Wartość CorrelId nie jest zakodowana w tabeli MQRFH2.

Łańcuch specyficzny dla aplikacji
Wartość zostanie skopiowana do pliku MQRFH2. Pierwsze 24 bajty łańcucha, w formacie UTF8 , są zapisywane w MQMD.CorrelID.

Odwzorowywanie pól właściwości JMS
Te uwagi odnoszą się do odwzorowania pól właściwości JMS w komunikatach WebSphere MQ .

JMSXUserID z tabeli MQMD UserIdentifier
Wartość JMSXUserID jest ustawiona w przypadku powrotu z wywołania wysyłania.

JMSXAppID z nazwy MQMD PutAppl
JSMXAppID jest ustawiona w przypadku powrotu z wywołania wysyłania.

JMSXGroupID do MQRFH2 (punkt-punkt)
W przypadku komunikatów typu punkt z punktem identyfikator JMSXGroupID jest kopiowany do pola GroupID deskryptora MQMD. Jeśli identyfikator JMSXGroupID zostanie uruchomiony z przedrostkiem ID:, zostanie on przekształcony w plik binarny. W przeciwnym razie jest on zakodowany jako łańcuch UTF8 . Wartość jest dopełniona lub obcięta, jeśli jest to konieczne, do długości 24 bajtów. Flaga MQMF_MSG_IN_GROUP jest ustawiona.

JMSXGroupID do MQRFH2 (publish/subscribe)
W przypadku komunikatów publikowania/subskrypcji wartość JMSXGroupID jest kopiowana do łańcucha MQRFH2 jako łańcuch.

JMSXGroupSeq MQMD MsgSeqLiczba (punkt-do-punktu)
W przypadku komunikatów w trybie punkt z punktem JMSXGroupSeq jest kopiowany do pola MsgSeqNumber w tabeli MQMD. Flaga MQMF_MSG_IN_GROUP jest ustawiona.

JMSXGroupSeq MQMD MsgSeqLiczba (publish/subscribe)
W przypadku komunikatów publikowania/subskrypcji wartość JMSXGroupSeq jest kopiowana do pliku MQRFH2 jako i4.

Odwzorowywanie pól specyficznych dla dostawcy JMS
Poniższe uwagi odnoszą się do odwzorowania pól specyficznych dla dostawcy JMS na komunikaty produktu IBM WebSphere MQ .

JMS_IBM_Report_ < nazwa > do raportu MQMD
Aplikacja JMS może ustawić opcje raportu MQMD przy użyciu następujących właściwości JMS_IBM_Report_XXX. Pojedynczy deskryptor MQMD jest odwzorowywany na kilka właściwości JMS_IBM_Report_XXX. Aplikacja musi ustawić wartość tych właściwości na standardowe stałe MQRO_ IBM WebSphere MQ (uwzględnione w pliku com.ibm.mq.MQC). Na przykład, aby zażądać ChZT z pełnymi danymi, aplikacja musi ustawić wartość JMS_IBM_Report_COD na wartość CMQC.MQRO_COD_WITH_FULL_DATA.

Wyjątek JMS_IBM_Report_Exception

MQRO_EXCEPTION lub
MQRO_EXCEPTION_WITH_DATA lub
MQRO_EXCEPTION_WITH_FULL_DATA

Wygaśnięcie JMS_IBM_Report_Expiration

MQRO_EXPIRATION lub
MQRO_EXPIRATION_WITH_DATA lub
MQRO_EXPIRATION_WITH_FULL_DATA

Plik JMS_IBM_Report_COA

MQRO_COA lub
MQRO_COA_WITH_DATA lub
MQRO_COA_WITH_FULL_DATA

JMS_IBM_Report_COD

MQRO_COD lub
MQRO_COD_WITH_DATA lub
MQRO_COD_WITH_FULL_DATA

Plik JMS_IBM_Report_PAN
MQRO_PAN

Plik JMS_IBM_Report_NAN
MQRO_NAN

JMS_IBM_Report_Pass_Msg_ID
MQRO_PASS_MSG_ID

JMS_IBM_Report_Pass_Correl_ID
MQRO_PASS_CORREL_ID

JMS_IBM_Report_Discard_Msg
MQRO_DISCARD_MSG

JMS_IBM_MsgType z MQMD MsgType
Wartość jest odwzorowywana bezpośrednio na element MQMD MsgType. Jeśli aplikacja nie ustała jawnej wartości właściwości JMS_IBM_MsgType, używana jest wartość domyślna. Ta wartość domyślna jest określana w następujący sposób:

- Jeśli właściwość JMSReplyTo jest ustawiona na miejsce docelowe kolejki produktu IBM WebSphere MQ , parametr MSGType jest ustawiany na wartość MQMT_REQUEST.
- Jeśli właściwość JMSReplyTo nie jest ustawiona lub jest ustawiona na wartość inną niż miejsce docelowe kolejki produktu IBM WebSphere MQ , wartość MsgType jest ustawiana na wartość MQMT_DATAGRAM.

Opinie JMS_IBM_Feedback z MQMD
Wartość jest odwzorowywana bezpośrednio na sprzężenie zwrotne MQMD.

Format JMS_IBM_Format do MQMD
Wartość jest odwzorowywana bezpośrednio na format MQMD.

Kodowanie JMS_IBM_Encoding w MQMD
Jeśli ta właściwość zostanie ustawiona, ta właściwość przesłania kodowanie liczbowe dla kolejki docelowej lub tematu.

JMS_IBM_Character_Set to MQMD CodedCharacterSetId
Jeśli ta właściwość zostanie ustawiona, ta właściwość przesłania właściwość kodowanego zestawu znaków dla kolejki docelowej lub tematu.

JMS_IBM_PutDate z deskryptora MQMD PutDate
Wartość tej właściwości jest ustawiona, podczas wysyłania, bezpośrednio z pola PutDate w strukturze MQMD. Każda wartość ustawiona na właściwość JMS_IBM_PutDate przed nadpisaniem wysyłania. To pole jest łańcuchem o długości ośmiu znaków, w formacie daty IBM WebSphere MQ w formacie RRRRMMDD. Ta właściwość może być używana razem z właściwością JMS_IBM_PutTime w celu określenia czasu umieszczenia komunikatu zgodnie z menedżerem kolejek.

JMS_IBM_PutTime z deskryptora MQMD PutTime
Wartość tej właściwości jest ustawiona, podczas wysyłania, bezpośrednio z pola PutTime w strukturze MQMD. Każda wartość, która jest ustawiana we właściwości JMS_IBM_PutTime przed nadpisaniem wysyłania. To pole zawiera łańcuch o długości ośmiu znaków w formacie IBM WebSphere MQ Time w formacie GGMMSSTH. Ta właściwość może być używana z właściwością JMS_IBM_PutDate w celu określenia czasu umieszczenia komunikatu zgodnie z menedżerem kolejek.

JMS_IBM_Last_Msg_In_Group do MQMD MsgFlags
W przypadku przesyłania komunikatów w trybie punkt z punktem ta wartość boolowska jest odwzorowana na flagę MQMF_LAST_MSG_IN_GROUP w polu MsgFlags deskryptora MQMD. Jest on zwykle używany z właściwościami JMSXGroupID i JMSXGroupSeq w celu wskazania starszej aplikacji produktu IBM WebSphere MQ , której ten komunikat jest ostatnim w grupie. Ta właściwość jest ignorowana w przypadku przesyłania komunikatów w trybie publikowania/subskrypcji.

Odwzorowywanie pól produktu WebSphere MQ na pola JMS (komunikaty przychodzące)
W tych tabelach przedstawiono sposób, w jaki pola nagłówka i właściwości JMS są odwzorowywane na pola MQMD i MQRFH2 w czasie get () lub receive ().

Tabela 116 na stronie 847 pokazuje, w jaki sposób pola nagłówka JMS są odwzorowywane na pola MQMD/MQRFH2 w czasie get () lub receive (). Tabela 117 na stronie 847 Pokazuje, w jaki sposób pola właściwości JMS są odwzorowywane na pola MQMD/MQRFH2 w czasie get () lub receive (). Tabela 118 na stronie 848 pokazuje, w jaki sposób odwzorowywane są właściwości specyficzne dla dostawcy JMS.

Tabela 116. Odwzorowanie pola nagłówka JMS komunikatu przychodzącego

Nazwa pola nagłówka JMS	**Pobrano pole MQMD z**	**Pole MQRFH2 pobrane z**
JMSDestination		jms.Dst lub mqps.Top"1" na stronie 847
JMSDeliveryMode	Trwałość"2" na stronie 847	jms.Dlv"2" na stronie 847
JMSExpiration		jms.Exp
JMSPriority	Priorytet	
JMSMessageID	MsgID	
JMSTimestamp	PutDate"2" na stronie 847 PutTime"2" na stronie 847	jms.Tms"2" na stronie 847
JMSCorrelationID	CorrelId"2" na stronie 847	jms.Cid"2" na stronie 847
JMSReplyTo	Kolejka_zwrotna"2" na stronie 847 Menedżer_kolejek_zwrotnych"2" na stronie 847	jms.Rto"2" na stronie 847
JMSType		mcd.Type, mcd.Set, mcd.Fmt
JMSRedelivered	BackoutCount	

Uwaga:

1. Jeśli ustawiona jest wartość zarówno jms.Dst , jak i mqps.Top , używana jest wartość w jms.Dst .
2. W przypadku właściwości, które mogą mieć wartości pobrane z MQRFH2 lub MQMD, jeśli oba te wartości są dostępne, używane jest ustawienie w tabeli MQRFH2 .
3. Wartość właściwości JMS_IBM_Character_Set jest wartością typu String, która zawiera zestaw znaków Java odpowiadający wartości liczbowej CodedCharacterSetId .

Tabela 117. Odwzorowanie właściwości komunikatu przychodzącego

Nazwa właściwości JMS	**Pobrano pole MQMD z**	**Pole MQRFH2 pobrane z**
JMSXUserID	UserIdentifier	
JMSXAppID	Nazwa_aplikacji_wstawiającej	
JMSXDeliveryCount	BackoutCount	
JMSXGroupID	GroupId"1" na stronie 848	jms.Gid"1" na stronie 848
JMSXGroupSeq	Numer_kolejny_komunikatu"1" na stronie 848	jms.Seq"1" na stronie 848

Tabela 117. Odwzorowanie właściwości komunikatu przychodzącego (kontynuacja)

Nazwa właściwości JMS	Pobrano pole MQMD z	Pole MQRFH2 pobrane z
Uwaga: 1. W przypadku właściwości, które mogą mieć wartości pobrane z MQRFH2 lub MQMD, jeśli oba te wartości są dostępne, używane jest ustawienie w tabeli MQRFH2 . Właściwości są ustawiane na podstawie wartości MQMD tylko wtedy, gdy ustawione są flagi komunikatu MQMF_MSG_IN_GROUP lub MQMF_LAST_MSG_IN_GROUP.		

Tabela 118. Odwzorowanie właściwości JMS specyficzne dla dostawcy komunikatów przychodzących

Nazwa właściwości JMS	Pobrano pole MQMD z	Pole MQRFH2 pobrane z
Wyjątek JMS_IBM_Report_Exception	Raport	
Wygaśnięcie JMS_IBM_Report_Expiration	Raport	
Plik JMS_IBM_Report_COA	Raport	
JMS_IBM_Report_COD	Raport	
Plik JMS_IBM_Report_PAN	Raport	
Plik JMS_IBM_Report_NAN	Raport	
JMS_IBM_Report_ Pass_Msg_ID	Raport	
JMS_IBM_Report_Pass_Correl_ID	Raport	
JMS_IBM_Report_Discard_Msg	Raport	
JMS_IBM_MsgType	MsgType	
Opinie JMS_IBM_Feedback	Opinie	
Format JMS_IBM_Format	Format	
Typ JMS_IBM_PutAppl	Typ_aplikacji_wstawiającej	
JMS_IBM_Encoding "1" na stronie 848	Kodowanie	
JMS_IBM_Character_Set "1" na stronie 848	CodedCharacterSetId	
JMS_IBM_PutDate	PutDate	
JMS_IBM_PutTime	PutTime	
Grupa JMS_IBM_Last_Msg_In_Group	MsgFlags	
1. Ustaw tylko wtedy, gdy komunikat przychodzący jest komunikatem bajtów.		

Wymiana komunikatów między aplikacją JMS a tradycyjną aplikacją WebSphere MQ
W tej sekcji opisano, co się dzieje, gdy aplikacja JMS wymienia komunikaty z tradycyjną aplikacją WebSphere MQ , która nie może przetwarzać nagłówka MQRFH2 .

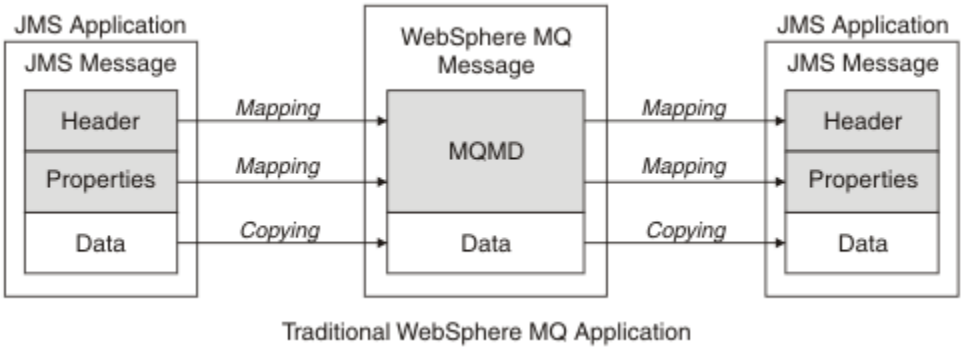
. Rysunek 129 na stronie 849 przedstawia odwzorowanie.

Administrator wskazuje, że aplikacja JMS komunikuje się z tradycyjną aplikacją produktu WebSphere MQ , ustawiając właściwość TARGCLIENT miejsca docelowego na wartość *MQ*. Oznacza to, że nie ma być

tworzony żaden nagłówek MQRFH2 . Jeśli ta opcja nie zostanie wykonana, aplikacja odbierający musi mieć możliwość obsługi nagłówka MQRFH2 .

Odwzorowanie z usługi JMS na deskryptor MQMD skierowane do tradycyjnej aplikacji produktu WebSphere MQ jest takie samo, jak odwzorowanie z usługi JMS na deskryptor MQMD ukierunkowany na aplikację JMS. Jeśli klasy WebSphere MQ classes for JMS odbierają komunikat WebSphere MQ z wartością pola MQMD *Format* ustawioną na wartość inną niż MQFMT_RFH2, dane są odbierane z aplikacji innej niż JMS. Jeśli formatem jest MQFMT_STRING, komunikat jest odbierany jako komunikat tekstowy JMS. W przeciwnym razie zostanie odebrany jako komunikat bajt JMS. Ponieważ nie ma komendy MQRFH2, można odtworzyć tylko te właściwości JMS, które są przesyłane w strukturze MQMD.

Jeśli klasy WebSphere MQ classes for JMS odbierają komunikat, który nie ma nagłówka MQRFH2 , właściwość TARGCLIENT obiektu Queue lub tematu pochodzącego z pola nagłówka JMSReplyTo komunikatu jest domyślnie ustawiona na wartość MQ . Oznacza to, że komunikat odpowiedzi wysłany do kolejki lub tematu również nie ma nagłówka MQRFH2 . Można wyłączyć to zachowanie, w tym nagłówek MQRFH2 w komunikacie odpowiedzi tylko wtedy, gdy oryginalny komunikat ma nagłówek MQRFH2 , ustawiając właściwość TARGCLIENTMATCHING dla fabryki połączeń na wartość NO.



Rysunek 129. Sposób transformowania komunikatów JMS na komunikaty produktu WebSphere MQ bez nagłówka MQRFH2 .

Treść komunikatu JMS
Ten temat zawiera informacje na temat kodowania samego treści komunikatu. Kodowanie zależy od typu komunikatu JMS.

ObjectMessage
Obiekt ObjectMessage jest obiektem serializowanym przez środowisko wykonawcze Java Runtime w normalny sposób.

TextMessage
Obiekt TextMessage jest zakodowanym łańcuchem. W przypadku komunikatu wychodzącego łańcuch jest zakodowany w zestawie znaków określonym przez obiekt docelowy. Wartością domyślną jest kodowanie UTF8 (kodowanie UTF8 rozpoczyna się od pierwszego znaku komunikatu; na początku pola nie ma pola długości). Możliwe jest jednak określenie dowolnego innego zestawu znaków obsługiwanego przez klasy produktu WebSphere MQ dla usługi JMS. Takie zestawy znaków są używane głównie w przypadku wysyłania komunikatu do aplikacji innej niż JMS.

Jeśli zestaw znaków jest zestawem typu double-byte (w tym UTF16), to specyfikacja kodowania liczb całkowitych obiektu docelowego określa kolejność bajtów.

Komunikat przychodzący jest interpretowany przy użyciu zestawu znaków i kodowania, które są określone w samym komunikacie. Te specyfikacje znajdują się w ostatnim nagłówku WebSphere MQ (lub MQMD, jeśli nie ma nagłówków). W przypadku komunikatów JMS ostatni nagłówek to zwykle MQRFH2.

BytesMessage
BytesMessage jest domyślnie sekwencją bajtów zdefiniowaną przez specyfikację JMS 1.0.2 i powiązaną dokumentację Java.

W przypadku komunikatu wychodzącego, który został złożony przez samą aplikację, właściwość kodowania obiektu docelowego może zostać użyta do przesłonięcia kodowania pól liczb całkowitych i zmiennopozycyjnych zawartych w komunikacie. Na przykład można zażądać, aby wartości zmiennopozycyjne były przechowywane w S/390 , a nie w formacie IEEE).

Komunikat przychodzący jest interpretowany przy użyciu kodowania liczbowego określonego w samym komunikacie. Ta specyfikacja znajduje się w ostatnim nagłówku produktu WebSphere MQ (lub w nagłówku MQMD, jeśli nie ma nagłówków). W przypadku komunikatów JMS ostatni nagłówek to zwykle MQRFH2.

Jeśli zostanie odebrana wartość BytesMessage i zostanie ona ponownie wysłana bez modyfikacji, jej treść jest przesyłana bajtem za bajt, ponieważ została odebrana. Właściwość kodowania obiektu docelowego nie ma wpływu na treść. Jedynym obiektem typu łańcuchowego, który może być jawnie wysyłany w BytesMessage jest łańcuch UTF8 . Jest on zakodowany w formacie Java UTF8 i rozpoczyna się od pola o długości 2 bajtów. Właściwość zestawu znaków dla obiektu docelowego nie ma wpływu na kodowanie wychodzącego elementu BytesMessage. Wartość zestawu znaków w przychodzącym komunikacie WebSphere MQ nie ma wpływu na interpretację tego komunikatu jako JMS BytesMessage.

Aplikacje inne niż Java nie rozpoznają kodowania UTF8 Java. Dlatego w przypadku aplikacji JMS w celu wysłania komunikatu BytesMessage , który zawiera dane tekstowe, sama aplikacja musi przekształcić swoje łańcuchy w tablice bajtowe i zapisać te tablice bajtowe w polu BytesMessage.

MapMessage

Element MapMessage jest łańcuchem zawierającym nazwy XML/typ/wartości zakodowane jako:

```
<map>
  <elt name="elementname1" dt="datatype1">value1</elt>
  <elt name="elementname2" dt="datatype2">value2</elt>
  ...
</map>
```

gdzie `datatype` jest jednym z typów danych wymienionych w sekcji Tabela 109 na stronie 838. Domyślnym typem danych jest `string`, a więc atrybut `dt="string"` jest pomijany w przypadku elementów łańcuchowych.

Zestaw znaków używany do kodowania lub interpretowania łańcucha XML, który tworzy treść komunikatu mapy, jest określany zgodnie z regułami, które mają zastosowanie do komunikatu tekstowego.

Wersje klas WebSphere MQ dla usługi JMS w wersjach wcześniejszych niż wersja 5.3 zakodowane są w treści komunikatu odwzorowania w następującym formacie:

```
<map>
  <elementname1 dt="datatype1">value1</elementname1>
  <elementname2 dt="datatype2">value2</elementname2>
  ...
</map>
```

Wersja 5.3 i nowsze wersje klas WebSphere MQ dla usługi JMS mogą interpretować dowolny format, ale wersje klas WebSphere MQ dla usługi JMS w wersjach wcześniejszych niż wersja 5.3 nie mogą interpretować bieżącego formatu.

Jeśli aplikacja musi wysłać komunikaty odwzorowania do innej aplikacji, która używa wersji produktu WebSphere MQ classes for JMS wcześniej niż wersja 5.3, aplikacja wysyłający musi wywołać metodę fabryki połączeń `setMapNameStyle(WMQConstants.WMQ_MAP_NAME_STYLE_COMPATIBLE)` , aby określić, że komunikaty mapy są wysyłane w poprzednim formacie. Domyślnie wszystkie komunikaty mapy są wysyłane w bieżącym formacie.

StreamMessage

Komunikat StreamMessage jest podobny do komunikatu mapy, ale bez nazw elementów:

```
<stream>
  <elt dt="datatype1">value1</elt>
  <elt dt="datatype2">value2</elt>
```

```
  ...
</stream>
```

gdzie `datatype` jest jednym z typów danych wymienionych w sekcji Tabela 109 na stronie 838. Domyślnym typem danych jest `string`, a więc atrybut `dt="string"` jest pomijany w przypadku elementów łańcuchowych.

Zestaw znaków używany do kodowania lub interpretowania łańcucha XML, który składa się z treści StreamMessage , jest określany na podstawie reguł, które mają zastosowanie do komunikatu TextMessage.

Pole MQRFH2.format jest ustawione w następujący sposób:

MQFMT_NONE
dla ObjectMessage, BytesMessagelub komunikatów bez treści.

MQFMT_STRING
dla TextMessage, StreamMessagelub MapMessage.

Konwersja komunikatów JMS

Konwersja danych komunikatów w usłudze JMS jest wykonywana podczas wysyłania i odbierania komunikatów. Produkt WebSphere MQ automatycznie wykonuje większość konwersji danych. Konwertuje dane tekstowe i liczbowe podczas przesyłania komunikatów między aplikacjami JMS. Tekst jest przekształcany podczas wymiany `JMSTextMessage` między aplikacją JMS a aplikacją WebSphere MQ .

Jeśli planowana jest bardziej złożona wymiana komunikatów, interesujące są następujące tematy. Złożone wymiany komunikatów obejmują:

- Przesyłanie komunikatów innych niż tekstowe między aplikacją WebSphere MQ a aplikacją JMS.
- Wymiana danych tekstowych w formacie bajtowym.
- Przekształcanie tekstu w aplikacji.

Dane komunikatu JMS

Konwersja danych jest konieczna do wymiany danych tekstowych i liczbowych między aplikacjami, nawet między dwoma aplikacjami JMS. Wewnętrzna reprezentacja tekstu i liczb musi być zakodowana w taki sposób, aby mogły zostać przesłane w komunikacie. Kodowanie wymusza podjęcie decyzji o tym, w jaki sposób reprezentowane są liczby i tekst. Produkt WebSphere MQ zarządza kodowaniem tekstu i liczb w komunikatach JMS, z wyjątkiem `JMSObjectMessage`, patrz “JMSObjectMessage” na stronie 858. Korzysta ona z trzech atrybutów komunikatu. Trzy atrybuty to: `CodedCharacterSetId`, `Encoding`i `Format`.

Te trzy atrybuty komunikatu są zwykle zapisywane w nagłówku JMS, `MQRFH2`, w polach komunikatu JMS. Jeśli typ komunikatu to `MQ`, a nie typ komunikatu `JMS` , atrybuty są zapisywane w deskryptorze komunikatu `MQMD`. Atrybuty są używane do przekształcania danych komunikatu JMS. Dane komunikatu JMS są przesyłane w części danych komunikatu komunikatu produktu WebSphere MQ .

Właściwości wiadomości JMS

Właściwości komunikatu JMS, takie jak `JMS_IBM_CHARACTER_SET`, są wymieniane w części nagłówka `MQRFH2` komunikatu JMS, o ile komunikat nie został wysłany bez `MQRFH2`. Tylko produkty `JMSTextMessage` i `JMSBytesMessage` mogą być wysyłane bez `MQRFH2`. Jeśli właściwość JMS jest przechowywana jako właściwość komunikatu WebSphere MQ w deskryptorze komunikatu, `MQMD`, jest ona przekształcana w ramach konwersji produktu `MQMD` . Jeśli właściwość JMS jest przechowywana w produkcie `MQRFH2`, jest ona przechowywana w zestawie znaków określonym przez produkt `MQRFH2.NameValueCCSID`. Po wysłaniu lub odebraniu komunikatu właściwości komunikatu są przekształcane do i z ich wewnętrznej reprezentacji w maszynie JVM. Konwersja jest ustawiona na i z zestawu znaków deskryptora komunikatu lub `MQRFH2.NameValueCCSID`. Dane liczbowe są przekształcane w tekst.

Konwersja komunikatów JMS

Poniższe tematy zawierają przykłady i zadania, które są przydatne, jeśli planowane jest wymianę bardziej złożonych komunikatów wymagających konwersji.

Podejścia do konwersji komunikatów JMS
Do projektantów aplikacji JMS otwarto wiele podejść do konwersji danych. Te podejścia nie są wyłączne; niektóre aplikacje mogą używać kombinacji tych podejść. Jeśli aplikacja zmienia tylko tekst lub jest wymieniana tylko z innymi aplikacjami JMS, konwersja danych nie jest zwykle uwzględniana. Konwersja danych jest automatycznie przeprowadzana dla użytkownika przez produkt WebSphere MQ.

Można zadać kilka pytań na temat sposobu podejścia do konwersji komunikatów:

Czy w ogóle trzeba myśleć o konkonwersji komunikatów?
W niektórych przypadkach, takich jak przesyłanie komunikatów JMS do JMS oraz wymiana komunikatów tekstowych z programami IBM WebSphere MQ , program IBM WebSphere MQ automatycznie wykonuje niezbędne konwersje dla użytkownika. Może zaistnieć potrzeba kontrolowania konwersji danych ze względu na wydajność lub wymiany złożonych komunikatów, które mają predefiniowany format. W takich przypadkach należy rozumieć konwersję komunikatów, a także przeczytać poniższe tematy.

Jakie są tam rodzaje konwersji?
Istnieją cztery główne typy konwersji, które zostały wyjaśnione w następujących sekcjach:

1. “Konwersja danych klienta JMS” na stronie 852
2. “Konwersja danych aplikacji” na stronie 853
3. “Konwersja danych menedżera kolejek” na stronie 853
4. “Konwersja danych kanału komunikatów” na stronie 854

Gdzie należy dokonać konwersji?
W sekcji “Wybieranie podejścia do konwersji komunikatów: odbiorca jest dobry” na stronie 855opisano typowe podejście "odbiornik sprawia, że jest on dobry". "Odbiornik sprawia, że dobry" ma również zastosowanie do konwersji danych JMS.

Konwersja danych klienta JMS

Klient JMS[4]konwersja danych to konwersja operacji podstawowych Java i obiektów na bajty w komunikacie JMS w postaci, w której jest ona wysyłana do miejsca docelowego, a następnie ponownie przekształcana, gdy zostanie odebrana. Konwersja danych klienta JMS korzysta z metod klas `JMSMessage` . Metody są wymienione według typu klasy `JMSMessage` w produkcie Tabela 119 na stronie 856.

Konwersja do i z wewnętrznej reprezentacji JVM liczb i tekstu jest wykonywana dla metod odczytu, pobierania, ustawiania i zapisu. Konwersja jest wykonywana po wysłaniu komunikatu i po wywołaniu dowolnej metody odczytu lub pobrania dla odebranego komunikatu.

Strona kodowa i kodowanie liczbowe używane do zapisu lub ustawiania treści komunikatu są definiowane jako atrybuty miejsca docelowego. Strona kodowa miejsca docelowego i kodowanie liczbowe mogą być zmieniane administracyjnie. Aplikacja może także przesłonić stronę kodową miejsca docelowego i kodowanie, ustawiając właściwości komunikatu sterujące zapisaniem lub ustawianiem treści komunikatu.

Aby przekształcić kodowanie liczb, gdy komunikat `JMSBytesMessage` jest wysyłany do miejsca docelowego, które nie jest zdefiniowane jako kodowanie `Native` , przed wysłaniem komunikatu należy ustawić właściwość komunikatu `JMS_IBM_ENCODING` . Jeśli śledzony jest wzorzec "Odbiornik tworzy dobry" lub jeśli komunikaty są wymieniane między aplikacjami JMS, to aplikacja nie musi ustawiać `JMS_IBM_ENCODING`. W większości przypadków można pozostawić właściwość `Encoding` jako `Native`.

[4] "Klient JMS" odwołuje się do klas produktu WebSphere MQ dla usługi JMS, które implementują interfejs JMS, który działa w trybie klienta lub powiązania.

W przypadku komunikatów JMSStreamMessage, JMSMapMessagei JMSTextMessage używane są właściwości identyfikatora zestawu znaków dla miejsca docelowego. Kodowanie jest ignorowane podczas wysyłania, ponieważ liczby są zapisywane w formacie tekstowym. Program użytkowy klienta JMS nie musi ustawiać JMS_IBM_CHARACTER_SET przed wysłaniem komunikatu, jeśli właściwość zestawu znaków miejsca docelowego ma zostać zastosowana.

Aby uzyskać dane w komunikacie, aplikacja wywołuje metody odczytu lub pobierania komunikatu JMS. Metody odwołują się do strony kodowej i kodowania zdefiniowanego w poprzednim nagłówku komunikatu w celu poprawnego utworzenia operacji podstawowych Java i obiektów.

Konwersja danych klienta JMS jest zgodna z potrzebami większości aplikacji JMS, które wymieniają komunikaty między klientem JMS a drugim. Nie są one kodowane jawną konwersją danych. Nie jest używana klasa `java.nio.charset.Charset` , która jest zwykle używana podczas zapisywania tekstu w pliku. Metody `writeString` i `setString` do konwersji są przeznaczone dla użytkownika.

Więcej informacji na temat konwersji danych klienta JMS zawiera sekcja “Konwersja i kodowanie komunikatów klienta JMS” na stronie 865.

Konwersja danych aplikacji

Aplikacja kliencka JMS może przeprowadzić jawną konwersję danych znakowych, korzystając z klasy `java.nio.charset.Charset` ; patrz przykłady w podręczniku Rysunek 132 na stronie 857 i Rysunek 133 na stronie 857 . Dane łańcuchowe są przekształcane w bajty przy użyciu metody `getBytes` i wysyłane w postaci bajtów. Bajty są przekształcane z powrotem w tekst przy użyciu konstruktora `String` , który pobiera tablicę bajtów i `Charset`. Character data is converted using the `encode` and `decode` `Charset` methods. Zwykle komunikat jest wysyłany lub odbierany jako JMSBytesMessage, ponieważ część komunikatu produktu JMSBytesMessage nie zawiera żadnych innych danych niż dane zapisane przez aplikację.[5]. Można także wysyłać i odbierać bajty za pomocą JMSStreamMessage, JMSMapMessagelub JMSObjectMessage.

Brak metod Java do kodowania i dekodowania bajtów, które zawierają dane liczbowe reprezentowane w różnych formatach kodowania. Dane liczbowe są kodowane i dekodowane automatycznie przy użyciu numerycznych metod odczytu i zapisu JMSMessage . Metody odczytu i zapisu korzystają z wartości atrybutu JMS_IBM_ENCODING danych komunikatu.

Typowym zastosowaniem dla konwersji danych aplikacji jest użycie przez klienta JMS sformatowanego komunikatu z aplikacji innej niż JMS lub odeśle go. Sformatowany komunikat zawiera dane tekstowe, liczbowe i bajty uporządkowane według długości pól danych. Jeśli w aplikacji innej niż JMS nie określono formatu komunikatu "MQSTR ", komunikat jest tworzony jako JMSBytesMessage. Aby można było odbierać sformatowane dane komunikatu w JMSBytesMessage , należy wywołać sekwencję metod. Metody muszą być wywoływane w tej samej kolejności, w jakiej pola zostały zapisane w komunikacie. Jeśli pola są liczbowe, należy znać kodowanie i długość danych liczbowych. Jeśli którekolwiek z pól zawierają dane bajtowe lub tekstowe, należy znać długość danych bajtowych w komunikacie. Istnieją dwa sposoby przekształcania sformatowanego komunikatu w obiekt Java, który jest łatwy w użyciu.

1. Narysuj klasę Java odpowiadającą zapisowi, aby hermetyzować odczytywanie i zapisywanie wiadomości. Dostęp do danych w rekordzie jest z get i set metod klasy.
2. Skonstruuj klasę Java odpowiadającą temu rekordowi, rozszerzając klasę `com.ibm.mq.headers` . Dostęp do danych w klasie jest typu akcesorów specyficznych dla typu, `getStringValue(`*`fieldname`*`);`

Patrz sekcja “Wymiana sformatowanego rekordu przy użyciu aplikacji innej niż JMS” na stronie 873.

Konwersja danych menedżera kolejek

W produkcie WebSphere MQ V7.0konwersja strony kodowej może być wykonywana przez menedżer kolejek, gdy program kliencki JMS pobiera komunikat. Konwersja jest taka sama, jak konwersja wykonana dla programu w języku C. Program w języku C ustawia MQGMO_CONVERT jako parametr parametru MQGET `GetMsgOpts` (GetMsg). Patrz Rysunek 131 na stronie 857. Menedżer kolejek wykonuje

[5] Jeden wyjątek: Dane zapisane przy użyciu programu `writeUTF` zaczynają się od pola o długości 2 bajtów.

konwersję dla programu klienckiego JMS odbierającego komunikat, jeśli właściwość miejsca docelowego WMQ_RECEIVE_CONVERSION jest ustawiona na wartość WMQ_RECEIVE_CONVERSION_QMGR, program kliencki JMS może także ustawić właściwość miejsca docelowego. Patrz sekcja Rysunek 130 na stronie 854.

Przed V7.0konwersacje były zawsze wykonywane przez klient JMS. Konwersja danych klienta JMS jest ograniczona do przekształcania sekwencji liczb i tekstu typu i długości znanych klientowi JMS. Nie można przekształcić struktur danych; patrz “Wymiana sformatowanego rekordu przy użyciu aplikacji innej niż JMS” na stronie 873. W wersji V7.0, do czasu wykonania pakietu poprawek 7.0.1.5, jeśli konwersja może zostać wykonana przez menedżer kolejek, jest ona zawsze wykonywana przez menedżer kolejek. Począwszy od wersji 7.0.1.5 domyślne zachowanie konwersji odwraca się do tego samego, co V6.0, a wszystkie konwersje są wykonywane przez klienta JMS. W przypadku produktu 7.0.1.5lub 7.0.1.4 z raportem APAR IC72897można ustawić opcję nowego miejsca docelowego WMQ_RECEIVE_CONVERSION, aby sterować miejscem, w którym wykonywana jest konwersja, oraz WMQ_RECEIVE_CCSID, aby ustawić stronę kodową elementu docelowego. Patrz Rysunek 130 na stronie 854.

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Lub

```
((MQDestination)destination).setReceiveConversion
             (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Rysunek 130. Włącz konwersję danych menedżera kolejek

Podstawowa korzyść z konwersji menedżera kolejek jest dostarczana podczas wymiany komunikatów z aplikacjami innymi niż JMS. Jeśli pole `Format` w komunikacie jest zdefiniowane, a docelowy zestaw znaków lub kodowanie jest inne niż komunikat, menedżer kolejek wykonuje konwersję danych dla aplikacji docelowej, o ile aplikacja zażąda jej. Menedżer kolejek przekształca dane komunikatu sformatowane zgodnie z jednym z predefiniowanych typów komunikatów produktu WebSphere MQ , takich jak nagłówek mostu CICS (`MQCIH`). Jeśli pole `Format` jest zdefiniowane przez użytkownika, menedżer kolejek szuka wyjścia konwersji danych z nazwą udostępnionej w polu `Format` .

Konwersja danych menedżera kolejek jest używana w celu uzyskania najlepszego efektu przy użyciu wzorca projektowego "odbiornika" . Wysyłanie klienta JMS nie jest wymagane do przeprowadzenia konwersji. Program odbierający inny niż JMS opiera się na wyjściu konwersji, aby upewnić się, że komunikat jest dostarczany w wymaganej stronie kodowej i kodowaniu. W przypadku wysyłania klienta JMS i odbiornika innego niż JMS przykład ma zastosowanie do produktu IBM WebSphere MQ pre- and post-V7.0. W przypadku produktu IBM WebSphere MQ V7.0wyjście konwersji może zostać wywołane dla odbierającego programu JMS.

Program obsługi wyjścia konwersji danych można utworzyć przy użyciu programu narzędziowego do obsługi wyjścia konwersji danych **`crtmqcvx`**, aby umożliwić menedżerowi kolejek przekształcenie własnych sformatowanych danych w rekordach. Użytkownik może zbudować własny format rekordu, użyć programu `com.ibm.mq.headers` w celu uzyskania dostępu do niego jako klasy Java, a następnie użyć własnego wyjścia konwersji w celu przekształcenia go. W systemie z/OS program narzędziowy nosi nazwę **CSQUCVX**, a w systemie IBM i, **CVTMQMDTA**. Patrz sekcja “Wymiana sformatowanego rekordu przy użyciu aplikacji innej niż JMS” na stronie 873.

Konwersja danych kanału komunikatów

WebSphere MQ Sender, Server, Cluster-receiver i Cluster-sender channels mają opcję konwersji komunikatów, `CONVERT`. Treść komunikatu może opcjonalnie zostać przekształcona po wysłaniu komunikatu. Konwersja odbywa się w wysyłającym końcu kanału. Definicja dziennika klastra jest używana do automatycznego definiowania odpowiedniego kanału nadawczego klastra.

Konwersja danych za pomocą kanałów komunikatów jest zwykle używana, jeśli nie jest możliwe użycie innych form konwersji.

Wybieranie podejścia do konwersji komunikatów: "odbiorca jest dobry"

Typowym podejściem w projekcie aplikacji WebSphere MQ dla konwersji kodu jest "odbiorca sprawia, że jest on dobry". "Odbiornik sprawia, że dobre" zmniejsza liczbę konwersji komunikatów. Zapobiega to również wystąpieniu nieoczekiwanych błędów kanału, jeśli konwersja komunikatów nie powiedzie się w jakimś pośrednim menedżerze kolejek podczas przesyłania komunikatów. Reguła "odbiornik jest dobry" jest uszkodzona tylko wtedy, gdy istnieje jakiś powód, dla którego odbiorca nie może się dobrze wyrobić. Platforma odbierający może nie mieć odpowiedniego zestawu znaków, na przykład.

"Odbiornik sprawia, że dobry" jest również dobrym ogólnym wskazówkami dla aplikacji klienckich JMS. Jednak w szczególnych przypadkach, konwersja na poprawny zestaw znaków w źródle może być bardziej wydajna. Konwersja z wewnętrznej reprezentacji maszyny JVM musi mieć miejsce, gdy wysyłany jest komunikat zawierający typy tekstowe lub liczbowe. Konwersja na zestaw znaków wymagany przez odbiornik, jeśli odbiornik nie jest klientem JMS, może usunąć potrzebę przeprowadzenia konwersji przez odbiorcę innego niż JMS. Jeśli odbiorcą jest klient JMS, zostanie on ponownie przekształty w celu zdekodowania danych komunikatu i utworzenia podstawowych elementów i obiektów Java.

Różnica między aplikacjami klienckim JMS a aplikacjami napisanych w języku, takim jak C, polega na tym, że środowisko Java musi przeprowadzić konwersję danych. Aplikacja Java musi konwertować liczby i tekst z wewnętrznej reprezentacji do zakodowanego formatu używanego w komunikatach.

Ustawiając miejsce docelowe lub właściwości komunikatu, można ustawić zestaw znaków i kodowanie używane przez produkt WebSphere MQ w celu kodowania liczb i tekstu w komunikatach. Zwykle należy pozostawić zestaw znaków jako `1208` i zakodować jako `Native`.

Produkt WebSphere MQ nie konwertuje tablic bajtów. Aby zakodować łańcuchy i tablice znaków w tablicach bajtów, należy użyć pakietu `java.nio.charset` . `Charset` określa zestaw znaków używany do przekształcania łańcucha lub tablicy znakowej w tablicę bajtów. Można również zdekodować tablicę bajtów w postaci łańcucha lub tablicy znakowej przy użyciu `Charset`. W przypadku kodowania łańcuchów i tablic znakowych nie jest dobrą praktyką poleganie na `java.nio.charset.Charset.defaultCodePage` . Domyślnie `Charset` jest zwykle `windows-1252` w systemie Windows, a `UTF-8` w systemie UNIX. `windows-1252` to zestaw znaków jednobajtowych, a `UTF-8` to zestaw znaków wielobajtowych.

W przypadku wymiany komunikatów z innymi aplikacjami JMS na ogół należy pozostawić właściwości zestawu znaków docelowych i właściwości kodowania w wartościach domyślnych produktu `UTF-8` i `Native` . W przypadku wymiany wiadomości zawierających liczby lub tekst z aplikacją JMS należy wybrać jeden z typów komunikatów `JMSTextMessage`, `JMSStreamMessage`, `JMSMapMessage`lub `JMSObjectMessage` , które są zgodne z danym celem. Nie ma żadnych innych zadań konwersji do wykonania.

W przypadku wymiany komunikatów z aplikacjami innymi niż JMS, które korzystają z formatu rekordu, jest on bardziej skomplikowany. Jeśli cały rekord nie zawiera tekstu i może zostać przesłany jako `JMSTextMessage`, należy zakodować i zdekodować tekst w aplikacji. Ustaw typ komunikatu docelowego na wartość `MQ`, a następnie użyj opcji `JMSBytesMessage` , aby uniknąć dodawania dodatkowych nagłówków i informacji znaczników do danych komunikatu przez klasy produktu IBM WebSphere MQ dla usługi JMS. Metody `JMSBytesMessage` umożliwiają zapisywanie liczb i bajtów, a klasa `Charset` w sposób jawny przekształcają tekst w tablice bajtów. Wybór zestawu znaków może mieć wpływ na wiele czynników:

- Wydajność: Czy można zmniejszyć liczbę konwersji poprzez przekształcenie tekstu w zestaw znaków, który jest używany na największej liczbie serwerów?
- Jednolitość: Prześlij wszystkie komunikaty w tym samym zestawie znaków.
- Bogactwo: jakie zestawy znaków mają wszystkie punkty kodowe, których aplikacje muszą używać?
- Prostota: jednobajtowe zestawy znaków są prostsze do użycia niż zmienne długości i wielobajtowe zestawy znaków.

Patrz sekcja “Wymiana sformatowanego rekordu przy użyciu aplikacji innej niż JMS” na stronie 873. Przykłady konwersji komunikatów wymienianych z aplikacjami innymi niż JMS.

Przykłady

Tabela typów komunikatów i typów konwersji

Tabela 119. Typy komunikatów i typy konwersji

	Typ konwersji			
Typ komunikatu	**Tekstowy**	**Liczbowy**	**Inne**	**Brak**
JMSObjectMessage				getObject setObject
JMSTextMessage	getText setText			
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

```
gmo.Options = MQGMO_WAIT         /* wait for new messages           */
            | MQGMO_NO_SYNCPOINT /* no transaction                  */
            | MQGMO_CONVERT;     /* convert if necessary            */

     while (CompCode != MQCC_FAILED) {
     buflen = sizeof(buffer) - 1; /* buffer size available for GET   */
     memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
     memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
     md.Encoding       = MQENC_NATIVE;
     md.CodedCharSetId = MQCCSI_Q_MGR;

           MQGET(Hcon,          /* connection handle               */
           Hobj,                /* object handle                   */
           &md,                 /* message descriptor              */
           &gmo,                /* get message options             */
           buflen,              /* buffer length                   */
           buffer,              /* message buffer                  */
           &messlen,            /* message length                  */
           &CompCode,           /* completion code                 */
           &Reason);            /* reason code                     */
```

Rysunek 131. Fragment kodu z amqsget0.c

Wysyłanie i odbieranie tekstu w JMSBytesMessage

Kod w programie Rysunek 132 na stronie 857 wysyła łańcuch w polu BytesMessage(BytesMessage). W przypadku prostoty przykład wysyła pojedynczy łańcuch, dla którego JMSTextMessage jest bardziej odpowiedni. Aby otrzymać łańcuch tekstowy w bajtach, zawierający mieszaninę typów, należy znać długość łańcucha w bajtach, nazywanych *TEXT_LENGTH* w produkcie Rysunek 133 na stronie 857. Nawet w przypadku łańcucha o stałej liczbie znaków długość reprezentacji bajtów może być dłuższa.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                  .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Rysunek 132. Wysyłanie String w JMSBytesMessage

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Rysunek 133. Odbieranie String z JMSBytesMessage

Pojęcia pokrewne

Konwersja i kodowanie komunikatów klienta JMS
Na liście znajdują się metody używane do konwersji i kodowania komunikatów klienta JMS, z przykładami kodu każdego typu konwersji.

Konwersja danych menedżera kolejek
Konwersja danych menedżera kolejek zawsze była dostępna dla aplikacji innych niż JMS odbierających komunikaty z klientów JMS. Od wersji V7.0klienty JMS odbierające komunikaty korzystają również z konwersji danych menedżera kolejek. Konwersja danych menedżera kolejek z wersji 7.0.1.5lub 7.0.1.4 z poprawką APAR IC72897jest opcjonalna.

Zadania pokrewne

Wymiana sformatowanego rekordu przy użyciu aplikacji innej niż JMS
Wykonaj kroki proponowane w tym zadaniu, aby zaprojektować i zbudować wyjście konwersji danych, a także aplikację kliencką JMS, która może wymieniać komunikaty z aplikacją inną niż JMS przy użyciu produktu JMSBytesMessage. Wymiana sformatowanego komunikatu z aplikacją inną niż JMS może być podejmowana z wywołaniem wyjścia konwersji danych lub bez niej.

Odsyłacze pokrewne

Typy komunikatów JMS i konwersja
Wybór typu komunikatu wpływa na podejście do konwersji komunikatów. Interakcja konwersji komunikatów i typu komunikatu jest opisana dla typów komunikatów JMS, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagei JMSBytesMessage.

Typy komunikatów JMS i konwersja
Wybór typu komunikatu wpływa na podejście do konwersji komunikatów. Interakcja konwersji komunikatów i typu komunikatu jest opisana dla typów komunikatów JMS, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagei JMSBytesMessage.

JMSObjectMessage

JMSObjectMessage zawiera jeden obiekt i wszystkie obiekty, do których się odwołuje, przekształcony do postaci szeregowej strumienia bajtów przez maszynę JVM. Tekst jest przekształcany do postaci szeregowej w produkcie UTF-8i jest ograniczony do łańcuchów lub tablic znakowych o długości nie większej niż 65534 bajtów. Zaletą produktu JMSObjectMessage jest to, że aplikacje nie są zaangażowane w żadne problemy związane z konwersją danych, o ile tylko te metody i atrybuty są używane tylko w tych metodach. Produkt JMSObjectMessage udostępnia konwersję danych dla obiektów złożonych bez programisty aplikacji rozważając, jak zakodować obiekt w komunikacie. Wadą korzystania z produktu JMSObjectMessage jest możliwość wymiany tylko z innymi aplikacjami JMS. Wybierając jeden z innych typów komunikatów JMS, możliwe jest wymianę komunikatów JMS z aplikacjami innymi niż JMS.

“Wysyłanie i odbieranie JMSObjectMessage” na stronie 861 przedstawia obiekt String , który jest wymieniany w komunikacie.

Aplikacja kliencka JMS może odbierać JMSObjectMessage tylko w komunikacie, który ma treść w stylu JMS. Miejsce docelowe musi określać treść stylu JMS.

JMSTextMessage

JMSTextMessage zawiera pojedynczy łańcuch tekstowy. Gdy wysyłany jest komunikat tekstowy, tekst Format jest ustawiany na wartość "MQSTR ", WMQConstants.MQFMT_STRING. CodedCharacterSetId tekstu jest ustawiany na identyfikator kodowanego zestawu znaków zdefiniowany dla miejsca docelowego. Tekst jest zakodowany w produkcie CodedCharacterSetId przez produkt WebSphere MQ. Pola CodedCharacterSetId i Format są albo ustawione w deskryptorze komunikatu, MQMD, albo w polach JMS w MQRFH2. Jeśli komunikat jest zdefiniowany jako mający styl treści komunikatu produktu WMQ_MESSAGE_BODY_MQ lub styl treści nie został określony, ale miejscem docelowym jest WMQ_TARGET_DEST_MQ, to pola deskryptora komunikatu są ustawiane. W przeciwnym razie komunikat ma RFH2 , a pola są ustawiane w stałej części MQRFH2.

Aplikacja może przesłonić identyfikator kodowanego zestawu znaków zdefiniowany dla miejsca docelowego. Musi ona ustawić właściwość komunikatu JMS_IBM_CHARACTER_SET na identyfikator kodowanego zestawu znaków; patrz przykład w sekcji “Wysyłanie i odbieranie JMSTextmessage” na stronie 861.

Gdy klient JMS wywoła konwersję menedżera kolejek metody consumer.receive , jest ona opcjonalna. Konwersję menedżera kolejek można włączyć, ustawiając właściwość miejsca docelowego WMQ_RECEIVE_CONVERSION na wartość WMQ_RECEIVE_CONVERSION_QMGR. Menedżer kolejek przekształca komunikat tekstowy z JMS_IBM_CHARACTER_SET określonego dla komunikatu przed przesłaniem komunikatu do klienta JMS. Zestaw znaków przekształconego komunikatu to 1208, UTF-8, chyba że miejsce docelowe ma inną wartość WMQ_RECEIVE_CCSID. CodedCharacterSetId

w komunikacie, który odwołuje się do JMSTextMessage , jest aktualizowany do docelowego identyfikatora zestawu znaków. Tekst jest zdekodowany z docelowego zestawu znaków w kodzie Unicode za pomocą metody getText . Patrz przykład w sekcji “Wysyłanie i odbieranie JMSTextmessage” na stronie 861.

JMSTextMessage może być wysyłany w treści komunikatu w stylu MQbez nagłówka MQRFH2 JMS. Wartość atrybutów miejsca docelowego, WMQ_MESSAGE_BODY i WMQ_TARGET_DEST określa styl treści komunikatu, o ile nie zostanie przesłonięty przez aplikację.
Aplikacja może przesłonić wartości ustawione w miejscu docelowym przez wywołanie funkcji destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ) lub destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ).

Jeśli produkt JMSTextMessage jest wysyłany z treścią stylu MQ , wysyłając je do miejsca docelowego o nazwie WMQ_MESSAGE_BODY ustawionej na wartość WMQ_MESSAGE_BODY_MQ, nie można odbierać go jako JMSTextMessage z tego samego miejsca docelowego. Wszystkie komunikaty odebrane z miejsca docelowego z wartością WMQ_MESSAGE_BODY ustawioną na wartość WMQ_MESSAGE_BODY_MQ są odbierane jako JMSBytesMessage. W przypadku próby odebrania komunikatu jako JMSTextMessage powoduje to wyjątek ClassCastException: com.ibm.jms.JMSBytesMessage cannot be cast to javax.jms.TextMessage.

Uwaga: Tekst w JMSBytesMessage nie jest przekształcany przez klienta JMS. Klient może odbierać tekst tylko w komunikacie jako tablicę bajtów. Jeśli konwersja menedżera kolejek jest włączona, tekst jest przekształcany przez menedżer kolejek, ale klient JMS musi nadal odbierać go jako tablicę bajtów w JMSBytesMessage.

Zwykle lepszym zastosowaniem jest użycie właściwości WMQ_TARGET_DEST w celu określenia, czy produkt JMSTextMessage jest wysyłany z użyciem stylu treści MQ czy JMS. Następnie można odebrać komunikat z miejsca docelowego, które ma wartość WMQ_TARGET_DEST ustawioną na wartość WMQ_TARGET_DEST_MQ lub WMQ_TARGET_DEST_JMS. Wartość WMQ_TARGET_DEST nie ma wpływu na odbiornik.

JMSMapMessage i JMSStreamMessage

Te dwa typy komunikatów JMS są podobne. Typy podstawowe można odczytywać i zapisywać na komunikatach przy użyciu metod opartych na interfejsach DataInputStream i DataOutputStream . Patrz sekcja “Tabela typów komunikatów i typów konwersji” na stronie 863. Szczegółowe informacje są opisane w sekcji “Konwersja i kodowanie komunikatów klienta JMS” na stronie 865. Każda operacja podstawowa jest oznaczona; patrz “Treść komunikatu JMS” na stronie 849.

Dane liczbowe są odczytywaniei zapisywane w komunikacie zakodowanym jako tekst XML. Do właściwości miejsca docelowego nie jest odwołanie JMS_IBM_ENCODING. Dane tekstowe są traktowane w taki sam sposób, jak tekst w JMSTextMessage. Jeśli użytkownik miał przejrzeć treść komunikatu utworzoną na podstawie przykładu w programie Rysunek 138 na stronie 862, wszystkie dane komunikatu będą w kodzie EBCDIC, ponieważ zostały wysłane z wartością zestawu znaków 37.

Istnieje możliwość wysłania wielu elementów w JMSMapMessage lub JMSStreamMessage.

Poszczególne elementy danych można pobierać według nazwy z JMSMapMessagelub według pozycji z JMSStreamMessage. Każdy element jest dekodowany, gdy metoda get lub read jest wywoływana przy użyciu wartości CodedCharacterSetId zapisanej w komunikacie. Jeśli metoda użyta do pobrania elementu zwraca inny typ do typu, który został wysłany, typ jest przekształcany. Jeśli typ nie może zostać przekształcony, zgłaszany jest wyjątek. Szczegółowe informacje na ten temat zawiera sekcja Klasa JMSStreamMessage . Przykład w sekcji “Wysyłanie danych w serwerach JMSStreamMessage i JMSMapMessage” na stronie 862 ilustruje konwersję typu i pobieranie treści JMSMapMessage z sekwencji.

Pole MQRFH2.format dla partycji JMSMapMessage i JMSStreamMessage jest ustawione na wartość "MQSTR ". Jeśli właściwość miejsca docelowego WMQ_RECEIVE_CONVERSION jest ustawiona na wartość WMQ_RECEIVE_CONVERSION_QMGR, dane komunikatu są przekształcane przez menedżer kolejek przed wysłaniem do klienta JMS. MQRFH2.CodedCharacterSetId komunikatu to WMQ_RECEIVE_CCSID miejsca docelowego. MQRFH2.Encoding to

Native. If WMQ_RECEIVE_CONVERSION is WMQ_RECEIVE_CONVERSION_CLIENT_MSG the CodedCharacterSetId and Encoding of the MQRFH2 is the value set by the sender.

Aplikacja kliencka JMS może odbierać JMSMapMessage lub JMSStreamMessage tylko w komunikacie, który ma treść w stylu JMS, oraz z miejsca docelowego, które nie określa treści w stylu MQ .

JMSBytesMessage

JMSBytesMessage może zawierać wiele typów podstawowych. Typy podstawowe można odczytywać i zapisywać na komunikatach przy użyciu metod opartych na interfejsach DataInputStream i DataOutputStream . Patrz sekcja “Tabela typów komunikatów i typów konwersji” na stronie 863. Szczegółowe informacje są opisane w sekcji “Typy komunikatów JMS i konwersja” na stronie 858.

Kodowanie danych liczbowych w komunikacie jest sterowane przez wartość JMS_IBM_ENCODING , która jest ustawiona przed zapisaniem danych liczbowych w JMSBytesMessage. Aplikacja może przesłonić domyślne kodowanie produktu Native zdefiniowane dla produktu JMSBytesMessage , ustawiając właściwość komunikatu JMS_IBM_ENCODING.

Dane tekstowe można odczytywać i zapisywać w programie UTF-8 przy użyciu produktów readUTF i writeUTF, a także w kodzie Unicode za pomocą metod readChar i writeChar . Nie ma metod, które używają produktu CodedCharacterSetId. Alternatywnie klient JMS może kodować i dekodować tekst w bajtach przy użyciu klasy Charset . Przenosi ona bajty między maszyną JVM i komunikatem bez klas produktu WebSphere MQ dla usługi JMS wykonujących konwersję; patrz sekcja “Wysyłanie i odbieranie tekstu w JMSBytesMessage” na stronie 862.

Produkt JMSBytesMessage wysyłany do aplikacji MQ jest zwykle wysyłany w treści komunikatu w stylu MQbez nagłówka MQRFH2 JMS. Jeśli jest ona wysyłana do aplikacji JMS, styl treści komunikatu jest zazwyczaj JMS. Wartość atrybutów miejsca docelowego, WMQ_MESSAGE_BODY i WMQ_TARGET_DEST określa styl treści komunikatu, o ile nie zostanie przesłonięty przez aplikację. Aplikacja może przesłonić wartości ustawione w miejscu docelowym przez wywołanie funkcji destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ) lub destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ).

Jeśli produkt JMSBytesMessage jest wysyłany z treścią stylu MQ , komunikat można odebrać z miejsca docelowego, które definiuje styl treści komunikatu MQ lub JMS. W przypadku wysyłania JMSBytesMessage z treścią stylu JMS konieczne jest odebranie komunikatu z miejsca docelowego, które definiuje styl treści komunikatu JMS. W przeciwnym razie produkt MQRFH2 będzie traktowany jako część danych komunikatu użytkownika, co może nie być tym, czego się spodziewa.

Niezależnie od tego, czy komunikat ma styl treści MQ , czy JMS, ustawienie WMQ_TARGET_DESTnie ma wpływu na sposób jego odebrania.

Komunikat może zostać później przetransformowany przez menedżer kolejek, jeśli dla danych komunikatu podano Format , a konwersja danych menedżera kolejek jest włączona. Nie należy używać pola formatu dla niczego innego niż określenie formatu danych komunikatu lub pozostawienie pustego pola, MQConstants.MQFMT_NONE

Istnieje możliwość wysłania wielu elementów w JMSBytesMessage. Każda pozycja liczbowa jest przekształcana po wysłaniu komunikatu z użyciem kodowania zdefiniowanego dla komunikatu.

Istnieje możliwość pobrania poszczególnych elementów danych z programu JMSBytesMessage. Wywołaj metody odczytu w tej samej kolejności, w której wywołano metody zapisu w celu utworzenia komunikatu. Każdy element numeryczny jest przekształcany, gdy komunikat jest wywoływany przy użyciu wartości Encoding zapisanej w komunikacie.

W przeciwieństwie do produktów JMSMapMessage i JMSStreamMessage, produkt JMSBytesMessage zawiera tylko dane zapisane przez aplikację. W danych komunikatu nie są zapisywane żadne dodatkowe dane, takie jak znaczniki XML używane do definiowania elementów w serwerach JMSMapMessage i JMSStreamMessage. Z tego powodu należy użyć programu JMSBytesMessage do przesyłania komunikatów sformatowanych dla innych aplikacji.

Konwersja między JMSBytesMessage a DataInputStream i DataOutputStream jest przydatna w niektórych aplikacjach. Kod oparty na przykładzie, “Odczytywanie i zapisywanie komunikatów przy

użyciu produktów DataInputStream i DataOutputStream” na stronie 862, jest niezbędny do korzystania z pakietu `com.ibm.mq.header` z usługą JMS.

Przykłady

Wysyłanie i odbieranie JMSObjectMessage

```
ObjectMessage omo = session.createObjectMessage();
omo.setObject(new String("A string"));
producer.send(omo);
...
ObjectMessage omi = (ObjectMessage)consumer.receive();
System.out.println((String)omi.getObject());
...
A string
```

Rysunek 134. Wysyłanie i odbieranie JMSObjectMessage

Wysyłanie i odbieranie JMSTextmessage

Komunikat tekstowy nie może zawierać tekstu w różnych zestawach znaków. W przykładzie przedstawiono tekst w różnych zestawach znaków, które są wysyłane w dwóch różnych komunikatach.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Rysunek 135. Wyślij wiadomość tekstową w zestawie znaków zdefiniowanym przez miejsce docelowe

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Rysunek 136. Wyślij wiadomość tekstową w ccsid 37

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Rysunek 137. Odbierz wiadomość tekstową

Wysyłanie danych w serwerach JMSStreamMessage i JMSMapMessage

```
StreamMessage smo = session.createStreamMessage();
smo.writeString("256");
smo.writeInt(512);
smo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
producer.send(smo);
...
MapMessage mmo = session.createMapMessage();
mmo.setString("First", "256");
mmo.setInt("Second", 512);
mmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
producer.send(mmo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println("Stream: First as float "   + smi.readFloat() +
                         " Second as String " + smi.readString());
...
Stream: First as float: 256.0, Second as String: 512
...
MapMessage mmi = (MapMessage)consumer.receive();
System.out.println("Map: Second as String " + mmi.getString("Second") +
                      " First as double "  + mmi.getDouble("First"));
...
Map: Second as String: 512, First as double: 256.0
```

Rysunek 138. Wysyłanie danych w serwerach JMSStreamMessage i JMSMapMessage

Wysyłanie i odbieranie tekstu w JMSBytesMessage

Kod w programie Rysunek 139 na stronie 862 wysyła łańcuch w polu BytesMessage(BytesMessage). W przypadku prostoty przykład wysyła pojedynczy łańcuch, dla którego JMSTextMessage jest bardziej odpowiedni. Aby otrzymać łańcuch tekstowy w bajtach, zawierający mieszaninę typów, należy znać długość łańcucha w bajtach, nazywanych *TEXT_LENGTH* w produkcie Rysunek 140 na stronie 862. Nawet w przypadku łańcucha o stałej liczbie znaków długość reprezentacji bajtów może być dłuższa.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                  .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Rysunek 139. Wysyłanie String w JMSBytesMessage

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Rysunek 140. Odbieranie String z JMSBytesMessage

Odczytywanie i zapisywanie komunikatów przy użyciu produktów DataInputStream i DataOutputStream

Kod w programie Rysunek 141 na stronie 863 tworzy JMSBytesMessage przy użyciu DataOutputStream.

```
ByteArrayOutputStream bout = new ByteArrayOutputStream();
DataOutputStream dout = new DataOutputStream(bout);
BytesMessage messageOut = prod.session.createBytesMessage();
// messageOut.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
//                           ((MQDestination) (prod.destination)).getIntProperty
//                           (WMQConstants.WMQ_ENCODING));
int ccsidOut = (((MQDestination)prod.destination).getIntProperty(WMQConstants.WMQ_CCSID));
String codePageOut = CCSID.getCodepage(ccsidOut);
dout.writeInt(ccsidOut);
dout.write(codePageOut.getBytes(codePageOut));
messageOut.writeBytes(bout.toByteArray());
producer.send(messageOut);
```

Rysunek 141. Wysyłanie JMSBytesMessage za pomocą DataOutputStream

Instrukcja, która ustawia właściwość JMS_IBM_ENCODING , jest przekształcana w komentarz. Instrukcja jest poprawna, jeśli zapis jest bezpośrednio do komunikatu JMSBytesMessage, ale nie ma wpływu na zapis do DataOutputStream. Liczby zapisywane w pliku DataOutputStream są kodowane w kodowaniu produktu Native . Ustawienie JMS_IBM_ENCODING nie ma żadnego efektu.

Kod w programie Rysunek 142 na stronie 863 otrzymuje JMSBytesMessage za pomocą DataInputStream.

```
static final int ccsidIn_SIZE = (Integer.SIZE)/8;
...
connection.start();
BytesMessage messageIn = (BytesMessage) consumer.receive();
int messageLength = new Long(messageIn.getBodyLength()).intValue();
byte [] bin = new byte[messageLength];
messageIn.readBytes(bin, messageLength);
DataInputStream din = new DataInputStream(new ByteArrayInputStream(bin));
int ccsidIn = din.readInt();
byte [] codePageByte = new byte[messageLength - ccsidIn_SIZE];
din.read(codePageByte, 0, codePageByte.length);
System.out.println("CCSID " + ccsidIn + " code page " + new String(codePageByte,
                    messageIn.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET)));
```

Rysunek 142. Otrzymywanie JMSBytesMessage przy użyciu DataInputStream

Strona kodowa jest drukowana za pomocą właściwości strony kodowej danych komunikatu wejściowego, JMS_IBM_CHARACTER_SET. Na wejściu JMS_IBM_CHARACTER_SET jest to strona kodowa języka Java, a nie liczbowy identyfikator kodowanego zestawu znaków.

Tabela typów komunikatów i typów konwersji

Tabela 120. Typy komunikatów i typy konwersji

	Typ konwersji			
Typ komunikatu	**Tekstowy**	**Liczbowy**	**Inne**	**Brak**
JMSObjectMessage				getObject setObject
JMSTextMessage	getText setText			

Tabela 120. Typy komunikatów i typy konwersji (kontynuacja)

	Typ konwersji			
Typ komunikatu	**Tekstowy**	**Liczbowy**	**Inne**	**Brak**
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

Pojęcia pokrewne

Podejścia do konwersji komunikatów JMS

Do projektantów aplikacji JMS otwarto wiele podejść do konwersji danych. Te podejścia nie są wyłączne; niektóre aplikacje mogą używać kombinacji tych podejść. Jeśli aplikacja zmienia tylko tekst lub jest wymieniana tylko z innymi aplikacjami JMS, konwersja danych nie jest zwykle uwzględniana. Konwersja danych jest automatycznie przeprowadzana dla użytkownika przez produkt WebSphere MQ.

Konwersja i kodowanie komunikatów klienta JMS

Na liście znajdują się metody używane do konwersji i kodowania komunikatów klienta JMS, z przykładami kodu każdego typu konwersji.

Konwersja danych menedżera kolejek

Konwersja danych menedżera kolejek zawsze była dostępna dla aplikacji innych niż JMS odbierających komunikaty z klientów JMS. Od wersji V7.0klienty JMS odbierające komunikaty korzystają również z konwersji danych menedżera kolejek. Konwersja danych menedżera kolejek z wersji 7.0.1.5lub 7.0.1.4 z poprawką APAR IC72897jest opcjonalna.

Zadania pokrewne

Wymiana sformatowanego rekordu przy użyciu aplikacji innej niż JMS
Wykonaj kroki proponowane w tym zadaniu, aby zaprojektować i zbudować wyjście konwersji danych, a także aplikację kliencką JMS, która może wymieniać komunikaty z aplikacją inną niż JMS przy użyciu produktu JMSBytesMessage. Wymiana sformatowanego komunikatu z aplikacją inną niż JMS może być podejmowana z wywołaniem wyjścia konwersji danych lub bez niej.

Konwersja i kodowanie komunikatów klienta JMS
Na liście znajdują się metody używane do konwersji i kodowania komunikatów klienta JMS, z przykładami kodu każdego typu konwersji.

Konwersja i kodowanie występują, gdy operacje podstawowe Java lub obiekty są odczytywanielub zapisywane do i z komunikatów JMS. Konwersja jest nazywana konwersją danych klienta JMS w celu odróżnienia jej od konwersji danych menedżera kolejek i konwersji danych aplikacji. Konwersja odbywa się ściśle w przypadku, gdy dane są odczytyane z komunikatu JMS lub zapisywane w komunikacie JMS. Tekst jest przekształcany w wewnętrzną reprezentację 16-bitową Unicode i z wewnętrznej reprezentacji Unicode.[6]do zestawu znaków używanego dla tekstu w komunikatach. Dane liczbowe są przekształcane w typ liczbowy podstawowy Java i są przekształcane w kodowanie zdefiniowane dla komunikatu. To, czy konwersja jest wykonywana, oraz jaki typ konwersji jest wykonywany, zależy od typu komunikatu JMS oraz operacji odczytu lub zapisu.

Tabela 121 na stronie 865 kategoryzuje metody odczytu i zapisu dla różnych typów komunikatów JMS według typu wykonywanego konwersji. Typy konwersji są opisane w tekście po tabeli.

Tabela 121. Typy komunikatów i typy konwersji

	Typ konwersji			
Typ komunikatu	**Tekstowy**	**Liczbowy**	**Inne**	**Brak**
JMSObjectMessage				getObject setObject
JMSTextMessage	getText setText			
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar

[6] Niektóre reprezentacje Unicode wymagają więcej niż 16 bitów. Patrz odwołanie do języka Java SE.

Tabela 121. Typy komunikatów i typy konwersji (kontynuacja)				
	Typ konwersji			
Typ komunikatu	**Tekstowy**	**Liczbowy**	**Inne**	**Brak**
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

Tekstowy

Domyślną wartością CodedCharacterSetId dla miejsca docelowego jest 1208, UTF-8. Domyślnie tekst jest przekształcany z formatu Unicode i wysyłany jako łańcuch tekstowy produktu UTF-8 . Po odebraniu tekst jest przekształcany z zakodowanego zestawu znaków w komunikacie odebranym przez klienta do kodu Unicode.

Metody setText i writeString przekształcają tekst z kodu Unicode w zestaw znaków zdefiniowany dla miejsca docelowego. Aplikacja może przesłonić docelowy zestaw znaków, ustawiając właściwość komunikatu JMS_IBM_CHARACTER_SET. JMS_IBM_CHARACTER_SET, podczas wysyłania komunikatu musi być to liczbowy identyfikator kodowanego zestawu znaków[7].

Fragmenty kodu w programie "Wysyłanie i odbieranie JMSTextmessage" na stronie 869 wysyłają dwa komunikaty. Jeden z nich jest wysyłany w zestawie znaków zdefiniowanym dla miejsca docelowego, a drugi w zestawie znaków 37, zdefiniowanym przez aplikację.

Metody getText i readString przekształcają tekst w komunikacie z zestawu znaków zdefiniowanego w komunikacie na kod Unicode. Metody korzystają ze strony kodowej zdefiniowanej we właściwości komunikatu JMS_IBM_CHARACTER_SET. Strona kodowa jest odwzorowywana z programu MQRFH2.CodedCharacterSetId , chyba że komunikat jest komunikatem typu MQi nie zawiera MQRFH2. Jeśli komunikat jest komunikatem typu MQ, bez elementu MQRFH2, strona kodowa jest odwzorowywana z produktu MQMD.CodedCharacterSetId.

Fragment kodu w programie Rysunek 147 na stronie 869 odbiera komunikat, który został wysłany do miejsca docelowego. Tekst w komunikacie jest przekształcany ze strony kodowej IBM037 z powrotem na kod Unicode.

[7] Podczas odbierania komunikatu JMS_IBM_CHARACTER_SET jest to nazwa strony kodowej języka Java Charset .

Uwaga: Prostym sposobem sprawdzenia, czy tekst jest przekształcany w kodowany zestaw znaków 37, jest użycie programu WebSphere MQ Explorer. Należy przejrzeć kolejkę i wyświetlić właściwości komunikatu przed jego pobraniem.

Przeciwstawny fragment kodu w produkcie Rysunek 146 na stronie 869 z niepoprawnym fragmentem kodu w produkcie Rysunek 143 na stronie 867. W niepoprawnym fragmencie kodu łańcuch tekstowy jest przekształcany dwukrotnie, raz przez aplikację, a następnie ponownie przez WebSphere MQ.

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText(new String("Sent in EBCDIC character set 37".getBytes(CCSID.getCodepage(37))));
producer.send(tmo);
```

Rysunek 143. Niepoprawna konwersja strony kodowej

Metoda `writeUTF` przekształca tekst z kodu Unicode na wartość 1208, UTF-8. Łańcuch tekstowy jest poprzedzony o długości 2 bajtów. Maksymalna długość łańcucha tekstowego to 65534 bajtów. Metoda `readUTF` odczytuje element w komunikacie zapisanej w metodzie `writeUTF` . Odczytuje dokładnie liczbę bajtów zapisanych w metodzie `writeUTF` .

Liczbowy

Domyślnym kodowaniem liczbowym dla miejsca docelowego jest `Native`. Stała kodowana `Native` dla języka Java ma wartość 273, `x'00000111'`, która jest taka sama dla wszystkich platform. Po odebraniu numery w komunikacie są poprawnie przekształcane w liczbowe podstawowe elementy Java. Transformacja korzysta z kodowania zdefiniowanego w komunikacie i typu zwróconego przez metodę odczytu.

Metoda wysyłania przekształca liczby dodawane do komunikatu przez `set` i `write` do kodowania liczbowego zdefiniowanego dla miejsca docelowego. Kodowanie docelowe może zostać przesłonięte dla komunikatu przez aplikację ustawiające właściwość komunikatu `JMS_IBM_ENCODING`, na przykład:

```
message.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
WMQConstants.WMQ_ENCODING_INTEGER_REVERSED);
```

Metody numeryczne `get` i `read` przekształcają liczby w komunikacie z kodowania liczbowego zdefiniowanego w komunikacie. Te liczby są przekształcane do typu określonego za pomocą metody `read` lub `get` . Patrz sekcja Właściwość ENCODING . Metody używają kodowania zdefiniowanego w produkcie `JMS_IBM_ENCODING`. Kodowanie jest odwzorowywane z programu `MQRFH2.Encoding` , o ile komunikat nie jest komunikatem typu MQi nie zawiera `MQRFH2`. Jeśli komunikat jest komunikatem typu MQ, w którym nie ma `MQRFH2`, metody używają kodowania zdefiniowanego w produkcie `MQMD.Encoding`.

W przykładzie w produkcie Rysunek 148 na stronie 869 jest wyświetlana aplikacja koduje liczbę w formacie docelowym i wysyła ją w pliku `JMSStreamMessage`. Należy porównać przykład z Rysunek 148 na stronie 869 z przykładem w Rysunek 149 na stronie 870. Różnica polega na tym, że wartość `JMS_IBM_ENCODING` musi być ustawiona w `JMSBytesMessage`.

Uwaga: Prostym sposobem sprawdzenia, czy numer jest zakodowany poprawnie, jest użycie programu WebSphere MQ Explorer. Należy przejrzeć kolejkę i wyświetlić właściwości komunikatu przed jego konsumowaniu.

Inne

The `boolean` methods encode `true` and `false` as `x'01'` and `x'00'` in a `JMSByteMessage`, `JMSStreamMessage`, and `JMSMapMessage`.

Metody UTF kodują i dekodują kod Unicode do łańcuchów tekstowych UTF-8 . Łańcuchy są ograniczone do mniej niż 65536 znaków i są poprzedzone polem o długości 2 bajtów.

Metody obiektu zawijają podstawowe typy jako obiekty. Typy liczbowe i tekstowe są kodowane lub przekształcane tak, jakby typy podstawowe zostały odczytane lub zapisane przy użyciu metod numerycznych i tekstowych.

Brak

Metody `readByte`, `readBytes`, `readUnsignedByte`, `writeByte`i `writeBytes` otrzymują lub umieszczają pojedyncze bajty lub tablice bajtów, między aplikacją a komunikatem bez konwersji. Metody `readChar` i `writeChar` dostają i umieszczają 2 bajtowe znaki Unicode między aplikacją a komunikatem bez konwersji.

Za pomocą metod `readBytes` i `writeBytes` aplikacja może przeprowadzić konwersję własnego punktu kodowego, tak jak to jest w sekcji "Wysyłanie i odbieranie tekstu w JMSBytesMessage" na stronie 870.

Produkt WebSphere MQ nie wykonuje żadnej konwersji strony kodowej w kliencie, ponieważ jest to komunikat `JMSBytesMessage`, a także dlatego, że używane są metody `readBytes` i `writeBytes` . Jeśli jednak bajty reprezentują tekst, należy się upewnić, że strona kodowa używana przez aplikację jest zgodna z kodowanym zestawem znaków miejsca docelowego. Komunikat może zostać ponownie przekształcony przez wyjście konwersji menedżera kolejek. Inną możliwością jest to, że odbierający program kliencki JMS może stosować się do konwencji przekształcania tablic bajtowych reprezentujących tekst w komunikacie na łańcuchy lub znaki przy użyciu właściwości `JMS_IBM_CHARACTER_SET` w komunikacie.

W tym przykładzie klient używa docelowego zestawu znaków określonego dla jego konwersji:

```
bytes.writeBytes("In the destination code page".getBytes(
      CCSID.getCodepage(((MQDestination) destination)
            .getIntProperty(WMQConstants.WMQ_CCSID))));
```

Alternatywnie, klient mógł wybrać stronę kodową, a następnie ustawić odpowiedni kodowany zestaw znaków w właściwości `JMS_IBM_CHARACTER_SET` komunikatu. Klasy produktu WebSphere MQ dla języka Java używają zmiennej `JMS_IBM_CHARACTER_SET` do ustawienia pola `CodedCharacterSetId` we właściwościach JMS w produkcie `MQRFH2`lub w deskryptorze komunikatu `MQMD`:

```
String codePage = CCSID.getCodepage(37);
message.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage);[8]
```

Jeśli tablica bajtów jest zapisywana w produkcie `JMSStringMessage` lub `JMSMapMessage`, klasy WebSphere MQ classes for JMS nie wykonują konwersji danych, ponieważ bajty są wpisywane jako dane szesnastkowe, a nie jako tekst w `JMSStringMessage` i `JMSMapMessage`.

Jeśli bajty reprezentują znaki w aplikacji, należy wziąć pod uwagę, które punkty kodowe mają zostać odczytane i zapisane w komunikacie. Kod w programie Rysunek 144 na stronie 868 jest zgodny z konwencją używania docelowego zestawu znaków kodowanych. Jeśli łańcuch zostanie utworzony przy użyciu domyślnego zestawu znaków dla maszyny JVM, zawartość bajtów będzie zależeć od platformy. Wirtualna maszyna języka Java w systemie Windows ma zwykle domyślną wartość `Charset` w systemach `windows-1252`i UNIX `UTF-8`. Wymiana między systemami Windows i UNIX wymaga wybrania jawnej strony kodowej w celu wymiany tekstu w postaci bajtów.

```
StreamMessage smo = producer.session.createStreamMessage();
smo.writeBytes("123".getBytes(CCSID.getCodepage(((MQDestination) destination)
            .getIntProperty(WMQConstants.WMQ_CCSID))));
```

Rysunek 144. Zapisywanie bajtów reprezentujących łańcuch w JMS`StreamMessage` przy użyciu docelowego zestawu znaków

[8] `SetStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage) currently accepts only numeric character set identifiers.`

Przykłady

Wysyłanie i odbieranie JMSTextmessage

Komunikat tekstowy nie może zawierać tekstu w różnych zestawach znaków. W przykładzie przedstawiono tekst w różnych zestawach znaków, które są wysyłane w dwóch różnych komunikatach.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Rysunek 145. Wyślij wiadomość tekstową w zestawie znaków zdefiniowanym przez miejsce docelowe

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Rysunek 146. Wyślij wiadomość tekstową w ccsid 37

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Rysunek 147. Odbierz wiadomość tekstową

Przykłady kodowania

Przykłady pokazujące liczbę wysyłanej w kodowaniu definicji dla miejsca docelowego. Należy zauważyć, że należy ustawić właściwość JMS_IBM_ENCODING dla JMSBytesMessage na wartość określoną dla miejsca docelowego.

```
StreamMessage smo = session.createStreamMessage();
smo.writeInt(256);
producer.send(smo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println(smi.readInt());
...
256
```

Rysunek 148. Wysyłanie numeru przy użyciu kodowania miejsca docelowego w produkcie JMSStreamMessage

```
BytesMessage bmo = session.createBytesMessage();
bmo.writeInt(256);
int encoding = ((MQDestination) (destination)).getIntProperty
    (WMQConstants.WMQ_ENCODING)
bmo.setIntProperty(WMQConstants.JMS_IBM_ENCODING, encoding);
producer.send(bmo);
...
BytesMessage bmi = (BytesMessage)consumer.receive();
System.out.println(bmi.readInt());
...
256
```

Rysunek 149. Wysyłanie numeru przy użyciu kodowania miejsca docelowego w produkcie JMSBytesMessage

Wysyłanie i odbieranie tekstu w JMSBytesMessage

Kod w programie Rysunek 150 na stronie 870 wysyła łańcuch w polu BytesMessage(BytesMessage). W przypadku prostoty przykład wysyła pojedynczy łańcuch, dla którego JMSTextMessage jest bardziej odpowiedni. Aby otrzymać łańcuch tekstowy w bajtach, zawierający mieszaninę typów, należy znać długość łańcucha w bajtach, nazywanych *TEXT_LENGTH* w produkcie Rysunek 151 na stronie 870. Nawet w przypadku łańcucha o stałej liczbie znaków długość reprezentacji bajtów może być dłuższa.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                  .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Rysunek 150. Wysyłanie String w JMSBytesMessage

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Rysunek 151. Odbieranie String z JMSBytesMessage

Pojęcia pokrewne

Podejścia do konwersji komunikatów JMS
Do projektantów aplikacji JMS otwarto wiele podejść do konwersji danych. Te podejścia nie są wyłączne; niektóre aplikacje mogą używać kombinacji tych podejść. Jeśli aplikacja zmienia tylko tekst lub jest wymieniana tylko z innymi aplikacjami JMS, konwersja danych nie jest zwykle uwzględniana. Konwersja danych jest automatycznie przeprowadzana dla użytkownika przez produkt WebSphere MQ.

Konwersja danych menedżera kolejek
Konwersja danych menedżera kolejek zawsze była dostępna dla aplikacji innych niż JMS odbierających komunikaty z klientów JMS. Od wersji V7.0klienty JMS odbierające komunikaty korzystają również z konwersji danych menedżera kolejek. Konwersja danych menedżera kolejek z wersji 7.0.1.5lub 7.0.1.4 z poprawką APAR IC72897jest opcjonalna.

Zadania pokrewne

Wymiana sformatowanego rekordu przy użyciu aplikacji innej niż JMS
Wykonaj kroki proponowane w tym zadaniu, aby zaprojektować i zbudować wyjście konwersji danych, a także aplikację kliencką JMS, która może wymieniać komunikaty z aplikacją inną niż JMS przy użyciu produktu JMSBytesMessage. Wymiana sformatowanego komunikatu z aplikacją inną niż JMS może być podejmowana z wywołaniem wyjścia konwersji danych lub bez niej.

Odsyłacze pokrewne

Typy komunikatów JMS i konwersja
Wybór typu komunikatu wpływa na podejście do konwersji komunikatów. Interakcja konwersji komunikatów i typu komunikatu jest opisana dla typów komunikatów JMS, `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`i `JMSBytesMessage`.

Konwersja danych menedżera kolejek
Konwersja danych menedżera kolejek zawsze była dostępna dla aplikacji innych niż JMS odbierających komunikaty z klientów JMS. Od wersji V7.0klienty JMS odbierające komunikaty korzystają również z konwersji danych menedżera kolejek. Konwersja danych menedżera kolejek z wersji 7.0.1.5lub 7.0.1.4 z poprawką APAR IC72897jest opcjonalna.

Menedżer kolejek może przekształcić dane znakowe i liczbowe w dane komunikatu przy użyciu wartości `CodedCharacterSetId`, `Encoding`i `Format` ustawionych dla danych komunikatu. W przypadku aplikacji innych niż JMS możliwość konwersji była zawsze dostępna, ustawiając opcję `GetMessageOption`(Opcja GetMessage), `GMO_CONVERT`. Możliwość konwersji menedżera kolejek nie była dostępna dla aplikacji JMS odbierającej komunikat do wersji V7.0.

Istnieje możliwość użycia konwersji menedżera kolejek (przed wersją V7.0) z aplikacją kliencką JMS, która wysyła komunikat. Klient JMS buduje sformatowany rekord, ustawia atrybuty `CodedCharacterSetId`, `Encoding`i `Format` odpowiadające danym umieszczonym w komunikacie. Aplikacja odbierający inne niż JMS odczytuje komunikat przy użyciu programu `GMO_CONVERT`i powoduje wywołanie wyjścia konwersji danych napisanych przez użytkownika. Wyjście konwersji danych jest biblioteką współużytkowaną, która ma nazwę ustawioną w polu `Format` .

Od wersji V7.0menedżer kolejek jest w stanie przekształcić komunikaty wysyłane do klientów JMS. Od 7.0.0.0 do 7.0.1.4 włącznie, konwersja menedżera kolejek jest zawsze wywoływana dla klientów JMS. Z 7.0.1.5lub z 7.0.1.4 z raportem APAR IC72897 konwersja menedżera kolejek jest sterowana przez ustawienie właściwości miejsca docelowego `WMQ_RECEIVE_CONVERSION`, na `WMQ_RECEIVE_CONVERSION_QMGR`lub `WMQ_RECEIVE_CONVERSION_CLIENT_MSG`. `WMQ_RECEIVE_CONVERSION_CLIENT_MSG` jest ustawieniem domyślnym, zgodnym z działaniem produktu WebSphere MQ V6.0, który nie obsługuje konwersji danych menedżera kolejek dla klientów JMS. Aplikacja może zmienić ustawienie miejsca docelowego:

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Lub

```
((MQDestination)destination).setReceiveConversion
             (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Rysunek 152. Włącz konwersję danych menedżera kolejek

Konwersję danych menedżera kolejek dla klienta JMS przyjmuje się, gdy klient wywołuje metodę `consumer.receive` . Dane tekstowe są domyślnie przekształcane w format UTF-8 (1208). Kolejne metody odczytu i pobierania tekstu są dekodowane w otrzymywanych danych z UTF-8, tworząc operacje podstawowe Java w ich wewnętrznym kodowaniu Unicode. UTF-8 nie jest jedynym docelowym zestawem znaków z konwersji danych menedżera kolejek. Aby wybrać inny identyfikator CCSID, należy ustawić właściwość miejsca docelowego `WMQ_RECEIVE_CCSID` .

Aplikacja może również zmienić ustawienie miejsca docelowego, na przykład ustawiając go na wartość 437, DOS-US:

```
((MQDestination)destination).setIntProperty
            (WMQConstants.WMQ_RECEIVE_CCSID, 437);
```

Lub

```
((MQDestination)destination).setReceiveCCSID(437);
```

Rysunek 153. Ustaw docelowy zestaw znaków dla konwersji menedżera kolejek

Przyczyna zmiany parametru WMQ_RECEIVE_CCSID jest wyspecjalizowana. Wybrany identyfikator CCSID nie różni się od obiektów tekstowych utworzonych w maszynie JVM. Jednak niektóre maszyny JVM, na niektórych platformach, mogą nie być w stanie obsłużyć konwersji z identyfikatora CCSID tekstu w komunikacie na Unicode. Opcja ta umożliwia wybór identyfikatora CCSID dla dowolnego tekstu dostarczanego do klienta w komunikacie. Niektóre platformy klienckie JMS miały problemy z tekstem komunikatu dostarczonym w formacie UTF-8.

Kod JMS jest odpowiednikiem pogrubionego tekstu w kodzie C w produkcie Rysunek 154 na stronie 872.

```
gmo.Options = MQGMO_WAIT         /* wait for new messages           */
            | MQGMO_NO_SYNCPOINT /* no transaction                  */
            | MQGMO_CONVERT;     /* convert if necessary            */

     while (CompCode != MQCC_FAILED) {
     buflen = sizeof(buffer) - 1; /* buffer size available for GET   */
     memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
     memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
     md.Encoding       = MQENC_NATIVE;
     md.CodedCharSetId = MQCCSI_Q_MGR;

           MQGET(Hcon,          /* connection handle               */
           Hobj,                /* object handle                   */
           &md,                 /* message descriptor              */
           &gmo,                /* get message options             */
           buflen,              /* buffer length                   */
           buffer,              /* message buffer                  */
           &messlen,            /* message length                  */
           &CompCode,           /* completion code                 */
           &Reason);            /* reason code                     */
```

Rysunek 154. Fragment kodu z amqsget0.c

Uwaga:

Konwersja menedżera kolejek jest wykonywana tylko dla danych komunikatu, które mają znany format produktu WebSphere MQ . MQSTR, lub MQCIH są przykładami znanych formatów, które są predefiniowane. Znanym formatem może być również format zdefiniowany przez użytkownika, o ile użytkownik dostarczył wyjście konwersji danych.

Komunikaty tworzone w postaci JMSTextMessage, JMSMapMessage i JMSStreamMessagemają format MQSTR i mogą być przekształcane przez menedżer kolejek.

Pojęcia pokrewne

Podejścia do konwersji komunikatów JMS
Do projektantów aplikacji JMS otwarto wiele podejść do konwersji danych. Te podejścia nie są wyłączne; niektóre aplikacje mogą używać kombinacji tych podejść. Jeśli aplikacja zmienia tylko tekst lub jest wymieniana tylko z innymi aplikacjami JMS, konwersja danych nie jest zwykle uwzględniana. Konwersja danych jest automatycznie przeprowadzana dla użytkownika przez produkt WebSphere MQ.

Konwersja i kodowanie komunikatów klienta JMS
Na liście znajdują się metody używane do konwersji i kodowania komunikatów klienta JMS, z przykładami kodu każdego typu konwersji.

“Wywoływanie wyjścia konwersji danych” na stronie 426

Wyjście konwersji danych to wyjście napisane przez użytkownika, które odbiera sterowanie podczas przetwarzania wywołania MQGET.

Zadania pokrewne

Wymiana sformatowanego rekordu przy użyciu aplikacji innej niż JMS
Wykonaj kroki proponowane w tym zadaniu, aby zaprojektować i zbudować wyjście konwersji danych, a także aplikację kliencką JMS, która może wymieniać komunikaty z aplikacją inną niż JMS przy użyciu produktu `JMSBytesMessage`. Wymiana sformatowanego komunikatu z aplikacją inną niż JMS może być podejmowana z wywołaniem wyjścia konwersji danych lub bez niej.

Odsyłacze pokrewne

Typy komunikatów JMS i konwersja
Wybór typu komunikatu wpływa na podejście do konwersji komunikatów. Interakcja konwersji komunikatów i typu komunikatu jest opisana dla typów komunikatów JMS, `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`ei `JMSBytesMessage`.

Wymiana sformatowanego rekordu przy użyciu aplikacji innej niż JMS
Wykonaj kroki proponowane w tym zadaniu, aby zaprojektować i zbudować wyjście konwersji danych, a także aplikację kliencką JMS, która może wymieniać komunikaty z aplikacją inną niż JMS przy użyciu produktu `JMSBytesMessage`. Wymiana sformatowanego komunikatu z aplikacją inną niż JMS może być podejmowana z wywołaniem wyjścia konwersji danych lub bez niej.

Zanim rozpoczniesz

Użytkownik może zaprojektować prostsze rozwiązanie do wymiany komunikatów za pomocą aplikacji innej niż JMS przy użyciu `JMSTextMessage`. Wyeliminuj tę możliwość przed krokami kroków w tym zadaniu.

O tym zadaniu

Klient JMS jest łatwiejszy do zapisu, jeśli nie jest on zaangażowany w szczegóły formatowania komunikatów JMS wymienianych z innymi klientami JMS. Jeśli typem komunikatu jest `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`lub `JMSObjectMessage`, produkt WebSphere MQ będzie wyglądać po szczegółach formatowania komunikatu. Produkt WebSphere MQ zajmuje się różnicami w stronach kodowych i kodowaniem liczbowym na różnych platformach.

Tych typów komunikatów można używać do wymiany komunikatów z aplikacjami innymi niż JMS. Aby to zrobić, należy zrozumieć, w jaki sposób te komunikaty są tworzone przez klasy WebSphere MQ dla usługi JMS. Użytkownik może mieć możliwość zmodyfikowania aplikacji innej niż JMS w celu zinterpretowania komunikatów; patrz “Odwzorowywanie komunikatów JMS na komunikaty produktu WebSphere MQ” na stronie 833.

Zaletą korzystania z jednego z tych typów komunikatów jest programowanie klienta JMS, które nie zależy od typu aplikacji, z którą jest wymieniany. Wadą jest to, że może wymagać modyfikacji innego programu, a użytkownik może nie być w stanie zmienić innego programu.

Alternatywnym podejściem jest napisanie aplikacji klienckiej JMS, która może zajmować się istniejącymi formatami komunikatów. Często istniejące komunikaty mają stały format i zawierają mieszaninę niesformatowanych danych, tekstu i liczb. Wykonaj kroki opisane w tym zadaniu oraz przykład klienta JMS w produkcie “Zapisywanie klas w celu hermetyzowania układu rekordów w JMSBytesMessage” na stronie 877, jako punkt wyjścia do budowania klienta JMS, który może wymieniać sformatowane rekordy z aplikacjami innymi niż JMS.

Procedura

1. Zdefiniuj układ rekordu lub użyj jednej z predefiniowanych klas nagłówków WebSphere MQ .

   Informacje na temat obsługi predefiniowanych nagłówków produktu WebSphere MQ można znaleźć w sekcji Obsługa nagłówków komunikatów produktu WebSphere MQ.

   Rysunek 155 na stronie 875 jest przykładem zdefiniowanego przez użytkownika układu rekordu o stałej długości, który może być przetwarzany przez program narzędziowy do konwersji danych.

2. Utwórz wyjście konwersji danych.

   Postępuj zgodnie z instrukcjami w sekcji Pisanie programu obsługi wyjścia konwersji danych , aby zapisać wyjście konwersji danych.

   Aby wypróbować przykład w programie “Zapisywanie klas w celu hermetyzowania układu rekordów w JMSBytesMessage” na stronie 877, należy podać nazwę wyjścia konwersji danych MYRECORD.
3. Napisz klasy Java, aby hermetyzować układ rekordów, a także wysyłać i odbierać rekordy. Dostępne są dwa sposoby podejścia:

   - Napisz klasę do tego odczytu i zapisze JMSBytesMessage , który zawiera rekord; patrz “Zapisywanie klas w celu hermetyzowania układu rekordów w JMSBytesMessage” na stronie 877.
   - Napisz klasę rozszerzająca com.ibm.mq.header.Header , aby zdefiniować strukturę danych dla rekordu. Patrz sekcja Tworzenie klas dla nowych typów nagłówków.
4. Zdecyduj, jaki kodowany zestaw znaków ma być używany do wymiany komunikatów.

   Patrz “Wybieranie podejścia do konwersji komunikatów: odbiorca jest dobry” na stronie 855.
5. Skonfiguruj miejsce docelowe w celu wymiany komunikatów typu MQ, bez nagłówka MQRFH2 JMS.

   Zarówno miejsce docelowe wysyłania, jak i odbierania musi być skonfigurowane do wymiany komunikatów typu MQ. Tego samego miejsca docelowego można użyć zarówno do wysyłania, jak i do odbierania.

   Aplikacja może przesłonić właściwość treści komunikatu docelowego:

   ```
   ((MQDestination)destination).setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ```

   Przykład w produkcie “Zapisywanie klas w celu hermetyzowania układu rekordów w JMSBytesMessage” na stronie 877 przesłania właściwość treści komunikatu docelowego, upewniając się, że wysyłany jest komunikat w stylu MQ.
6. Testowanie rozwiązania za pomocą aplikacji JMS i innych niż JMS

   Przydatne narzędzia do testowania wyjścia konwersji danych to:

   - Przykładowy program amqsgetc0.c jest przydatny do testowania odbierania komunikatu wysłanego przez klienta JMS. Zapoznaj się z sugerowanymi modyfikacjami, aby użyć przykładowego nagłówka, RECORD.h, w sekcji Rysunek 156 na stronie 876. Wraz z modyfikacjami program amqsgetc0.c otrzymuje komunikat wysłany przez przykładowego klienta JMS TryMyRecord.java; patrz “Zapisywanie klas w celu hermetyzowania układu rekordów w JMSBytesMessage” na stronie 877.
   - Przykładowy program do przeglądania WebSphere MQ amqsbcg0.cjest przydatny do sprawdzania treści nagłówka komunikatu, nagłówka JMS, MQRFH2oraz treści komunikatu.
   - Program **rfhutil** , który jest wcześniej dostępny w pliku SupportPac IH03, umożliwia przechwytywanie komunikatów testowych i przechowywanie ich w plikach, a następnie ich użycie w celu kierowania przepływów komunikatów. Komunikaty wyjściowe mogą być również odczytyowane i wyświetlane w różnych formatach. Formaty te obejmują dwa typy kodu XML, a także dopasowanie do struktury copybook języka COBOL. Dane mogą być w formacie EBCDIC lub ASCII. Nagłówek RFH2 może zostać dodany do komunikatu przed wysłaniem komunikatu.

   W przypadku próby odebrania komunikatów za pomocą zmodyfikowanego przykładowego programu amqsgetc0.c i uzyskania błędu o kodzie przyczyny 2080należy sprawdzić, czy komunikat ma MQRFH2. W przypadku modyfikacji założono, że komunikat został wysłany do miejsca docelowego, które nie określa MQRFH2.

Przykłady

```
struct RECORD { MQCHAR StrucID[4];
                MQLONG Version;
                MQLONG StructLength;
                MQLONG Encoding;
                MQLONG CodeCharSetId;
                MQCHAR Format[8];
                MQLONG Flags;
                MQCHAR RecordData[32];
    };
```

Rysunek 155. RECORD.h

- Zadeklaruj strukturę danych produktu RECORD.h

```
struct tagRECORD {
   MQCHAR4    StrucId;
   MQLONG     Version;
   MQLONG     StrucLength;
   MQLONG     Encoding;
   MQLONG     CCSID;
   MQCHAR8    Format;
   MQLONG     Flags;
   MQCHAR32   RecordData;
 };
typedef struct tagRECORD RECORD;
   typedef RECORD MQPOINTER PRECORD;
   RECORD record;
   PRECORD pRecord = &(record);
```

- Zmodyfikuj wywołanie MQGET tak, aby korzystało z RECORD,

  1. Przed modyfikacją:

```
    MQGET(Hcon,            /* connection handle     */
          Hobj,            /* object handle         */
          &md,             /* message descriptor    */
          &gmo,            /* get message options   */
          buflen,          /* buffer length         */
          buffer,          /* message buffer        */
          &messlen,        /* message length        */
          &CompCode,       /* completion code       */
          &Reason);        /* reason code           */
```

  2. Po modyfikacji:

```
    MQGET(Hcon,            /* connection handle     */
          Hobj,            /* object handle         */
          &md,             /* message descriptor    */
          &gmo,            /* get message options   */
          sizeof(RECORD),  /* buffer length         */
          pRecord,         /* message buffer        */
          &messlen,        /* message length        */
          &CompCode,       /* completion code       */
          &Reason);        /* reason code           */
```

- Zmień instrukcję print,

  1. Od:

```
buffer[messlen] = '\0';            /* add terminator          */
printf("message <%s>\n", buffer);
```

  2. to:

```
/* buffer[messlen] = '\0';            add terminator          */
printf("ccsid <%d>, flags <%d>, message <%32.32s>\n \0",
        md.CodedCharSetId, record.Flags, record.RecordData);
```

Rysunek 156. Zmodyfikuj plik amqsget0.c .

Pojęcia pokrewne

Podejścia do konwersji komunikatów JMS
Do projektantów aplikacji JMS otwarto wiele podejść do konwersji danych. Te podejścia nie są wyłączne; niektóre aplikacje mogą używać kombinacji tych podejść. Jeśli aplikacja zmienia tylko tekst lub jest wymieniana tylko z innymi aplikacjami JMS, konwersja danych nie jest zwykle uwzględniana. Konwersja danych jest automatycznie przeprowadzana dla użytkownika przez produkt WebSphere MQ.

Konwersja i kodowanie komunikatów klienta JMS
Na liście znajdują się metody używane do konwersji i kodowania komunikatów klienta JMS, z przykładami kodu każdego typu konwersji.

Konwersja danych menedżera kolejek

Konwersja danych menedżera kolejek zawsze była dostępna dla aplikacji innych niż JMS odbierających komunikaty z klientów JMS. Od wersji V7.0klienty JMS odbierające komunikaty korzystają również z konwersji danych menedżera kolejek. Konwersja danych menedżera kolejek z wersji 7.0.1.5lub 7.0.1.4 z poprawką APAR IC72897jest opcjonalna.

Odsyłacze pokrewne

Typy komunikatów JMS i konwersja
Wybór typu komunikatu wpływa na podejście do konwersji komunikatów. Interakcja konwersji komunikatów i typu komunikatu jest opisana dla typów komunikatów JMS, `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`i `JMSBytesMessage`.

Program narzędziowy do tworzenia kodu wyjścia konwersji

Zapisywanie klas w celu hermetyzowania układu rekordów w JMSBytesMessage
Celem tego zadania jest eksplorowanie, na przykład, sposobu łączenia konwersji danych i stałego układu rekordów w `JMSBytesMessage`. W ramach czynności można utworzyć niektóre klasy Java w celu wymiany przykładowej struktury rekordu w `JMSBytesMessage`. Istnieje możliwość zmodyfikowania przykładu w celu zapisania klas w celu wymiany innych struktur akt.

`JMSBytesMessage` jest najlepszym wyborem typu komunikatu JMS w celu wymiany rekordów mieszanych typów danych z programami innymi niż JMS. Nie ma żadnych dodatkowych danych wstawianych do treści komunikatu przez dostawcę JMS. Jest to zatem najlepszy wybór typu komunikatu, który ma być używany, jeśli program kliencki JMS współdziała z istniejącym programem IBM WebSphere MQ . Główne wyzwanie związane z używaniem `JMSBytesMessage` jest zgodne z kodowaniem i zestawem znaków oczekiwanym przez inny program. Rozwiązaniem jest utworzenie klasy, która hermetyzuje rekord. Klasa, która hermetyzuje odczytywanie i zapisywanie `JMSBytesMessage`dla konkretnego typu akt, ułatwia wysyłanie i odbieranie rekordów w formacie stałym w programie JMS. Przechwytywanie ogólnych aspektów interfejsu w klasie abstrakcyjnej, wiele z rozwiązań może być ponownie wykorzystane dla różnych formatów rekordu. Różne formaty rekordów mogą być implementowane w klasach, które rozszerzają abstrakcyjną klasę rodzajową.

Alternatywnym podejściem jest rozszerzenie klasy `com.ibm.mq.headers.Header` . Klasa `Header` ma metody, takie jak `addMQLONG`, w celu budowania formatu rekordu w bardziej deklaratywny sposób. Wadą korzystania z klasy `Header` jest pobieranie i ustawianie atrybutów przy użyciu bardziej skomplikowanego interfejsu interpretacyjnego. Oba podejścia powodują w dużej ilości tyle samo kodu aplikacji.

Program `JMSBytesMessage` może hermetyzować tylko jeden format, oprócz `MQRFH2`, w jednym komunikacie, o ile każdy rekord nie używa tego samego formatu, kodowanego zestawu znaków i kodowania. Format, kodowanie i zestaw znaków `JMSBytesMessage` to właściwości wszystkich komunikatów następujących po `MQRFH2`. Przykład ten jest zapisany przy założeniu, że `JMSBytesMessage` zawiera tylko jeden rekord użytkownika.

Zanim rozpoczniesz

1. Poziom umiejętności: należy zapoznać się z programowaniem Java i JMS. Nie są dostępne żadne instrukcje dotyczące konfigurowania środowiska programistycznego Java. Zaleca się napisanie programu do wymiany serwerów `JMSTextMessage`, `JMSStreamMessage`lub `JMSMapMessage`. Użytkownik może następnie zapoznać się z różnicami w wymianie komunikatu przy użyciu `JMSBytesMessage`.
2. W tym przykładzie wymagany jest produkt IBM WebSphere MQ V7.0.
3. Przykład został utworzony przy użyciu perspektywy Java środowiska roboczego Eclipse . Wymaga ona środowiska JRE 6.0 lub nowszego. Aby utworzyć i uruchomić klasy Java, można użyć perspektywy Java w programie IBM WebSphere MQ Explorer. Można również użyć własnego środowiska programistycznego Java.
4. Korzystanie z programu IBM WebSphere MQ Explorer powoduje skonfigurowanie środowiska testowego oraz debugowanie, prostsze niż korzystanie z programów narzędziowych wiersza komend.

O tym zadaniu

Użytkownik jest prowadzony przez utworzenie dwóch klas: RECORD i MyRecord. Razem te dwie klasy hermetyzują rekord o stałym formacie. Mają one metody pobierania i ustawiania atrybutów. Metoda get odczytuje rekord z JMSBytesMessage , a metoda put zapisuje rekord w JMSBytesMessage.

Zadaniem tego zadania nie jest utworzenie klasy jakości produkcyjnej, którą można ponownie wykorzystać. Można użyć przykładów w zadaniu, aby rozpocząć działanie na własnych zajęciach. Celem tego zadania jest udostępnienie uwag dotyczących wskazówek, przede wszystkim dotyczących używania zestawów znaków, formatów i kodowania, w przypadku korzystania z produktu JMSBytesMessage. Każdy krok tworzenia klas jest wyjaśniony, a aspekty korzystania z produktu JMSBytesMessage, które czasami są pomijane, są opisane.

Klasa RECORD jest abstrakcyjna i definiuje niektóre wspólne pola dla rekordu użytkownika. Wspólne pola są modelowane na standardowym układzie nagłówka IBM WebSphere MQ , w którym znajduje się program catcher, wersja i pole długości. Pola kodowania, zestawu znaków i formatu, które znajdują się w wielu nagłówkach IBM WebSphere MQ , są pomijane. Inny nagłówek nie może być zgodny z formatem zdefiniowanym przez użytkownika. Klasa MyRecord , która rozszerza klasę RECORD , robi to poprzez dosłownie rozszerzanie rekordu z dodatkowymi polami użytkownika. Program JMSBytesMessage, utworzony przez klasy, może być przetwarzany przez wyjście konwersji danych menedżera kolejek.

Produkt “Klasy używane do uruchamiania przykładu” na stronie 884 zawiera pełną listę produktów RECORD i MyRecord. Zawiera również listy dodatkowych klas "rusztowań" w celu przetestowania produktów RECORD i MyRecord. Dodatkowe klasy to:

TryMyRecord
Główny program do testowania produktów RECORD i MyRecord.

EndPoint
Klasa abstrakcyjna, która hermetyzuje połączenie JMS, miejsce docelowe i sesję w jednej klasie. Jego interfejs spełnia tylko potrzeby testowania klas RECORD i MyRecord . Nie jest to ustalony wzorzec projektowania do pisania aplikacji JMS.

Uwaga: Klasa Endpoint zawiera ten wiersz kodu po utworzeniu miejsca docelowego:

```
((MQDestination)destination).setReceiveConversion
                            (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

W produkcie V7.0z wersji V7.0.1.5konieczne jest włączenie konwersji menedżera kolejek. Jest ona domyślnie wyłączona. W wersji V7.0, konwersja do V7.0.1.4 menedżera kolejek jest domyślnie włączona, a ten wiersz kodu powoduje wystąpienie błędu.

MyProducer i MyConsumer
Klasy, które rozszerzają EndPointi tworzą MessageConsumer i MessageProducer, połączone i gotowe do akceptowania żądań.

Razem wszystkie klasy składają się na kompletną aplikację, z którą można zbudować i eksperymentować, aby zrozumieć, jak używać konwersji danych w JMSBytesMessage.

Procedura

1. Utwórz klasę abstrakcyjną, aby hermetyzować standardowe pola w nagłówku IBM WebSphere MQ , przy użyciu konstruktora domyślnego. Później rozszerz klasę, aby dostosować nagłówek do wymagań.

```
public abstract class RECORD implements Serializable {
    private static final long serialVersionUID = -1616617232750561712L;
    protected final static int UTF8 = 1208;
    protected final static int MQLONG_LENGTH = 4;
    protected final static int RECORD_STRUCT_ID_LENGTH = 4;
    protected final static int RECORD_VERSION_1 = 1;
    protected final String RECORD_STRUCT_ID = "BLNK";
    protected final String RECORD_TYPE = "BLANK   ";
    private String structID = RECORD_STRUCT_ID;
    private int version = RECORD_VERSION_1;
    private int structLength = RECORD_STRUCT_ID_LENGTH + MQLONG_LENGTH * 2;
    private int headerEncoding = WMQConstants.WMQ_ENCODING_NATIVE;
    private String headerCharset = "UTF-8";
```

```
    private String headerFormat = RECORD_TYPE;

    public RECORD() {
        super();
    }
```

Uwaga:

a. Atrybuty `structID` do `nextFormat`są wymienione w kolejności, w jakiej są one określone w standardowym nagłówku komunikatu produktu IBM WebSphere MQ .

b. Atrybuty, `format`, `messageEncoding`i `messageCharset`, opisują sam nagłówek i nie są częścią nagłówka.

c. Należy zdecydować, czy ma być przechowywany identyfikator kodowanego zestawu znaków, czy zestaw znaków rekordu. W języku Java używane są zestawy znaków, a komunikaty produktu IBM WebSphere MQ używają identyfikatorów kodowanego zestawu znaków. Przykładowy kod używa zestawów znaków.

d. `int` jest serializowany do `MQLONG` przez IBM WebSphere MQ. `MQLONG` to 4 bajty.

2. Utwórz procedury pobierające i ustawiające dla atrybutów prywatnych.

a) Utwórz lub wygeneruj procedury pobierające:

```
    public String getHeaderFormat() { return headerFormat; }
    public int getHeaderEncoding() { return headerEncoding;    }
    public String getMessageCharset() { return headerCharset; }
    public int getMessageEncoding() { return headerEncoding; }
    public String getStructID() { return structID; }
    public int getStructLength() { return structLength; }
    public int getVersion() { return version; }
```

b) Utwórz lub wygeneruj procedury ustawiające:

```
    public void setHeaderCharset(String charset) {
        this.headerCharset = charset; }
    public void setHeaderEncoding(int encoding) {
        this.headerEncoding = encoding; }
    public void setHeaderFormat(String headerFormat) {
        this.headerFormat = headerFormat; }
    public void setStructID(String structID) {
        this.structID = structID; }
    public void setStructLength(int structLength) {
        this.structLength = structLength; }
    public void setVersion(int version) {
        this.version = version; }
}
```

3. Utwórz konstruktor, aby utworzyć instancję produktu `RECORD` na podstawie `JMSBytesMessage`.

```
    public RECORD(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super();
        setHeaderCharset(message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET));
        setHeaderEncoding(message.getIntProperty(WMQConstants.JMS_IBM_ENCODING));
        byte[] structID = new byte[RECORD_STRUCT_ID_LENGTH];
        message.readBytes(structID, RECORD_STRUCT_ID_LENGTH);
        setStructID(new String(structID, getMessageCharset()));
        setVersion(message.readInt());
        setStructLength(message.readInt());
    }
```

Uwaga:

a. Wartości `messageCharset` i `messageEncoding`są przechwytywane z właściwości komunikatu, ponieważ zastępują one wartości ustawione dla miejsca docelowego. Produkt `format` nie został zaktualizowany. W przykładzie nie jest sprawdzane żadne błędy. Jeśli wywołano konstruktor `Record(BytesMessage)` , przyjmuje się, że `JMSBytesMessage` jest komunikatem typu `RECORD` . Linia "`setStructID(new String(structID, getMessageCharset()))`" ustawia łapacz oczu.

b. Wiersze kodu, które wypełniają pola z deserializacją metody w komunikacie, w kolejności aktualizowania wartości domyślnych ustawionych w instancji RECORD.

4. Utwórz metodę put, aby zapisać pola nagłówka w JMSBytesMessage.

```
protected BytesMessage put(MyProducer myProducer) throws IOException,
        JMSException, UnsupportedEncodingException {
    setHeaderEncoding(myProducer.getEncoding());
    setHeaderCharset(myProducer.getCharset());
    myProducer.setMQClient(true);
    BytesMessage bytes = myProducer.session.createBytesMessage();
    bytes.setStringProperty(WMQConstants.JMS_IBM_FORMAT, getHeaderFormat());
    bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getHeaderEncoding());
    bytes.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET,
                         myProducer.getCCSID());
    bytes.writeBytes(String.format("%1$-" + RECORD_STRUCT_ID_LENGTH + "."
                    + RECORD_STRUCT_ID_LENGTH + "s", getStructID())
                    .getBytes(getMessageCharset()), 0, RECORD_STRUCT_ID_LENGTH);
    bytes.writeInt(getVersion());
    bytes.writeInt(getStructLength());
    return bytes;
}
```

Uwaga:

a. Produkt MyProducer hermetykuje elementy JMS Connection, Destination, Sessioni MessageProducer w jednej klasie. MyConsumer, używany później, hermetykuje JMS Connection, Destination, Sessioni MessageConsumer w jednej klasie.

b. W przypadku partycji JMSBytesMessage, jeśli kodowanie jest inne niż Native, kodowanie musi być ustawione w komunikacie. Kodowanie docelowe jest kopiowane do atrybutu kodowania komunikatów JMS_IBM_CHARACTER_SETi zapisywane jako atrybut klasy RECORD .

i) Program "setMessageEncoding(myProducer.getEncoding());" wywołuje "(((MQDestination) destination).getIntProperty(WMQConstants.WMQ_ENCODING));" , aby uzyskać kodowanie miejsca docelowego.

ii) "Bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getMessageEncoding());" ustawia kodowanie komunikatów.

c. Zestaw znaków używany do transformowania tekstu w bajty jest uzyskiwany z miejsca docelowego i zapisywany jako atrybut klasy RECORD . Nie jest on ustawiony w komunikacie, ponieważ nie jest używany przez klasy produktu IBM WebSphere MQ dla usługi JMS podczas zapisywania JMSBytesMessage.

Wywołania "messageCharset = myProducer.getCharset();"

```
public String getCharset() throws UnsupportedEncodingException,
        JMSException {
    return CCSID.getCodepage(getCCSID());
}
```

Pobiera on zestaw znaków Java z identyfikatora kodowanego zestawu znaków.

"CCSID.getCodepage(ccsid)" znajduje się w pakiecie com.ibm.mq.headers. ccsid jest uzyskiwane z innej metody w produkcie MyProducer, która wysyła zapytania do miejsca docelowego:

```
public int getCCSID() throws JMSException {
    return (((MQDestination) destination)
            .getIntProperty(WMQConstants.WMQ_CCSID));
}
```

d. "myProducer.setMQClient(true);" przesłania ustawienie miejsca docelowego dla typu klienta, wymuszając go na kliencie MQI produktu IBM WebSphere MQ . Może być konieczne pominięcie tej linii kodu, ponieważ powoduje to wystąpienie błędu konfiguracji administracyjnej.

"myProducer.setMQClient(true);" wywołuje:

```
((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ); }
if (!getMQDest()) setMQBody();
```

Kod ten ma efekt uboczny ustawienia stylu treści IBM WebSphere MQ na nieokreślony, jeśli musi przesłonić ustawienie JMS.

Uwaga:

Klasy IBM WebSphere MQ classes for JMS zapisują format, kodowanie i identyfikator zestawu znaków komunikatu w deskryptorze komunikatu, MQMDlub w nagłówku JMS MQRFH2. Zależy to od tego, czy komunikat ma treść w stylu IBM WebSphere MQ . Nie należy ustawiać pól MQMD ręcznie.

Istnieje metoda ręcznego ustawiania właściwości deskryptora komunikatu. Używa on właściwości produktu JMS_IBM_MQMD_* . Aby ustawić właściwości produktu JMS_IBM_MQMD_* , należy ustawić właściwość docelową WMQ_MQMD_WRITE_ENABLED :

```
((MQDestination)destination).setMQMDWriteEnabled(true);
```

Aby odczytać właściwości, należy ustawić właściwość miejsca docelowego WMQ_MQMD_READ_ENABLED.

JMS_IBM_MQMD_* należy używać tylko wtedy, gdy użytkownik zapełni kontrolę nad całym ładunkiem komunikatu. W przeciwieństwie do właściwości produktu JMS_IBM_* właściwości produktu JMS_IBM_MQMD_* nie sterują sposobem, w jaki klasy IBM WebSphere MQ classes for JMS konstruują komunikat JMS. Możliwe jest utworzenie właściwości deskryptora komunikatu, które powodują konflikt z właściwościami komunikatu JMS.

e. Wiersze kodu, które ukończy metodę, serializują atrybuty w klasie jako pola w komunikacie.

   Atrybuty łańcuchowe są dopełniane spacjami. Łańcuchy są przekształcane w bajty przy użyciu zestawu znaków zdefiniowanego dla rekordu, a następnie obcinane do długości pól komunikatu.

5. Zakończ klasę, dodając importy.

```
package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;
```

6. Utwórz klasę, aby rozszerzyć klasę RECORD w celu uwzględnienia dodatkowych pól. Uwzględnij konstruktor domyślny.

```
public class MyRecord extends RECORD {
    private static final long serialVersionUID = -370551723162299429L;
    private final static int FLAGS = 1;
    private final static String STRUCT_ID = "MYRD";
    private final static int DATA_LENGTH = 32;
    private final static String FORMAT = "MYRECORD";
    private int flags = FLAGS;
    private String recordData = "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345";

    public MyRecord() {
        super();
        super.setStructID(STRUCT_ID);
        super.setHeaderFormat(FORMAT);
        super.setStructLength(super.getStructLength() + MQLONG_LENGTH
                + DATA_LENGTH);
    }
```

Uwaga:

a. Podklasa RECORD , MyRecord, dostosowuje chwytnik do oczu, format i długość nagłówka.

7. Utwórz lub wygeneruj procedury pobierające i ustawiające.

a) Utwórz metody pobierające:

```
    public int getFlags() { return flags; }
    public String getRecordData() { return recordData; }Â .
```

b) Utwórz procedury ustawiające:

```
    public void setFlags(int flags) {
        this.flags = flags; }
    public void setRecordData(String recordData) {
        this.recordData = recordData; }
}
```

8. Utwórz konstruktor, aby utworzyć instancję produktu MyRecord na podstawie JMSBytesMessage.

```
    public MyRecord(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super(message);
        setFlags(message.readInt());
        byte[] recordData = new byte[DATA_LENGTH];
        message.readBytes(recordData, DATA_LENGTH);
        setRecordData(new String(recordData, super.getMessageCharset()));
    }
```

Uwaga:

a. Pola, które składają się na standardowy szablon komunikatu, są odczytyane najpierw przez klasę RECORD .

b. Tekst recordData jest przekształcany w produkt String przy użyciu właściwości zestawu znaków komunikatu.

9. Utwórz metodę statyczną, aby pobrać komunikat od konsumenta i utworzyć nową instancję produktu MyRecord .

```
    public static MyRecord get(MyConsumer myConsumer) throws JMSException,
            MQDataException, IOException {
        BytesMessage message = (BytesMessage) myConsumer.receive();
        return new MyRecord(message);
    }
```

Uwaga:

a. W przykładzie, w przypadku zapierania, konstruktor MyRecord(BytesMessage) jest wywoływany z metody get static. Zwykle użytkownik może oddzielić odbieranie komunikatu od utworzenia nowej instancji produktu MyRecord .

10. Utwórz metodę put, aby dodać pola klienta do JMSBytesMessage zawierającego nagłówek komunikatu.

```
    public BytesMessage put(MyProducer myProducer) throws JMSException,
            IOException {
        BytesMessage bytes = super.put(myProducer);
        bytes.writeInt(getFlags());
        bytes.writeBytes(String.format("%1$-" + DATA_LENGTH + "."
                        + DATA_LENGTH + "s",getRecordData())
                        .getBytes(super.getMessageCharset()), 0, DATA_LENGTH);
        myProducer.send(bytes);
        return bytes;
    }
```

Uwaga:

a. Metoda wywołuje w kodzie serializację atrybutów w klasie MyRecord jako pola w komunikacie.

- Atrybut recordData String jest dopełniany spacjami, przekształcony w bajty przy użyciu zestawu znaków zdefiniowanego dla rekordu i obcięty do długości pól RecordData .

11. Zakończ klasę, dodając instrukcje include.

```
package com.ibm.mq.id;
import java.io.IOException;
```

```
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.headers.MQDataException;
```

Wyniki

Wyniki:

- Wyniki działania klasy TryMyRecord :
  - Wysyłanie komunikatu z kodowanego zestawu znaków 37 i użycie wyjścia konwersji menedżera kolejek:

```
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
In  flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 273 CCSID UTF-8
```

  - Wysyłanie komunikatu z kodowanego zestawu znaków 37, a *nie* przy użyciu wyjścia konwersji menedżera kolejek:

```
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
In  flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID IBM037
```

- Wynikiem modyfikacji klasy TryMyRecord nie jest odebranie komunikatu, a następnie odbieranie go za pomocą zmodyfikowanej próbki produktu amqsget0.c . Zmodyfikowana próbka akceptuje sformatowany rekord; patrz Rysunek 156 na stronie 876 w “Wymiana sformatowanego rekordu przy użyciu aplikacji innej niż JMS” na stronie 873.
  - Wysyłanie komunikatu z kodowanego zestawu znaków 37 i użycie wyjścia konwersji menedżera kolejek:

```
Sample AMQSGET0 start
ccsid <850>, flags <1>, message <ABCDEFGHIJKLMNOPQRSTUVWXYZ012345>
no more messages
Sample AMQSGET0 end
```

  - Wysyłanie komunikatu z kodowanego zestawu znaków 37, a *nie* przy użyciu wyjścia konwersji menedżera kolejek:

```
Sample AMQSGET0 start
MQGET ended with reason code 2110
ccsid <37>, flags <1>, message <--+-+ãÃ++ÐÊËÈiÐÎÐ+ÔÒõõµþÞÚ-±=¾¶§>
no more messages
Sample AMQSGET0 end
```

W celu wypróbowania przykładu i eksperymentu z różnymi stronami kodowymi i wyjściem konwersji danych. Utwórz klasy Java, skonfiguruj produkt IBM WebSphere MQi uruchom program główny, TryMyRecord; patrz Rysunek 157 na stronie 884.

1. Aby uruchomić przykład, należy skonfigurować produkt IBM WebSphere MQ i JMS. Instrukcje są przeznaczone do uruchamiania przykładu w systemie Windows.

   1. Tworzenie menedżera kolejek

   ```
   crtmqm -sa -u SYSTEM.DEAD.LETTER.QUEUE QM1
   strmqm QM1
   ```

   2. Tworzenie kolejki

   ```
   echo DEFINE QL('Q1') REPLACE | runmqsc QM1
   ```

   3. Tworzenie katalogu JNDI

   ```
   cd c:\
   md JNDI-Directory
   ```

4. **Przełącz na katalog bin JMS**
   Program administracyjny JMS musi być uruchomiony z tego miejsca. Ścieżka to *MQ_INSTALLATION_PATH*\java\bin.
5. **Tworzenie następujących definicji JMS w pliku o nazwie `JMSQM1Q1.txt`**

```
DEF CF(QM1) PROVIDERVERSION(7) QMANAGER(QM1)
DEF Q(Q1) CCSID(37) ENCODING(RRR) MSGBODY(MQ) QMANAGER(QM1) QUEUE(Q1) TARGCLIENT(MQ)
VERSION(7)
END
```

6. **Uruchom program JMSAdmin w celu utworzenia zasobów JMS**

```
JMSAdmin < JMSQM1Q1.txt
```

2. Użytkownik może tworzyć, zmieniać i przeglądać definicje utworzone za pomocą programu IBM WebSphere MQ Explorer.
3. Uruchom program `TryMyRecord`.

Klasy używane do uruchamiania przykładu

Klasy wymienione w rysunkach Rysunek 157 na stronie 884 do Rysunek 162 na stronie 888 są również dostępne w pliku skompresowanym; należy pobrać plik jm25529_.zip lub jm25529_.tar.gz.

```
package com.ibm.mq.id;
public class TryMyRecord {
    public static void main(String[] args) throws Exception {
        MyProducer producer = new MyProducer();
        MyRecord outrec = new MyRecord();
        System.out.println("Out flags " + outrec.getFlags() + " text "
                + outrec.getRecordData() + " Encoding "
                + producer.getEncoding() + " CCSID " + producer.getCCSID()
                + " MQ " + producer.getMQDest());
        outrec.put(producer);
        System.out.println("Out flags " + outrec.getFlags() + " text "
                + outrec.getRecordData() + " Encoding "
                + producer.getEncoding() + " CCSID " + producer.getCCSID()
                + " MQ " + producer.getMQDest());
        MyRecord inrec = MyRecord.get(new MyConsumer());
        System.out.println("In  flags " + inrec.getFlags() + " text "
                + inrec.getRecordData() + " Encoding "
                + inrec.getMessageEncoding() + " CCSID "
                + inrec.getMessageCharset());
    }
}
```

Rysunek 157. `TryMyRecord`

```
package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;

public abstract class RECORD implements Serializable {
    private static final long serialVersionUID = -1616617232750561712L;
    protected final static int UTF8 = 1208;
    protected final static int MQLONG_LENGTH = 4;
    protected final static int RECORD_STRUCT_ID_LENGTH = 4;
    protected final static int RECORD_VERSION_1 = 1;
    protected final String RECORD_STRUCT_ID = "BLNK";
    protected final String RECORD_TYPE = "BLANK   ";
    private String structID = RECORD_STRUCT_ID;
    private int version = RECORD_VERSION_1;
    private int structLength = RECORD_STRUCT_ID_LENGTH + MQLONG_LENGTH * 2;
    private int headerEncoding = WMQConstants.WMQ_ENCODING_NATIVE;
    private String headerCharset = "UTF-8";
    private String headerFormat = RECORD_TYPE;

    public RECORD() {
        super();
    }

    public RECORD(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super();
        setHeaderCharset(message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET));
        setHeaderEncoding(message.getIntProperty(WMQConstants.JMS_IBM_ENCODING));
        byte[] structID = new byte[RECORD_STRUCT_ID_LENGTH];
        message.readBytes(structID, RECORD_STRUCT_ID_LENGTH);
        setStructID(new String(structID, getMessageCharset()));
        setVersion(message.readInt());
        setStructLength(message.readInt());
    }

    public String getHeaderFormat() { return headerFormat; }
    public int getHeaderEncoding() { return headerEncoding;    }
    public String getMessageCharset() { return headerCharset; }
    public int getMessageEncoding() { return headerEncoding; }
    public String getStructID() { return structID; }
    public int getStructLength() { return structLength; }
    public int getVersion() { return version; }


    protected BytesMessage put(MyProducer myProducer) throws IOException,
            JMSException, UnsupportedEncodingException {
        setHeaderEncoding(myProducer.getEncoding());
        setHeaderCharset(myProducer.getCharset());
        myProducer.setMQClient(true);
        BytesMessage bytes = myProducer.session.createBytesMessage();
        bytes.setStringProperty(WMQConstants.JMS_IBM_FORMAT, getHeaderFormat());
        bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getHeaderEncoding());
        bytes.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET,
                             myProducer.getCCSID());
        bytes.writeBytes(String.format("%1$-" + RECORD_STRUCT_ID_LENGTH + "."
                         + RECORD_STRUCT_ID_LENGTH + "s", getStructID())
                         .getBytes(getMessageCharset()), 0, RECORD_STRUCT_ID_LENGTH);
        bytes.writeInt(getVersion());
        bytes.writeInt(getStructLength());
        return bytes;
    }

    public void setHeaderCharset(String charset) {
        this.headerCharset = charset; }
    public void setHeaderEncoding(int encoding) {
        this.headerEncoding = encoding; }
    public void setHeaderFormat(String headerFormat) {
        this.headerFormat = headerFormat; }
    public void setStructID(String structID) {
        this.structID = structID; }
    public void setStructLength(int structLength) {
        this.structLength = structLength; }
    public void setVersion(int version) {
        this.version = version; }
}
```

Rysunek 158. RECORD

```
package com.ibm.mq.id;
import java.io.IOException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.headers.MQDataException;

public class MyRecord extends RECORD {
    private static final long serialVersionUID = -370551723162299429L;
    private final static int FLAGS = 1;
    private final static String STRUCT_ID = "MYRD";
    private final static int DATA_LENGTH = 32;
    private final static String FORMAT = "MYRECORD";
    private int flags = FLAGS;
    private String recordData = "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345";

    public MyRecord() {
        super();
        super.setStructID(STRUCT_ID);
        super.setHeaderFormat(FORMAT);
        super.setStructLength(super.getStructLength() + MQLONG_LENGTH
                + DATA_LENGTH);
    }

    public MyRecord(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super(message);
        setFlags(message.readInt());
        byte[] recordData = new byte[DATA_LENGTH];
        message.readBytes(recordData, DATA_LENGTH);
        setRecordData(new String(recordData, super.getMessageCharset()));
    }

    public static MyRecord get(MyConsumer myConsumer) throws JMSException,
            MQDataException, IOException {
        BytesMessage message = (BytesMessage) myConsumer.receive();
        return new MyRecord(message);
    }

    public int getFlags() { return flags; }
    public String getRecordData() { return recordData; }Â .


    public BytesMessage put(MyProducer myProducer) throws JMSException,
            IOException {
        BytesMessage bytes = super.put(myProducer);
        bytes.writeInt(getFlags());
        bytes.writeBytes(String.format("%1$-" + DATA_LENGTH + "."
                        + DATA_LENGTH + "s",getRecordData())
                        .getBytes(super.getMessageCharset()), 0, DATA_LENGTH);
        myProducer.send(bytes);
        return bytes;
    }

    public void setFlags(int flags) {
        this.flags = flags; }
    public void setRecordData(String recordData) {
        this.recordData = recordData; }
}
```

Rysunek 159. MyRecord

```
package com.ibm.mq.id;
import java.io.UnsupportedEncodingException;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.Session;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import com.ibm.mq.headers.CCSID;
import com.ibm.mq.jms.MQDestination;
import com.ibm.msg.client.wmq.WMQConstants;
public abstract class EndPoint {
    public Context ctx;
    public ConnectionFactory cf;
    public Connection connection;
    public Destination destination;
    public Session session;
    protected EndPoint() throws NamingException, JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup("QM1");
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup("Q1");
        ((MQDestination)destination).setReceiveConversion
                                    (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); }
    protected EndPoint(String cFactory, String dest) throws NamingException,
            JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup(cFactory);
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup(dest);
        ((MQDestination)destination).setReceiveConversion
                                    (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);  }
    public int getCCSID() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_CCSID));  }
    public String getCharset() throws UnsupportedEncodingException,
            JMSException {
        return CCSID.getCodepage(getCCSID());  }
    public int getEncoding() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_ENCODING));  }
    public boolean getMQDest() throws JMSException {
        if ((((MQDestination) destination).getMessageBodyStyle()
               == WMQConstants.WMQ_MESSAGE_BODY_MQ)
            || ((((MQDestination) destination).getMessageBodyStyle()
                  == WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED)
               && (((MQDestination) destination).getTargetClient()
                   == WMQConstants.WMQ_TARGET_DEST_MQ)))
            return true;
        else
            return false; }
    public void setCCSID(int ccsid) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_CCSID,
                ccsid); }
    public void setEncoding(int encoding) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_ENCODING,
                encoding); }
    public void setMQBody() throws JMSException {
        ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);  }
    public void setMQBody(boolean mqbody) throws JMSException {
        if (mqbody)  ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
        else         ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_JMS);  }
    public void setMQClient(boolean mqclient) throws JMSException {
       if (mqclient){
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
            if (!getMQDest()) setMQBody();
       }
       else
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_JMS); }
    }
```

Rysunek 160. EndPoint

```
package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageProducer;
import javax.naming.NamingException;
public class MyProducer extends EndPoint {
    public MessageProducer producer;
    public MyProducer() throws NamingException, JMSException {
        super();
        producer = session.createProducer(destination); }
    public MyProducer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        producer = session.createProducer(destination); }
    public void send(Message message) throws JMSException {
        producer.send(message); }
}
```

Rysunek 161. MyProducer

```
package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.naming.NamingException;
public class MyConsumer extends EndPoint {
    public MessageConsumer consumer;
    public MyConsumer() throws NamingException, JMSException {
        super();
        consumer = session.createConsumer(destination);
        connection.start();  }
    public MyConsumer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        consumer = session.createConsumer(destination);
        connection.start();  }
    public Message receive() throws JMSException {
        return consumer.receive();  }
}
```

Rysunek 162. MyConsumer

Tworzenie i konfigurowanie fabryk połączeń i miejsc docelowych w klasach produktu WebSphere MQ dla aplikacji JMS

Klasy WebSphere MQ dla aplikacji JMS mogą tworzyć fabryki połączeń i miejsca docelowe, pobierając je jako administrowane obiekty z przestrzeni nazw JNDI (Java Naming and Directory Interface), korzystając z rozszerzeń IBM JMS lub przy użyciu rozszerzeń JMS produktu WebSphere MQ . Aplikacja może również używać rozszerzeń IBM JMS lub rozszerzeń JMS produktu WebSphere MQ do ustawiania właściwości fabryk połączeń i miejsc docelowych.

Fabryki połączeń i miejsca docelowe są punktami startów w przepływie logiki aplikacji JMS. Aplikacja korzysta z obiektu ConnectionFactory w celu utworzenia połączenia z serwerem przesyłania komunikatów i używa obiektu Queue lub Topic jako elementu docelowego do wysyłania komunikatów do lub do źródła, z którego mają być odbierane komunikaty. W związku z tym aplikacja musi utworzyć co najmniej jedną fabrykę połączeń i co najmniej jedno miejsce docelowe. Po utworzeniu fabryki połączeń lub miejsca docelowego, aplikacja może wymagać skonfigurowania obiektu poprzez ustawienie jednej lub większej liczby jej właściwości.

Podsumowując, aplikacja może tworzyć i konfigurować fabryki połączeń i miejsca docelowe w jeden z następujących sposobów:

Używanie interfejsu JNDI do pobierania administrowanych obiektów
Administrator może użyć narzędzia administracyjnego WebSphere MQ JMS lub programu WebSphere MQ Explorer w celu utworzenia i skonfigurowania fabryk połączeń i miejsc docelowych jako obiektów administrowanych w przestrzeni nazw JNDI. Aplikacja może następnie pobrać administrowane obiekty z przestrzeni nazw JNDI. Po pobraniu administrowanego obiektu aplikacja może, jeśli jest to wymagane, ustawić lub zmienić jedną lub kilka jej właściwości, korzystając z rozszerzeń IBM JMS lub rozszerzeń JMS produktu WebSphere MQ .

Korzystanie z rozszerzeń IBM JMS
Aplikacja może korzystać z rozszerzeń IBM JMS w celu dynamicznego tworzenia fabryk połączeń i miejsc docelowych w czasie wykonywania. Aplikacja najpierw tworzy obiekt fabryki JmsFactory, a następnie używa metod tego obiektu do tworzenia fabryk połączeń i miejsc docelowych. Po utworzeniu fabryki połączeń lub miejsca docelowego aplikacja może użyć metod dziedziczonych z interfejsu kontekstu JmsPropertyw celu ustawienia jego właściwości. Alternatywnie aplikacja może użyć identyfikatora URI (Uniform Resource Identifier) w celu określenia jednej lub większej liczby właściwości miejsca docelowego podczas tworzenia miejsca docelowego.

Korzystanie z rozszerzeń JMS produktu WebSphere MQ
Aplikacja może również używać rozszerzeń JMS produktu WebSphere MQ do dynamicznego tworzenia fabryk połączeń i miejsc docelowych w czasie wykonywania. Aplikacja korzysta z dostarczonych konstruktorów w celu utworzenia fabryk połączeń i miejsc docelowych. Po utworzeniu fabryki połączeń lub miejsca docelowego aplikacja może użyć metod obiektu w celu ustawienia jego właściwości. Alternatywnie aplikacja może użyć identyfikatora URI, aby określić jedną lub więcej właściwości miejsca docelowego podczas tworzenia miejsca docelowego.

Używanie interfejsu JNDI do pobierania administrowanych obiektów w aplikacji JMS

Aby pobrać administrowane obiekty z przestrzeni nazw JNDI (Java Naming and Directory Interface), aplikacja JMS musi utworzyć kontekst początkowy, a następnie użyć metody lookup () w celu pobrania obiektów.

Zanim aplikacja będzie mogła pobierać administrowane obiekty z przestrzeni nazw JNDI, administrator musi najpierw utworzyć administrowane obiekty. Administrator może używać narzędzia administracyjnego WebSphere MQ JMS lub programu WebSphere MQ Explorer do tworzenia i utrzymywania administrowanych obiektów w przestrzeni nazw JNDI. Więcej informacji na temat korzystania z narzędzia administracyjnego WebSphere MQ JMS zawiera sekcja “Korzystanie z narzędzia administracyjnego JMS produktu WebSphere MQ” na stronie 960. Informacje na temat korzystania z programu WebSphere MQ Explorer można znaleźć w pomocy dostarczonej z programem WebSphere MQ Explorer. Jednak serwer aplikacji zwykle udostępnia własne repozytorium dla administrowanych obiektów oraz własnych narzędzi do tworzenia i obsługi obiektów.

Aby pobrać obiekty administrowane z przestrzeni nazw JNDI, aplikacja musi najpierw utworzyć kontekst początkowy, tak jak pokazano to w poniższym przykładzie:

```
import javax.jms.*;
import javax.naming.*;
import javax.naming.directory.*;
.
.
.
String url = "ldap://server.company.com/o=company_us,c=us";
String icf = "com.sun.jndi.ldap.LdapCtxFactory";
.
java.util.Hashtable environment = new java.util.Hashtable();
environment.put(Context.PROVIDER_URL, url);
environment.put(Context.INITIAL_CONTEXT_FACTORY, icf);
Context ctx = new InitialDirContext(environment);
```

W tym kodzie zmienne łańcuchowe `url` i `icf` mają następujące znaczenie:

url
Adres URL (Uniform Resource Locator) usługi katalogowej. Adres URL może mieć jeden z następujących formatów:

- `ldap://`*`hostname`*`/`*`contextName`*dla usługi katalogowej opartej na serwerze LDAP
- `file:/`*`directoryPath`*dla usługi katalogowej opartej na lokalnym systemie plików

icf
Nazwa klasy fabryki kontekstu początkowego, która może mieć jedną z następujących wartości:

- `com.sun.jndi.ldap.LdapCtxFactory`dla usługi katalogowej opartej na serwerze LDAP
- `com.sun.jndi.fscontext.RefFSContextFactory`dla usługi katalogowej opartej na lokalnym systemie plików

Należy zauważyć, że niektóre kombinacje pakietu JNDI i dostawcy usług LDAP (Lightweight Directory Access Protocol) mogą powodować wystąpienie błędu LDAP 84. Aby rozwiązać ten problem, przed wywołaniem funkcji InitialDirContext () należy wstawić następujący wiersz kodu:

```
environment.put(Context.REFERRAL, "throw");
```

Po uzyskaniu kontekstu początkowego aplikacja może pobierać administrowane obiekty z przestrzeni nazw JNDI przy użyciu metody lookup (), jak pokazano w poniższym przykładzie:

```
ConnectionFactory factory;
Queue queue;
Topic topic;
.
.
.
factory = (ConnectionFactory)ctx.lookup("cn=myCF");
queue = (Queue)ctx.lookup("cn=myQ");
topic = (Topic)ctx.lookup("cn=myT");
```

Ten kod pobiera następujące obiekty z przestrzeni nazw opartej na protokole LDAP:

- Obiekt ConnectionFactory powiązany z nazwą myCF
- Obiekt kolejki powiązany z nazwą myQ
- Obiekt tematu powiązany z nazwą myT

Korzystanie z rozszerzeń IBM JMS

Klasy WebSphere MQ classes for JMS zawierają zestaw rozszerzeń interfejsu API JMS o nazwie IBM JMS extensions. Aplikacja może używać tych rozszerzeń do dynamicznego tworzenia fabryk połączeń i miejsc docelowych w czasie wykonywania, a także do ustawiania właściwości klas WebSphere MQ dla obiektów JMS. Rozszerzenia mogą być używane z dowolnym dostawcą przesyłania komunikatów.

Rozszerzenia JMS produktu IBM są zestawem interfejsów i klas w następujących pakietach:

- com.ibm.msg.client.jms
- com.ibm.msg.client.services

Pakiety te można znaleźć w programie `com.ibm.mqjms.jar` , który znajduje się w katalogu `<MQ_Install_Dir>/java/lib`.

Rozszerzenia te udostępniają następujące funkcje:

- Mechanizm fabryczny do dynamicznego tworzenia fabryk połączeń i miejsc docelowych w czasie wykonywania, zamiast pobierania ich jako administrowanych obiektów z przestrzeni nazw JNDI (Java Naming and Directory Interface).
- Zestaw metod ustawiania właściwości klas produktu WebSphere MQ dla obiektów JMS.
- Zestaw klas wyjątków z metodami uzyskiwania szczegółowych informacji na temat problemu
- Zestaw metod sterowania śledzeniem
- Zestaw metod uzyskiwania informacji o wersji dotyczących klas produktu WebSphere MQ dla usługi JMS

Jeśli chodzi o dynamiczne tworzenie fabryk połączeń i miejsc docelowych w czasie wykonywania, a także ustawianie i pobieranie ich właściwości, rozszerzenia IBM JMS udostępniają alternatywny zestaw interfejsów do rozszerzeń JMS produktu WebSphere MQ . Jednak, podczas gdy rozszerzenia WebSphere MQ JMS są specyficzne dla dostawcy przesyłania komunikatów produktu WebSphere MQ , rozszerzenia JMS produktu IBM nie są specyficzne dla produktu WebSphere MQ i mogą być używane z dowolnym dostawcą przesyłania komunikatów w architekturze warstwowej opisanej w sekcji “Architektura warstwowa” na stronie 820.

Interfejs com.ibm.msg.client.wmq.WMQConstants zawiera definicje stałych, które mogą być używane przez aplikację podczas ustawiania właściwości klas produktu WebSphere MQ dla obiektów JMS przy użyciu rozszerzeń IBM JMS. Interfejs zawiera stałe dla dostawców przesyłania komunikatów produktu WebSphere MQ i stałych JMS, które są niezależne od dowolnego dostawcy przesyłania komunikatów.

W poniższych przykładach kodu przyjęto założenie, że zostały uruchomione następujące instrukcje importu:

```
import com.ibm.msg.client.jms.*;
import com.ibm.msg.client.services.*;
import com.ibm.msg.client.wmq.WMQConstants;
```

Tworzenie fabryk połączeń i miejsc docelowych

Zanim aplikacja będzie mogła utworzyć fabryki połączeń i miejsca docelowe przy użyciu rozszerzeń JMS IBM , musi najpierw utworzyć obiekt fabryki JmsFactory. Aby utworzyć obiekt fabryki JmsFactory, aplikacja wywołuje metodę getInstance() klasy fabryki JmsFactory, tak jak przedstawiono to w poniższym przykładzie:

```
JmsFactoryFactory ff = JmsFactoryFactory.getInstance(JmsConstants.WMQ_PROVIDER);
```

Parametr w wywołaniu metody getInstance() jest stałą, która identyfikuje dostawcę przesyłania komunikatów produktu WebSphere MQ jako wybranego dostawcę przesyłania komunikatów. Aplikacja może następnie użyć obiektu fabryki JmsFactorydo utworzenia fabryk połączeń i miejsc docelowych.

Aby utworzyć fabrykę połączeń, aplikacja wywołuje metodę createConnectionFactory () obiektu fabryki JmsFactory, tak jak przedstawiono to w poniższym przykładzie:

```
JmsConnectionFactory factory = ff.createConnectionFactory();
```

Ta instrukcja tworzy obiekt fabryki JmsConnectionz wartościami domyślnymi dla wszystkich jego właściwości, co oznacza, że aplikacja nawiązuje połączenie z domyślnym menedżerem kolejek w trybie powiązań. Jeśli aplikacja ma łączyć się w trybie klienckim lub łączyć się z menedżerem kolejek innym niż domyślny menedżer kolejek, przed utworzeniem połączenia aplikacja musi ustawić odpowiednie właściwości obiektu fabryki JmsConnection. Informacje na temat sposobu wykonania tej czynności zawiera sekcja “Ustawianie właściwości klas produktu WebSphere MQ dla obiektów JMS” na stronie 892.

Klasa fabryki JmsFactoryzawiera również metody tworzenia fabryk połączeń dla następujących typów:

- JmsQueueConnectionFactory
- JmsTopicConnectionFactory
- Fabryka JmsXAConnection
- JmsXAQueueConnectionFactory
- JmsXATopicConnectionFactory

Aby utworzyć obiekt kolejki, aplikacja wywołuje metodę createQueue() obiektu fabryki JmsFactory, tak jak przedstawiono to w poniższym przykładzie:

```
JmsQueue q1 = ff.createQueue("Q1");
```

Ta instrukcja tworzy obiekt JmsQueue z wartościami domyślnymi dla wszystkich jego właściwości. Obiekt reprezentuje kolejkę WebSphere MQ o nazwie Q1 , która należy do lokalnego menedżera kolejek. Ta kolejka może być kolejką lokalną, kolejką aliasową lub definicją kolejki zdalnej.

Metoda createQueue() może także akceptować identyfikator URI (Uniform Resource Identifier) kolejki jako parametr. Identyfikator URI kolejki to łańcuch, który określa nazwę kolejki produktu WebSphere MQ i opcjonalnie nazwę menedżera kolejek, który jest właścicielem kolejki, oraz co najmniej jedną właściwość obiektu JmsQueue . Następująca instrukcja zawiera przykład identyfikatora URI kolejki:

```
JmsQueue q2 = ff.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

Obiekt JmsQueue utworzony przez tę instrukcję reprezentuje kolejkę WebSphere MQ o nazwie Q2 , która należy do menedżera kolejek QM2, a wszystkie komunikaty wysłane do tego miejsca docelowego są trwałe i mają priorytet 5. Więcej informacji na temat identyfikatorów URI kolejek zawiera sekcja “Jednolite identyfikatory zasobów (URI)” na stronie 904. Aby uzyskać alternatywny sposób ustawiania

właściwości obiektu JmsQueue , należy zapoznać się z sekcji “Ustawianie właściwości klas produktu WebSphere MQ dla obiektów JMS” na stronie 892.

Aby utworzyć obiekt tematu, aplikacja może użyć metody createTopic() obiektu fabryki JmsFactory, tak jak przedstawiono to w poniższym przykładzie:

```
JmsTopic t1 = ff.createTopic("Sport/Football/Results");
```

Ta instrukcja tworzy obiekt JmsTopic z wartościami domyślnymi dla wszystkich jego właściwości. Obiekt reprezentuje temat o nazwie Sport/Football/Results.

Metoda createTopic() może także akceptować identyfikator URI tematu jako parametr. Identyfikator URI tematu to łańcuch, który określa nazwę tematu i opcjonalnie jedną lub więcej właściwości obiektu JmsTopic . Następujące instrukcje zawierają przykładowy identyfikator URI tematu:

```
String s1 = "topic://Sport/Tennis/Results?persistence=1&priority=0";
JmsTopic t2 = ff.createTopic(s1);
```

Obiekt JmsTopic utworzony przez te instrukcje reprezentuje temat o nazwie Sport/Tennis/Results, a wszystkie komunikaty wysłane do tego miejsca docelowego są nietrwałe i mają priorytet równy 0. Więcej informacji na temat identyfikatorów URI tematów zawiera sekcja “Jednolite identyfikatory zasobów (URI)” na stronie 904. Aby uzyskać alternatywny sposób ustawiania właściwości obiektu JmsTopic , patrz sekcja “Ustawianie właściwości klas produktu WebSphere MQ dla obiektów JMS” na stronie 892.

Po utworzeniu fabryki połączeń lub miejsca docelowego przez aplikację ten obiekt może być używany tylko z wybranym dostawcą przesyłania komunikatów.

Ustawianie właściwości klas produktu WebSphere MQ dla obiektów JMS

Aby ustawić właściwości klas WebSphere MQ dla obiektów JMS przy użyciu rozszerzeń JMS IBM , aplikacja korzysta z metod interfejsu com.ibm.msg.client.JmsPropertyContext .

W przypadku każdego typu danych Java interfejs kontekstu JmsPropertyzawiera metodę ustawiania wartości właściwości z tym typem danych oraz metodę pobierania wartości właściwości z tym typem danych. Na przykład aplikacja wywołuje metodę setIntProperty () w celu ustawienia właściwości o wartości liczby całkowitej i wywołuje metodę getIntProperty () w celu pobrania właściwości z wartością całkowitą.

Instancje klas w pakiecie com.ibm.mq.jms dziedziczą również metody interfejsu kontekstu JmsProperty. Aplikacja może zatem użyć tych metod w celu ustawienia właściwości obiektów MQConnectionFactory, MQQueue i MQTopic.

Gdy aplikacja tworzy klasy produktu WebSphere MQ dla obiektu JMS, wszystkie właściwości z wartościami domyślnymi są ustawiane automatycznie. Gdy aplikacja ustawia właściwość, nowa wartość zastępuje poprzednią wartość posiadanej właściwości. Po ustawieniu właściwości nie można jej usunąć, ale jej wartość może zostać zmieniona.

Jeśli aplikacja próbuje ustawić właściwość na wartość, która nie jest poprawną wartością dla właściwości, klasy WebSphere MQ classes for JMS zgłasza wyjątek JMSException. Jeśli aplikacja podejmie próbę pobrania właściwości, która nie została ustawiona, zachowanie jest opisane w specyfikacji JMS. Produkt WebSphere MQ classes for JMS zgłasza wyjątek NumberFormatwyjątku dla podstawowych typów danych i zwraca wartość NULL dla przywoływanych typów danych.

Oprócz wstępnie zdefiniowanych właściwości klas WebSphere MQ dla obiektu JMS aplikacja może ustawiać własne właściwości. Te zdefiniowane właściwości aplikacji są ignorowane przez klasy produktu WebSphere MQ dla usługi JMS.

Więcej informacji na temat właściwości klas produktu WebSphere MQ dla obiektów JMS zawiera sekcja Właściwości obiektów produktu IBM WebSphere MQ classes for JMS.

Poniższy kod jest przykładem sposobu ustawiania właściwości przy użyciu rozszerzeń IBM JMS. Kod ustawia pięć właściwości fabryki połączeń.

```
factory.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE,
                       WMQConstants.WMQ_CM_CLIENT);
factory.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "QM1");
factory.setStringProperty(WMQConstants.WMQ_HOST_NAME, "HOST1");
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
factory.setStringProperty(WMQConstants.WMQ_CHANNEL, "QM1.SVR");
factory.setStringProperty(WMQConstants.WMQ_APPLICATIONNAME, "My Application");
```

Efektem ustawienia tych właściwości jest to, że aplikacja łączy się z menedżerem kolejek QM1 w trybie klienta, używając kanału MQI o nazwie QM1.SVR. Menedżer kolejek jest uruchomiony w systemie o nazwie hosta HOST1, a nasłuchiwanie dla menedżera kolejek nasłuchuje na porcie o numerze 1415. To połączenie i inne połączenia menedżera kolejek powiązane z sesjami, które są pod nią związane, mają powiązaną z nimi nazwę aplikacji "Moja aplikacja".

Uwaga: Menedżery kolejek działające na platformach z/OS nie obsługują ustawiania nazw aplikacji, a to ustawienie jest ignorowane.

Interfejs kontekstu JmsPropertyzawiera również metodę setObjectProperty (), która może być używana przez aplikację do ustawiania właściwości. Drugi parametr metody to obiekt, który hermetyzuje wartość właściwości. Na przykład następujący kod tworzy obiekt typu Integer, który hermetykuje liczbę całkowitą 1415, a następnie wywołuje właściwość setObject() w celu ustawienia właściwości PORT fabryki połączeń na wartość 1415:

```
Integer port = new Integer(1415);
factory.setObjectProperty(WMQConstants.WMQ_PORT, port);
```

Kod ten jest zatem równoznaczny z następującym stwierdzeniem:

```
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
```

Z drugiej strony metoda getObjectProperty () zwraca obiekt, który hermetykuje wartość właściwości.

Niejawne przekształcenie wartości właściwości z jednego typu danych na inny.

Jeśli aplikacja używa metody interfejsu kontekstu JmsPropertydo ustawiania lub pobierania właściwości klas WebSphere MQ dla obiektu JMS, wartość tej właściwości może być niejawnie przekształcona z jednego typu danych na inny.

Na przykład poniższa instrukcja ustawia właściwość PRIORITY obiektu JmsQueue q1: .

```
q1.setStringProperty(WMQConstants.WMQ_PRIORITY, "5");
```

Właściwość PRIORITY ma wartość całkowitą, a więc wywołanie właściwości setString() niejawnie przekształca łańcuch "5" (wartość źródłowa) na liczbę całkowitą 5 (wartość docelowa), która staje się wartością właściwości PRIORITY.

I odwrotnie, następująca instrukcja pobiera właściwość PRIORITY obiektu JmsQueue q1:

```
String s1 = q1.getStringProperty(WMQConstants.WMQ_PRIORITY);
```

Liczba całkowita 5 (wartość źródłowa), która jest wartością właściwości PRIORITY, jest niejawnie konwertowana na łańcuch "5" (wartość docelowa) przy użyciu wywołania właściwości getString().

Konwersje obsługiwane przez klasy produktu WebSphere MQ dla usługi JMS są wyświetlane w produkcie Tabela 122 na stronie 893.

Tabela 122. Obsługiwane konwersje z jednego typu danych do innego

Źródłowy typ danych	**Obsługiwane docelowe typy danych**
boolean (boolowskie)	Łańcuch
B	int, long, short, String

Tabela 122. Obsługiwane konwersje z jednego typu danych do innego (kontynuacja)	
Źródłowy typ danych	**Obsługiwane docelowe typy danych**
char	Łańcuch
double (podwójna)	Łańcuch
liczba zmiennopozycyjna	double, String
int	long, String
long	Łańcuch
short	int, long, String
Łańcuch	boolean, byte, double, float, int, long, short

Ogólne reguły dotyczące obsługiwanych konwersji są następujące:

- Wartości liczbowe mogą być przekształcane z jednego typu danych na inny, pod warunkiem, że w trakcie konwersji nie są tracone żadne dane. Na przykład wartość o typie danych `int` może zostać przekształcona w wartość o typie danych `long`, ale nie może zostać przekształcona w wartość o typie danych `short`.
- Wartość dowolnego typu danych może zostać przekształcona w łańcuch.
- Łańcuch może zostać przekształcony na wartość dowolnego innego typu danych (z wyjątkiem `char`), pod warunkiem, że łańcuch ma poprawny format konwersji. Jeśli aplikacja podejmie próbę przekształcenia łańcucha, który nie ma poprawnego formatu, klasy WebSphere MQ classes for JMS zgłasza wyjątek NumberFormatException.
- Jeśli aplikacja podejmie próbę konwersji, która nie jest obsługiwana, klasy WebSphere MQ classes for JMS zgłasza wyjątek MessageFormat.

Szczegółowe reguły przekształcania wartości z jednego typu danych do drugiego są następujące:

- Podczas przekształcania wartości boolowskiej w łańcuch wartość `true` jest przekształcana w łańcuch "true", a wartość `false` jest przekształcana w łańcuch "false".
- Podczas przekształcania łańcucha w wartość boolowskim łańcuch "true" (bez rozróżniania wielkości liter) jest przekształcany w łańcuch `true`, a łańcuch "false" (bez rozróżniania wielkości liter) jest przekształcany w wartość `false`. Dowolny inny łańcuch jest przekształcany w `false`.
- Podczas przekształcania łańcucha w wartość o typie danych `byte`, `int`, `long`lub `short`łańcuch musi mieć następujący format:

  [*odstępy*] [*znak*]*cyfry*

  Znaczenia składników tego łańcucha są następujące:

  wartości puste
  Opcjonalne wiodące puste znaki.

  znak
  Opcjonalny znak plus (+) lub znak minus (-).

  cyfry
  Ciągła sekwencja cyfr (0-9). Musi istnieć co najmniej jedna cyfra.

  Po sekwencji cyfr łańcuch może zawierać inne znaki, które nie są cyframi, ale konwersja zatrzymuje się, gdy tylko pierwszy z tych znaków zostanie osiągnięty. Przyjmuje się, że łańcuch reprezentuje dziesiętną liczbę całkowitą.

  Jeśli łańcuch nie jest w poprawnym formacie, klasy WebSphere MQ classes for JMS zgłasza wyjątek NumberFormatException.
- Podczas przekształcania łańcucha w wartość typu danych `double` lub `float`łańcuch musi mieć następujący format:

[*odstępy*] [*znak*]*cyfry*[*e_char*[*znak*]*e_cyfry*]

Znaczenia składników tego łańcucha są następujące:

wartości puste
Opcjonalne wiodące puste znaki.

znak
Opcjonalny znak plus (+) lub znak minus (-).

cyfry
Ciągła sekwencja cyfr (0-9). Musi istnieć co najmniej jedna cyfra.

znak
Znak wykładnika, który ma wartość *E* lub *e*.

znak
Opcjonalny znak plus (+) lub znak minus (-) dla wykładnika.

_cyfry
Ciągła sekwencja cyfr (0-9) dla wykładnika. Co najmniej jedna cyfra musi być obecna, jeśli łańcuch zawiera znak wykładnika.

Po sekwencji cyfr lub opcjonalnych znaków reprezentujących wykładnik, łańcuch może zawierać inne znaki, które nie są cyframi, ale konwersja zatrzymuje się, gdy tylko pierwszy z tych znaków zostanie osiągnięty. Przyjmuje się, że łańcuch reprezentuje liczbę dziesiętną zmiennopozycyjną z wykładnikiem, który jest potęgową liczbą 10.

Jeśli łańcuch nie jest w poprawnym formacie, klasy WebSphere MQ classes for JMS zgłasza wyjątek NumberFormatException.

- Podczas przekształcania wartości liczbowej (w tym wartości z typem danych `byte`) na łańcuch, wartość jest przekształcana w łańcuch reprezentujący wartość jako liczbę dziesiętną, a nie łańcuch zawierający znak ASCII dla tej wartości. Na przykład liczba całkowita 65 jest przekształcana w łańcuch "65", a nie na łańcuch "A".

Ustawianie więcej niż jednej właściwości w pojedynczym wywołaniu

Interfejs kontekstu JmsPropertyzawiera również metodę setBatchProperties (), która może być używana przez aplikację do ustawiania więcej niż jednej właściwości w pojedynczym wywołaniu. Parametr metody jest obiektem Map, który hermetyzuje zestaw par nazwa-wartość właściwości.

Na przykład poniższy kod używa metody setBatchProperties () do ustawienia tych samych pięciu właściwości fabryki połączeń, jak pokazano na rysunku “Ustawianie właściwości klas produktu WebSphere MQ dla obiektów JMS” na stronie 892. Kod tworzy instancję klasy HashMap , która implementuje interfejs Map.

```
HashMap batchProperties = new HashMap();
batchProperties.put(WMQConstants.WMQ_CONNECTION_MODE,
                    new Integer(WMQConstants.WMQ_CM_CLIENT));
batchProperties.put(WMQConstants.WMQ_QUEUE_MANAGER, "QM1");
batchProperties.put(WMQConstants.WMQ_WMQ_HOST_NAME, "HOST1");
batchProperties.put(WMQConstants.WMQ_PORT, "1414");
batchProperties.put(WMQConstants.WMQ_CHANNEL, "QM1.SVR");
factory.setBatchProperties(batchProperties);
```

Należy pamiętać, że drugi parametr metody Map.put() musi być obiektem. Dlatego wartość właściwości z podstawowym typem danych musi być hermetyzowana w obiekcie lub reprezentowana przez łańcuch, tak jak pokazano to w przykładzie.

Metoda setBatchProperties () sprawdza poprawność każdej właściwości. Jeśli metoda setBatchProperties () nie może ustawić właściwości, ponieważ na przykład jej wartość jest niepoprawna, żadna z podanych właściwości nie jest ustawiona.

Nazwy i wartości właściwości

Jeśli aplikacja korzysta z metod interfejsu kontekstu JmsPropertydo ustawiania i pobierania właściwości klas WebSphere MQ dla obiektów JMS, aplikacja może określać nazwy i wartości właściwości w jeden z następujących sposobów. Każdy z dołączonych przykładów pokazuje, jak ustawić właściwość PRIORITY obiektu JmsQueue q1 , tak aby komunikat wysłany do kolejki miał priorytet określony w wywołaniu funkcji send ().

Używanie nazw i wartości właściwości, które są zdefiniowane jako stałe w interfejsie com.ibm.msg.client.wmq.WMQConstants .

Poniższa instrukcja jest przykładem sposobu określania nazw i wartości właściwości w następujący sposób:

```
q1.setIntProperty(WMQConstants.WMQ_PRIORITY, WMQConstants.WMQ_PRI_APP);
```

Korzystanie z nazw i wartości właściwości, które mogą być używane w identyfikatorach URI kolejki i tematu (URI)

Poniższa instrukcja jest przykładem sposobu określania nazw i wartości właściwości w następujący sposób:

```
q1.setIntProperty("priority", -2);
```

W ten sposób można określić tylko nazwy i wartości właściwości miejsc docelowych.

Używanie nazw i wartości właściwości, które są rozpoznawane przez narzędzie administracyjne WebSphere MQ JMS

Poniższa instrukcja jest przykładem sposobu określania nazw i wartości właściwości w następujący sposób:

```
q1.setStringProperty("PRIORITY", "APP");
```

Skrócona forma nazwy właściwości jest również akceptowalna, co przedstawiono w poniższej instrukcji:

```
q1.setStringProperty("PRI", "APP");
```

Gdy aplikacja pobiera właściwość, zwracana wartość jest zależna od sposobu, w jaki aplikacja określa nazwę właściwości. Na przykład, jeśli aplikacja określa stałą wartość WMQConstants.WMQ_PRIORITY jako nazwę właściwości, zwracana wartość jest liczbą całkowitą -2:

```
int n1 = getIntProperty(WMQConstants.WMQ_PRIORITY);
```

Ta sama wartość jest zwracana, jeśli w aplikacji określono łańcuch "priority" jako nazwę właściwości:

```
int n2 = getIntProperty("priority");
```

Jeśli jednak aplikacja określi łańcuch "PRIORITY" lub "PRI" jako nazwę właściwości, zwracana wartość jest łańcuchem "APP":

```
String s1 = getStringProperty("PRI");
```

Wewnętrznie, klasy WebSphere MQ classes for JMS przechowuje nazwy i wartości właściwości jako wartości literałów zdefiniowane w interfejsie com.ibm.msg.client.wmq.WMQConstants . Jest to zdefiniowany format kanoniczny dla nazw właściwości i wartości. Co do zasady, jeśli aplikacja ustawia właściwości przy użyciu jednego z dwóch sposobów określania nazw i wartości właściwości, klasy WebSphere MQ classes for JMS muszą przekształcić nazwy i wartości z określonego formatu wejściowego w format kanoniczny. Podobnie, jeśli aplikacja pobiera właściwości przy użyciu jednego z dwóch innych sposobów określania nazw i wartości właściwości, klasy WebSphere MQ classes for JMS muszą przekształcić nazwy z określonego formatu wejściowego w format kanoniczny i przekształcić wartości z formatu kanonicznego w wymagany format wyjściowy. Wykonanie tych konwersji może mieć wpływ na wydajność.

Nazwy właściwości i wartości zwracane przez wyjątki, w plikach śledzenia lub w klasach WebSphere MQ dla dziennika JMS są zawsze w formacie kanonicznym.

Korzystanie z interfejsu mapy

Interfejs kontekstu JmsPropertyrozszerza interfejs java.util.Map . Aplikacja może zatem korzystać z metod interfejsu Map w celu uzyskania dostępu do właściwości klas WebSphere MQ dla obiektu JMS.

Na przykład poniższy kod drukuje nazwy i wartości wszystkich właściwości fabryki połączeń. Kod korzysta tylko z metod interfejsu odwzorowania, aby uzyskać nazwy i wartości właściwości.

```
// Get the names of all the properties
Set propNames = factory.keySet();

// Loop round all the property names and get the property values
Iterator iterator = propNames.iterator();
while (iterator.hasNext()){
    String pName = (String)iterator.next();
    System.out.println(pName+"="+factory.get(pName));
}
```

Korzystanie z metod interfejsu Map nie pomija żadnych operacji sprawdzania poprawności właściwości ani konwersji.

Korzystanie z rozszerzeń JMS produktu WebSphere MQ

Klasy produktu WebSphere MQ classes for JMS zawierają zestaw rozszerzeń interfejsu API JMS o nazwie WebSphere MQ JMS extensions. Aplikacja może korzystać z tych rozszerzeń w celu dynamicznego tworzenia fabryk połączeń i miejsc docelowych w czasie wykonywania, a także do ustawiania właściwości fabryk połączeń i miejsc docelowych.

Klasy produktu WebSphere MQ classes for JMS zawierają zestaw klas w pakietach com.ibm.jms i com.ibm.mq.jms. Te klasy implementują interfejsy JMS i zawierają rozszerzenia JMS produktu WebSphere MQ . W przykładach kodu, które są zgodne, przyjęto założenie, że te pakiety zostały zaimportowane za pomocą następujących instrukcji:

```
import com.ibm.jms.*;
import com.ibm.mq.jms.*;
```

Aplikacja może używać rozszerzeń JMS produktu WebSphere MQ do wykonywania następujących funkcji:

- Tworzenie fabryk połączeń i miejsc docelowych dynamicznie w czasie wykonywania, zamiast pobierania ich jako administrowanych obiektów z przestrzeni nazw JNDI (Java Naming and Directory Interface).
- Ustawianie właściwości fabryk połączeń i miejsc docelowych

Tworzenie fabryk połączeń

Aby utworzyć fabrykę połączeń, aplikacja może używać konstruktora MQConnectionFactory , jak pokazano w poniższym przykładzie:

```
MQConnectionFactory factory = new MQConnectionFactory();
```

Ta instrukcja tworzy obiekt MQConnectionFactory z wartościami domyślnymi dla wszystkich jego właściwości, co oznacza, że aplikacja nawiązuje połączenie z domyślnym menedżerem kolejek w trybie powiązań. Jeśli aplikacja ma łączyć się w trybie klienta lub łączyć się z menedżerem kolejek innym niż domyślny menedżer kolejek, to przed utworzeniem połączenia aplikacja musi ustawić odpowiednie właściwości obiektu MQConnectionFactory . Informacje na temat sposobu wykonania tej czynności zawiera sekcja “Ustawianie właściwości fabryk połączeń” na stronie 898.

Aplikacja może utworzyć fabryki połączeń następujących typów w podobny sposób:

- Fabryka MQQueueConnection
- Fabryka MQTopicConnection
- MQXAConnectionFactory

- Fabryka MQXAQueueConnection
- Fabryka MQXATopicConnection

Ustawianie właściwości fabryk połączeń

Aplikacja może ustawić właściwości fabryki połączeń, wywołując odpowiednie metody fabryki połączeń. Fabryka połączeń może być obiektem administrowanego obiektu lub obiektem utworzonym dynamicznie w czasie wykonywania.

Należy wziąć pod uwagę następujący kod, na przykład:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_CLIENT);
factory.setQueueManager("QM1");
factory.setHostName("HOST1");
factory.setPort(1415);
factory.setChannel("QM1.SVR");
```

Ten kod tworzy obiekt MQConnectionFactory , a następnie ustawia pięć właściwości obiektu. Efektem ustawienia tych właściwości jest to, że aplikacja łączy się z menedżerem kolejek QM1 w trybie klienta przy użyciu kanału MQI o nazwie QM1.SVR. Menedżer kolejek jest uruchomiony w systemie o nazwie hosta HOST1, a nasłuchiwanie dla menedżera kolejek nasłuchuje na porcie o numerze 1415.

W celu nawiązania połączenia w czasie rzeczywistym z brokerem aplikacja może użyć następującego kodu:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_DIRECT);
factory.setHostName("HOST2");
factory.setPort(1507);
```

W tym kodzie założono, że broker jest uruchomiony w systemie o nazwie hosta HOST2 i nasłuchuje na porcie o numerze 1507.

Aplikacja, która korzysta z połączenia w czasie rzeczywistym z brokerem, może używać tylko stylu publikowania/subskrypcji przesyłania komunikatów. Nie może on używać stylu "punkt z punktem" przesyłania komunikatów.

Poprawne są tylko niektóre kombinacje właściwości fabryki połączeń. Informacje o tym, które kombinacje są poprawne, zawiera sekcja Zależności między właściwościami klas WebSphere MQ dla obiektów JMS .

Więcej informacji na temat właściwości fabryki połączeń oraz metod ustawiania jej właściwości zawiera sekcja Właściwości obiektów produktu IBM WebSphere MQ classes for JMS.

Tworzenie miejsc docelowych

Aby utworzyć obiekt kolejki, aplikacja może użyć konstruktora MQQueue, jak pokazano w poniższym przykładzie:

```
MQQueue q1 = new MQQueue("Q1");
```

Ta instrukcja tworzy obiekt MQQueue z wartościami domyślnymi dla wszystkich jego właściwości. Obiekt reprezentuje kolejkę WebSphere MQ o nazwie Q1 , która należy do lokalnego menedżera kolejek. Ta kolejka może być kolejką lokalną, kolejką aliasową lub definicją kolejki zdalnej.

Alternatywna forma konstruktora MQQueue ma dwa parametry, jak pokazano w poniższym przykładzie:

```
MQQueue q2 = new MQQueue("QM2", "Q2");
```

Obiekt MQQueue utworzony za pomocą tej instrukcji reprezentuje kolejkę WebSphere MQ o nazwie Q2 , której właścicielem jest menedżer kolejek QM2. Menedżer kolejek wskazany w ten sposób może być lokalnym menedżerem kolejek lub menedżerem kolejek zdalnych. Jeśli jest to zdalny menedżer kolejek, WebSphere MQ musi być skonfigurowany w taki sposób, aby po wysłaniu komunikatu do tego

miejsca docelowego produkt Websphere MQ mógł skierować komunikat z lokalnego menedżera kolejek do zdalnego menedżera kolejek.

Konstruktor MQQueue może także akceptować identyfikator URI (Uniform Resource Identifier) kolejki jako pojedynczy parametr. Identyfikator URI kolejki to łańcuch, który określa nazwę kolejki produktu WebSphere MQ oraz opcjonalnie nazwę menedżera kolejek, który jest właścicielem kolejki, oraz co najmniej jedną właściwość obiektu MQQueue. Następująca instrukcja zawiera przykład identyfikatora URI kolejki:

```
MQQueue q3 = new MQQueue("queue://QM3/Q3?persistence=2&priority=5");
```

Obiekt MQQueue utworzony przez tę instrukcję reprezentuje kolejkę WebSphere MQ o nazwie Q3 , która należy do menedżera kolejek QM3, a wszystkie komunikaty wysłane do tego miejsca docelowego są trwałe i mają priorytet 5. Więcej informacji na temat identyfikatorów URI kolejek zawiera sekcja “Jednolite identyfikatory zasobów (URI)” na stronie 904. Aby uzyskać alternatywny sposób ustawiania właściwości obiektu MQQueue, należy zapoznać się z sekcji “Ustawianie właściwości miejsc docelowych” na stronie 899.

Aby utworzyć obiekt tematu, aplikacja może użyć konstruktora MQTopic, tak jak pokazano w poniższym przykładzie:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
```

Ta instrukcja tworzy obiekt MQTopic z wartościami domyślnymi dla wszystkich jego właściwości. Obiekt reprezentuje temat o nazwie Sport/Football/Results.

Konstruktor MQTopic może także akceptować identyfikator URI tematu jako parametr. Identyfikator URI tematu to łańcuch, który określa nazwę tematu i opcjonalnie jedną lub większą liczbę właściwości obiektu MQTopic. Następująca instrukcja zawiera przykład identyfikatora URI tematu:

```
MQTopic t2 = new MQTopic("topic://Sport/Tennis/Results?persistence=1&priority=0");
```

Obiekt MQTopic utworzony przez tę instrukcję reprezentuje temat o nazwie Sport/Tennis/Wyniki, a wszystkie komunikaty wysłane do tego miejsca docelowego są nietrwałe i mają priorytet równy 0. Więcej informacji na temat identyfikatorów URI tematów zawiera sekcja “Jednolite identyfikatory zasobów (URI)” na stronie 904. Alternatywny sposób ustawiania właściwości obiektu MQTopic można znaleźć w sekcji “Ustawianie właściwości miejsc docelowych” na stronie 899.

Ustawianie właściwości miejsc docelowych

Aplikacja może ustawić właściwości miejsca docelowego, wywołując odpowiednie metody miejsca docelowego. Miejscem docelowym może być obiekt administrowany lub obiekt utworzony dynamicznie w czasie wykonywania.

Należy wziąć pod uwagę następujący kod, na przykład:

```
MQQueue q1 = new MQQueue("Q1");
.
q1.setPersistence(WMQConstants.WMQ_PER_PER);
q1.setPriority(5);
```

Ten kod tworzy obiekt MQQueue, a następnie ustawia dwie właściwości obiektu. Efektem ustawienia tych właściwości jest to, że wszystkie komunikaty wysyłane do miejsca docelowego są trwałe i mają priorytet 5.

Aplikacja może ustawić właściwości obiektu MQTopic w podobny sposób, jak to pokazano w poniższym przykładzie:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
.
t1.setPersistence(WMQConstants.WMQ_PER_NON);
t1.setPriority(0);
```

Ten kod tworzy obiekt MQTopic, a następnie ustawia dwie właściwości obiektu. Efektem ustawienia tych właściwości jest to, że wszystkie komunikaty wysyłane do miejsca docelowego są nietrwałe i mają priorytet równy 0.

Więcej informacji na temat właściwości miejsca docelowego oraz metod ustawiania jej właściwości zawiera sekcja Właściwości obiektów produktu IBM WebSphere MQ classes for JMS.

Budowanie połączenia w aplikacji JMS

Aby zbudować połączenie, aplikacja JMS korzysta z obiektu ConnectionFactory w celu utworzenia obiektu połączenia, a następnie uruchamia połączenie.

Aby utworzyć obiekt połączenia, aplikacja korzysta z metody createConnection() obiektu ConnectionFactory , tak jak przedstawiono to w poniższym przykładzie:

```
ConnectionFactory factory;
Connection connection;
.
.
.
connection = factory.createConnection();
```

Po utworzeniu połączenia JMS klasa IBM WebSphere MQ classes for JMS tworzy uchwyt połączenia (Hconn) i rozpoczyna konwersację z menedżerem kolejek.

Interfejs fabryczny QueueConnectioni interfejs fabryki TopicConnectiondziedziczą metodę createConnection() z interfejsu ConnectionFactory . W związku z tym można użyć metody createConnection() do utworzenia obiektu specyficznego dla domeny, tak jak pokazano to w poniższym przykładzie:

```
QueueConnectionFactory qcf;
Connection connection;
.
.
.
connection = qcf.createConnection();
```

Ten fragment kodu tworzy obiekt QueueConnection . Aplikacja może teraz wykonać niezależną od domeny operację na tym obiekcie lub operację mającą zastosowanie tylko do domeny punkt z punktem. Jeśli jednak aplikacja podejmie próbę wykonania operacji, która ma zastosowanie tylko do domeny publikowania/subskrybowania, zostanie zgłoszony wyjątek IllegalState, z następującym komunikatem:

```
JMSMQ1112: Operation for a domain specific object was not valid.
           Operation createProducer() is not valid for type com.ibm.mq.jms.MQTopic
```

Jest to spowodowane tym, że połączenie zostało utworzone z fabryki połączeń specyficznych dla domeny.

Uwaga: Należy zauważyć, że identyfikator procesu aplikacji jest używany jako domyślna tożsamość użytkownika, która ma być przekazywana do menedżera kolejek. Jeśli aplikacja działa w trybie transportu klienta, ten identyfikator procesu musi istnieć, wraz z odpowiednimi autoryzacjami, na serwerze. Jeśli ma być używana inna tożsamość, należy użyć metody createConnection(nazwa użytkownika, hasło).

Specyfikacja JMS określa, że połączenie jest tworzone w stanie `stopped` . Dopóki połączenie nie zostanie uruchomione, konsument komunikatów, który jest powiązany z połączeniem, nie może odbierać żadnych komunikatów. Aby uruchomić połączenie, aplikacja korzysta z metody start () obiektu Connection, tak jak przedstawiono to w poniższym przykładzie:

```
connection.start();
```

Tworzenie sesji w aplikacji JMS

Aby utworzyć sesję, aplikacja JMS korzysta z metody createSession() obiektu Connection.

Metoda createSession() ma dwa parametry:

1. Parametr, który określa, czy sesja jest transakowana, czy nie,

2. Parametr określający tryb potwierdzania dla sesji

Na przykład następujący kod tworzy sesję, która nie jest transakcjana i ma tryb potwierdzania AUTO_ACKNOWLEDGE:

```
Session session;
.
boolean transacted = false;
session = connection.createSession(transacted, Session.AUTO_ACKNOWLEDGE);
```

Podczas tworzenia sesji JMS klasa IBM WebSphere MQ classes for JMS tworzy uchwyt połączenia (Hconn) i rozpoczyna konwersację z menedżerem kolejek.

Obiekt Session i dowolny obiekt MessageProducer lub MessageConsumer , który został utworzony na podstawie tego obiektu, nie mogą być używane jednocześnie przez różne wątki aplikacji wielowątkowej. Najprostszym sposobem zapewnienia, że obiekty te nie są używane współbieżnie, jest utworzenie osobnego obiektu sesji dla każdego wątku.

Sesje transakcyjne w aplikacjach JMS

Aplikacje JMS mogą uruchamiać transakcje lokalne, najpierw tworząc sesję transakcyjne. Aplikacja może zatwierdzić lub wycofać transakcję.

Aplikacje JMS mogą uruchamiać transakcje lokalne. Transakcja lokalna jest transakcją, która obejmuje zmiany tylko w przypadku zasobów menedżera kolejek, z którym połączona jest aplikacja. Aby uruchomić transakcje lokalne, aplikacja musi najpierw utworzyć sesję transakcyjne, wywołując metodę createSession() obiektu Connection, określając jako parametr, że sesja jest transakcjami transakcjami. Następnie wszystkie komunikaty wysłane i odebrane w ramach sesji są pogrupowane w sekwencję transakcji. Transakcja kończy się, gdy aplikacja zatwierdza lub wycofuje komunikaty, które zostały wysłane i odebrane od momentu rozpoczęcia transakcji.

Aby zatwierdzić transakcję, aplikacja wywołuje metodę commit () obiektu Session. Jeśli transakcja zostanie zatwierdzona, wszystkie komunikaty wysłane w ramach transakcji stają się dostępne do dostarczenia do innych aplikacji, a wszystkie komunikaty odebrane w ramach transakcji zostaną potwierdzone, tak aby serwer przesyłania komunikatów nie próbował ponownie dostarczyć ich do aplikacji. W domenie typu punkt z punktem serwer przesyłania komunikatów usuwa również odebrane komunikaty z ich kolejek.

Aby wycofać transakcję, aplikacja wywołuje metodę rollback () obiektu Session. Po wycofaniu transakcji wszystkie komunikaty wysłane w ramach transakcji są odrzucane przez serwer przesyłania komunikatów, a wszystkie komunikaty odebrane w ramach transakcji stają się dostępne do dostarczenia ponownie. W domenie typu punkt z punktem komunikaty, które zostały odebrane, są ponownie umieszczane w ich kolejkach i ponownie stają się widoczne dla innych aplikacji.

Nowa transakcja jest uruchamiana automatycznie, gdy aplikacja tworzy sesję transakcyjna lub wywołuje metodę commit () lub rollback (). Oznacza to, że sesja transakcyjna zawsze ma aktywną transakcję.

Gdy aplikacja zamknie sesję transakcyjne, następuje niejawne wycofanie zmian. Jeśli aplikacja zamknie połączenie, dla wszystkich sesji transakcyjnych połączenia zostanie wykonane niejawne wycofanie zmian.

Jeśli aplikacja zakończy działanie bez zamykania połączenia, dla wszystkich sesji transakcyjnych połączenia zostanie również wykonane niejawne wycofanie zmian.

Transakcja jest w całości zawarta w ramach sesji transakcyjnej. Transakcja nie może obejmować sesji. Oznacza to, że aplikacja nie może wysyłać i odbierać komunikatów w dwóch lub większej liczby sesji transakcyjnych, a następnie zatwierdzać lub wycofywanie wszystkich tych działań jako jednej transakcji.

Tryby potwierdzania sesji JMS

Każda sesja, która nie jest transakcowana, ma tryb potwierdzenia, który określa sposób, w jaki odbierane są komunikaty odbierane przez aplikację. Dostępne są trzy tryby potwierdzenia, a wybór trybu potwierdzania wpływa na konstrukcję aplikacji.

Jeśli sesja nie jest transakowana, sposób potwierdzania komunikatów odbieranych przez aplikację jest określany przez tryb potwierdzania sesji. Trzy tryby potwierdzenia są opisane w następujących akapitach:

AUTO_POTWIERDZANIE

Sesja automatycznie potwierdzi każdy komunikat otrzymany przez aplikację.

Jeśli komunikaty są dostarczane synchronicznie do aplikacji, sesja potwierdza otrzymanie komunikatu za każdym razem, gdy wywołanie Odbiór zakończy się pomyślnie. Jeśli komunikaty są dostarczane asynchronicznie, sesja potwierdza otrzymanie komunikatu za każdym razem, gdy wywołanie metody onMessage() procesu nasłuchiwania komunikatów zakończy się pomyślnie.

Jeśli aplikacja pomyślnie odbierze komunikat, ale niepowodzenie uniemożliwia potwierdzenie wystąpienia, komunikat staje się dostępny do ponownego dostarczenia. Dlatego aplikacja musi być w stanie obsłużyć komunikat, który został ponownie dostarczony.

DUPS_OK_ACKNOWLEDGE

Sesja potwierdza komunikaty odebrane przez aplikację w czasie, gdy jest ona wybierana.

Użycie tego trybu potwierdzenia zmniejsza ilość pracy, jaką musi wykonać sesja, ale błąd, który uniemożliwia potwierdzenie komunikatu, może spowodować ponowne udostępnienie więcej niż jednego komunikatu do dostarczenia. W związku z tym aplikacja musi mieć możliwość obsługi komunikatów, które są ponownie dostarczane.

Ograniczenie: W trybach AUTO_ACKNOWLEDGE i DUPS_OK_ACKNOWLEDGE usługa JMS nie obsługuje aplikacji zgłaszających nieobsłużone wyjątki w obiekcie nasłuchiwania komunikatów. Oznacza to, że komunikaty są zawsze potwierdzane w momencie powrotu programu nasłuchującego komunikatów, niezależnie od tego, czy został on pomyślnie przetworzony (pod warunkiem, że wszystkie niepowodzenia są niekrytyczne i nie uniemożliwiają kontynuowania aplikacji). Jeśli wymagane jest dokładniejsze sterowanie potwierdzeniem komunikatów, należy użyć trybów CLIENT_ACKNOWLEDGE lub transacted, które nadają aplikacji pełną kontrolę nad funkcjami potwierdzania.

POTWIERDZANIE_KLIENTA

Aplikacja potwierdza otrzymywane przez niego komunikaty, wywołując metodę Acknowledge klasy Message.

Aplikacja może potwierdzić odbiór każdego komunikatu indywidualnie lub może otrzymać partię komunikatów i wywołać metodę Acknowledge tylko dla ostatniego komunikatu, który otrzymuje. Gdy metoda Acknowledge jest nazywana wszystkimi wiadomościami otrzymanymi od czasu ostatniego wywołania metody, są one potwierdzane.

W połączeniu z dowolnym z tych trybów potwierdzania aplikacja może zatrzymać i ponownie uruchomić dostarczanie komunikatów w sesji, wywołując metodę Recover klasy Session. Komunikaty odebrane, ale wcześniej niepotwierdzone, są ponownie dostarczane. Mogą one jednak nie być dostarczane w tej samej kolejności, w jakiej zostały dostarczone wcześniej. W międzyczasie mogły zostać wysłane komunikaty o wyższym priorytecie, a niektóre z oryginalnych komunikatów mogły utraci ważność. W domenie typu punkt z punktem niektóre z oryginalnych komunikatów mogły zostać skonsumowane przez inną aplikację.

Aplikacja może określić, czy komunikat jest ponownie dostarczany, sprawdzając zawartość pola nagłówka JMSRedelivered komunikatu. Aplikacja wykonuje tę funkcję, wywołując metodę getJMSRedelivered() klasy Message.

Tworzenie miejsc docelowych w aplikacji JMS

Zamiast pobierania miejsc docelowych jako administrowanych obiektów z przestrzeni nazw JNDI (Java Naming and Directory Interface) aplikacja JMS może używać sesji w celu dynamicznego tworzenia miejsc docelowych w czasie wykonywania. Aplikacja może użyć identyfikatora URI (Uniform Resource Identifier) w celu zidentyfikowania kolejki WebSphere MQ lub tematu oraz, opcjonalnie, w celu określenia jednej lub większej liczby właściwości obiektu Queue lub Topic.

Korzystanie z sesji do tworzenia obiektów kolejki

Aby utworzyć obiekt kolejki, aplikacja może użyć metody createQueue() obiektu Session, tak jak przedstawiono to w poniższym przykładzie:

```
Session session;
.
Queue q1 = session.createQueue("Q1");
```

Ten kod tworzy obiekt kolejki z wartościami domyślnymi dla wszystkich jego właściwości. Obiekt reprezentuje kolejkę WebSphere MQ o nazwie Q1 , która należy do lokalnego menedżera kolejek. Ta kolejka może być kolejką lokalną, kolejką aliasową lub definicją kolejki zdalnej.

Metoda createQueue() akceptuje również identyfikator URI kolejki jako parametr. Identyfikator URI kolejki to łańcuch, który określa nazwę kolejki produktu WebSphere MQ i opcjonalnie nazwę menedżera kolejek, który jest właścicielem kolejki, oraz co najmniej jedną właściwość obiektu kolejki. Następująca instrukcja zawiera przykład identyfikatora URI kolejki:

```
Queue q2 = session.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

Obiekt kolejki utworzony przez tę instrukcję reprezentuje kolejkę WebSphere MQ o nazwie Q2 , której właścicielem jest menedżer kolejek o nazwie QM2, a wszystkie komunikaty wysłane do tego miejsca docelowego są trwałe i mają priorytet 5. Menedżer kolejek wskazany w ten sposób może być lokalnym menedżerem kolejek lub menedżerem kolejek zdalnych. Jeśli jest to menedżer kolejek zdalnych, należy skonfigurować produkt WebSphere MQ w taki sposób, aby po wysłaniu komunikatu do tego miejsca docelowego produkt Websphere MQ mógł skierować komunikat z lokalnego menedżera kolejek do menedżera kolejek QM2. Więcej informacji na temat identyfikatorów URI zawiera sekcja “Jednolite identyfikatory zasobów (URI)” na stronie 904.

Należy zauważyć, że parametr w metodzie createQueue() zawiera informacje specyficzne dla dostawcy. Dlatego użycie metody createQueue() w celu utworzenia obiektu kolejki zamiast pobierania obiektu kolejki jako obiektu administrowanego z przestrzeni nazw JNDI może spowodować, że aplikacja będzie mniej przenośna.

Aplikacja może utworzyć obiekt TemporaryQueue przy użyciu metody createTemporaryQueue () obiektu Session, tak jak przedstawiono to w poniższym przykładzie:

```
TemporaryQueue q3 = session.createTemporaryQueue();
```

Chociaż do utworzenia kolejki tymczasowej używana jest sesja, to zasięg kolejki tymczasowej to połączenie, które zostało użyte do utworzenia sesji. Każdy z sesji połączenia może tworzyć producentów komunikatów i konsumentów komunikatów dla kolejki tymczasowej. Kolejka tymczasowa pozostaje do momentu zakończenia połączenia lub aplikacja jawnie usuwa kolejkę tymczasową za pomocą metody TemporaryQueue.delete (), w zależności od tego, która z tych wartości jest wcześniej.

Gdy aplikacja tworzy kolejkę tymczasową, klasy produktu WebSphere MQ dla usługi JMS tworzy kolejkę dynamiczną w menedżerze kolejek, z którym połączona jest aplikacja. Właściwość TEMPMODEL fabryki połączeń określa nazwę kolejki modelowej, która jest używana do tworzenia kolejki dynamicznej, a właściwość TEMPQPREFIX fabryki połączeń określa przedrostek używany do tworzenia nazwy kolejki dynamicznej.

Korzystanie z sesji do tworzenia obiektów tematów

Aby utworzyć obiekt tematu, aplikacja może użyć metody createTopic() obiektu Session, tak jak przedstawiono to w poniższym przykładzie:

```
Session session;
.
Topic t1 = session.createTopic("Sport/Football/Results");
```

W tym kodzie tworzony jest obiekt tematu z wartościami domyślnymi dla wszystkich jego właściwości. Obiekt reprezentuje temat o nazwie Sport/Football/Results.

Metoda createTopic() akceptuje również identyfikator URI tematu jako parametr. Identyfikator URI tematu to łańcuch, który określa nazwę tematu i opcjonalnie jedną lub większą liczbę właściwości obiektu tematu. Następujący kod zawiera przykład identyfikatora URI tematu:

```
String uri = "topic://Sport/Tennis/Results?persistence=1&priority=0";
Topic t2 = session.createTopic(uri);
```

Obiekt tematu utworzony przez ten kod reprezentuje temat o nazwie Sport/Tenni/Wyniki, a wszystkie komunikaty wysłane do tego miejsca docelowego są nietrwałe i mają priorytet równy 0. Więcej informacji na temat identyfikatorów URI tematów zawiera sekcja “Jednolite identyfikatory zasobów (URI)” na stronie 904.

Należy zauważyć, że parametr w metodzie createTopic() zawiera informacje specyficzne dla dostawcy. Dlatego użycie metody createTopic() do utworzenia obiektu tematu zamiast pobierania obiektu tematu jako obiektu administrowanego z przestrzeni nazw JNDI może spowodować, że aplikacja będzie mniej przenośna.

Aplikacja może utworzyć obiekt TemporaryTopic przy użyciu metody createTemporaryTopic () obiektu Session, tak jak przedstawiono to w poniższym przykładzie:

```
TemporaryTopic t3 = session.createTemporaryTopic();
```

Chociaż sesja jest używana do tworzenia tematu tymczasowego, to zasięg tymczasowego tematu to połączenie, które zostało użyte do utworzenia sesji. Każdy z sesji połączenia może tworzyć producentów komunikatów i konsumentów komunikatów dla tematu tymczasowego. Wątek tymczasowy pozostaje do czasu zakończenia połączenia lub aplikacja jawnie usuwa temat tymczasowy za pomocą metody TemporaryTopic.delete (), w zależności od tego, która z tych dat jest wcześniejsza.

Gdy aplikacja tworzy temat tymczasowy, klasy WebSphere MQ classes for JMS tworzą temat o nazwie rozpoczynający się od znaków TEMP/*tempTopicPrefix*, gdzie *tempTopicPrefix* to wartość właściwości TEMPTOPICPREFIX fabryki połączeń.

Jednolite identyfikatory zasobów (URI)

Identyfikator URI kolejki to łańcuch, który określa nazwę kolejki produktu WebSphere MQ i opcjonalnie nazwę menedżera kolejek, który jest właścicielem kolejki, oraz co najmniej jedną właściwość obiektu kolejki utworzonego przez aplikację. Identyfikator URI tematu to łańcuch, który określa nazwę tematu i, opcjonalnie, jedną lub więcej właściwości obiektu tematu utworzonego przez aplikację.

Identyfikator URI kolejki ma następujący format:

```
queue://[qMgrName]/qName[?propertyName1=propertyValue1
                         &propertyName2=propertyValue2
                         &...]
```

Identyfikator URI tematu ma następujący format:

```
topic://topicName[?propertyName1=propertyValue1
                  &propertyName2=propertyValue2
                  &...]
```

Zmienne w tych formatach mają następujące znaczenia:

NazwaqMgr
Nazwa menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez identyfikator URI.

Menedżer kolejek może być lokalnym menedżerem kolejek lub menedżerem kolejek zdalnych. Jeśli jest to menedżer kolejek zdalnych, należy skonfigurować produkt WebSphere MQ w taki sposób, aby po wysłaniu komunikatu do kolejki przez aplikację produkt Websphere MQ mógł skierować komunikat z lokalnego menedżera kolejek do zdalnego menedżera kolejek.

Jeśli nie zostanie podana żadna nazwa, przyjmowany jest lokalny menedżer kolejek.

qName
Nazwa kolejki produktu WebSphere MQ .

Kolejka może być kolejką lokalną, kolejką aliasową lub definicją kolejki zdalnej.

Reguły tworzenia nazw kolejek znajdują się w sekcji Reguły nazewnictwa obiektów IBM WebSphere MQ .

topicName
Nazwa tematu.

Reguły tworzenia nazw tematów znajdują się w sekcji Reguły nazewnictwa obiektów IBM WebSphere MQ. Unikaj używania znaków wieloznacznych +, #, *, i? w nazwach tematów. Nazwy tematów zawierające te znaki mogą powodować nieoczekiwane wyniki podczas subskrybowania tych znaków. Patrz sekcja Korzystanie z łańcuchów tematów.

***propertyName1*, *propertyName2*, ...**
Nazwy właściwości obiektu kolejki lub tematu utworzonego przez aplikację. Tabela 123 na stronie 905 zawiera listę poprawnych nazw właściwości, które mogą być używane w identyfikatorze URI.

Jeśli nie zostaną określone żadne właściwości, obiekt Queue lub Topic ma wartości domyślne dla wszystkich jej właściwości.

***propertyValue1*, *propertyValue2*, ...**
Wartości właściwości obiektu Queue lub Topic utworzonego przez aplikację. Tabela 123 na stronie 905 zawiera listę poprawnych wartości właściwości, które mogą być używane w identyfikatorze URI.

Nawiasy kwadratowe ([]) oznaczają opcjonalny komponent, a wielokropek (...) oznacza, że lista par nazwa-wartość właściwości, jeśli jest obecna, może zawierać jedną lub więcej par nazwa-wartość.

Tabela 123 na stronie 905 zawiera listę poprawnych nazw właściwości i poprawnych wartości, które mogą być używane w identyfikatorach URI kolejek i tematów. Mimo że narzędzie administracyjne WebSphere MQ JMS używa stałych symbolicznych dla wartości właściwości, identyfikatory URI nie mogą zawierać stałych symbolicznych.

Tabela 123. Nazwy właściwości i poprawne wartości do użycia w identyfikatorach URI kolejek i tematów

Nazwa właściwości	**Opis**	**Poprawne wartości**
CCSID	Sposób, w jaki dane znakowe w treści komunikatu są reprezentowane, gdy klasy WebSphere MQ classes for JMS przekaże komunikat do miejsca docelowego.	• Dowolny identyfikator kodowanego zestawu znaków obsługiwany przez produkt WebSphere MQ.
encoding	Sposób, w jaki dane liczbowe w treści komunikatu są reprezentowane, gdy klasy WebSphere MQ classes for JMS przekaże komunikat do miejsca docelowego.	• Dowolna poprawna wartość w polu *Kodowanie* w deskryptorze komunikatu produktu WebSphere MQ .
Utrata ważności	Czas życia komunikatów wysyłanych do miejsca docelowego	• -2-Jak określono w wywołaniu funkcji send () lub, jeśli nie określono w wywołaniu funkcji send (), domyślny czas życia dla producenta wiadomości. • 0-Komunikat wysyłany do miejsca docelowego nigdy nie traci ważności. • Dodatnia liczba całkowita określająca czas życia (w milisekundach).

Tabela 123. Nazwy właściwości i poprawne wartości do użycia w identyfikatorach URI kolejek i tematów (kontynuacja)

Nazwa właściwości	**Opis**	**Poprawne wartości**
rozsyłanie	Ustawienie rozsyłania grupowego dla tematu podczas korzystania z połączenia w czasie rzeczywistym z brokerem	Poniższa lista zawiera poprawne wartości. Wartość powiązana z każdą wartością jest odpowiednią wartością właściwości MULTICAST, która jest używana w narzędziu administracyjnym WebSphere MQ JMS. Opis właściwości MULTICAST i jego poprawnych wartości znajduje się w sekcji Właściwości obiektów IBM WebSphere MQ classes for JMS. • -1-ASCF • 0 - wyłączone • 3-NOTR • 5-NIEZAWODNE • 7-WŁĄCZONE
trwałość	Trwałość komunikatów wysłanych do miejsca docelowego	• -2-Jak określono w wywołaniu funkcji send () lub, jeśli nie określono w wywołaniu funkcji send (), to domyślna trwałość producenta komunikatów. • -1-Jak określono w atrybucie *DefPersistence* w kolejce lub temacie WebSphere MQ . • 1-Nietrwały. • 2-Trwałe. • 3-ekwiwalent wartości HIGH dla właściwości PERSISTENCE, który jest używany w narzędziu administracyjnym WebSphere MQ JMS. Wyjaśnienie tej wartości znajduje się w sekcji “Trwałe komunikaty JMS” na stronie 931.
priorytet	Priorytet komunikatów wysyłanych do miejsca docelowego	• -2-Jak określono w wywołaniu funkcji send () lub, jeśli nie określono w wywołaniu funkcji send (), domyślnym priorytetem jest producent komunikatów. • -1-Jak określono w atrybucie *DefPriority* w kolejce lub temacie WebSphere MQ . • Liczba całkowita z zakresu od 0 do 9 określająca priorytet komunikatów wysyłanych do miejsca docelowego.

Tabela 123. Nazwy właściwości i poprawne wartości do użycia w identyfikatorach URI kolejek i tematów (kontynuacja)

Nazwa właściwości	Opis	Poprawne wartości
targetClient	Określa, czy komunikaty wysyłane do miejsca docelowego zawierają nagłówek MQRFH2 .	• 0-Komunikaty zawierają nagłówek MQRFH2 . • 1-Komunikaty nie zawierają nagłówka MQRFH2 .

Na przykład następujący identyfikator URI identyfikuje kolejkę WebSphere MQ o nazwie Q1 , która należy do lokalnego menedżera kolejek. Obiekt kolejki utworzony przy użyciu tego identyfikatora URI ma wartości domyślne dla wszystkich jego właściwości.

```
queue:///Q1
```

Poniższy identyfikator URI identyfikuje kolejkę WebSphere MQ o nazwie Q2 , która należy do menedżera kolejek o nazwie QM2. Wszystkie komunikaty wysłane do tego miejsca docelowego mają priorytet 6. Pozostałe właściwości obiektu Queue utworzonego przy użyciu tego identyfikatora URI mają swoje wartości domyślne.

```
queue://QM2/Q2?priority=6
```

Poniższy identyfikator URI identyfikuje temat o nazwie Sport/Athletics/Results. Wszystkie komunikaty wysłane do tego miejsca docelowego są nietrwałe i mają priorytet równy 0. Pozostałe właściwości obiektu tematu utworzonego przy użyciu tego identyfikatora URI mają swoje wartości domyślne.

```
topic://Sport/Athletics/Results?persistence=1&priority=0
```

Wysyłanie komunikatów w aplikacji JMS

Zanim aplikacja JMS może wysyłać komunikaty do miejsca docelowego, musi najpierw utworzyć obiekt MessageProducer dla miejsca docelowego. Aby wysłać komunikat do miejsca docelowego, aplikacja tworzy obiekt Message, a następnie wywołuje metodę send () obiektu MessageProducer .

Aplikacja korzysta z obiektu MessageProducer do wysyłania komunikatów. Aplikacja zwykle tworzy obiekt MessageProducer dla określonego miejsca docelowego, które może być kolejką lub tematem, tak aby wszystkie komunikaty wysłane przy użyciu producenta komunikatów były wysyłane do tego samego miejsca docelowego. Dlatego zanim aplikacja może utworzyć obiekt MessageProducer , musi on najpierw utworzyć obiekt kolejki lub tematu. Informacje na temat tworzenia kolejki lub obiektu tematu zawierają następujące tematy:

- “Używanie interfejsu JNDI do pobierania administrowanych obiektów w aplikacji JMS” na stronie 889
- “Korzystanie z rozszerzeń IBM JMS” na stronie 890
- “Korzystanie z rozszerzeń JMS produktu WebSphere MQ” na stronie 897
- “Tworzenie miejsc docelowych w aplikacji JMS” na stronie 902

Aby utworzyć obiekt MessageProducer , aplikacja korzysta z metody createProducer() obiektu Session, tak jak przedstawiono to w poniższym przykładzie:

```
MessageProducer producer = session.createProducer(destination);
```

Parametr `destination` to obiekt kolejki lub tematu, który został wcześniej utworzony przez aplikację.

Zanim aplikacja może wysłać komunikat, musi utworzyć obiekt komunikatu. Treść komunikatu zawiera dane aplikacji, a usługa JMS definiuje pięć typów treści komunikatu:

- Bajty
- Odwzoruj

- Obiekt
- Strumień
- Tekstowy

Każdy typ treści komunikatu ma własny interfejs JMS, który jest podinterfejsem interfejsu komunikatu, a także metodę w interfejsie sesji w celu utworzenia komunikatu z tym typem treści. Na przykład interfejs dla komunikatu tekstowego ma nazwę TextMessage, a aplikacja korzysta z metody createTextMessage () obiektu Session w celu utworzenia komunikatu tekstowego, tak jak przedstawiono to w następującej instrukcji:

```
TextMessage outMessage = session.createTextMessage(outString);
```

Więcej informacji na temat komunikatów i treści komunikatów zawiera sekcja "Komunikaty JMS" na stronie 829.

Aby wysłać komunikat, aplikacja używa metody send () obiektu MessageProducer , tak jak przedstawiono to w poniższym przykładzie:

```
producer.send(outMessage);
```

Aplikacja może używać metody send () do wysyłania komunikatów w dowolnej domenie przesyłania komunikatów. Rodzaj miejsca docelowego określa, która domena przesyłania komunikatów jest używana. Jednak interfejs TopicPublisher, podinterfejs MessageProducer , który jest specyficzny dla domeny publikowania/subskrybowania, ma również metodę publikowania (), która może być używana zamiast metody send (). Te dwie metody są funkcjonalnie takie same.

Aplikacja może utworzyć obiekt MessageProducer bez określonego miejsca docelowego. W takim przypadku aplikacja musi określić miejsce docelowe podczas wywoływania metody send ().

Jeśli aplikacja wysyła komunikat w ramach transakcji, komunikat nie zostanie dostarczony do miejsca docelowego, dopóki transakcja nie zostanie zatwierdzona. Oznacza to, że aplikacja nie może wysłać komunikatu i otrzymać odpowiedzi na komunikat w ramach tej samej transakcji.

Miejsce docelowe można skonfigurować w taki sposób, aby gdy aplikacja wysyła do niego komunikaty, klasy produktu WebSphere MQ dla usługi JMS przekazały komunikat i zwracają sterowanie z powrotem do aplikacji bez określania, czy menedżer kolejek odebrał komunikat w sposób bezpieczny. Jest to czasami nazywane *umieszczonym asynchronicznie*. Więcej informacji na ten temat zawiera sekcja "Asynchronicznie umieszczanie komunikatów w klasach produktu IBM WebSphere MQ dla usługi JMS" na stronie 946.

Odbieranie komunikatów w aplikacji JMS

Aplikacja używa konsumenta komunikatów do odbierania komunikatów. Trwałym subskrybentem tematu jest konsument komunikatów, który odbiera wszystkie komunikaty wysłane do miejsca docelowego, włącznie z tymi, które są wysyłane, gdy konsument jest nieaktywny. Aplikacja może wybrać, które komunikaty mają być odbierane za pomocą selektora komunikatów, i może odbierać komunikaty asynchronicznie, korzystając z obiektu nasłuchiwania komunikatów.

Aplikacja korzysta z obiektu MessageConsumer do odbierania komunikatów. Aplikacja tworzy obiekt MessageConsumer dla określonego miejsca docelowego, które może być kolejką lub tematem, dzięki czemu wszystkie komunikaty odebrane z użyciem konsumenta komunikatów są odbierane z tego samego miejsca docelowego. Dlatego zanim aplikacja może utworzyć obiekt MessageConsumer , musi on najpierw utworzyć obiekt kolejki lub tematu. Informacje na temat tworzenia kolejki lub obiektu tematu zawierają następujące tematy:

- "Używanie interfejsu JNDI do pobierania administrowanych obiektów w aplikacji JMS" na stronie 889
- "Korzystanie z rozszerzeń IBM JMS" na stronie 890
- "Korzystanie z rozszerzeń JMS produktu WebSphere MQ" na stronie 897
- "Tworzenie miejsc docelowych w aplikacji JMS" na stronie 902

Aby utworzyć obiekt MessageConsumer , aplikacja korzysta z metody createConsumer() obiektu Session, tak jak przedstawiono to w poniższym przykładzie:

```
MessageConsumer consumer = session.createConsumer(destination);
```

Parametr `destination` to obiekt kolejki lub tematu, który został wcześniej utworzony przez aplikację.

Następnie aplikacja korzysta z metody receive () obiektu MessageConsumer w celu odebrania komunikatu z miejsca docelowego, jak pokazano w poniższym przykładzie:

```
Message inMessage = consumer.receive(1000);
```

Parametr w wywołaniu metody receive () określa czas (w milisekundach), przez jaki metoda oczekuje na dotarcie odpowiedniego komunikatu, jeśli komunikat nie jest dostępny natychmiast. Jeśli ten parametr zostanie pominięty, połączenia będą blokowe przez czas nieokreślony aż do momentu nadejścia odpowiedniego komunikatu. Jeśli nie chcesz, aby aplikacja oczekiwała na komunikat, użyj metody receiveNoWait ().

Metoda receive () zwraca komunikat o określonym typie. Na przykład, gdy aplikacja odbierze komunikat tekstowy, obiekt zwrócony przez wywołanie receive () jest obiektem typu TextMessage .

Jednak zadeklarowany typ obiektu zwrócony przez wywołanie receive () jest obiektem komunikatu. Dlatego w celu wyodrębnienia danych z treści komunikatu, który został właśnie odebrany, aplikacja musi rzutować z klasy Message na bardziej konkretną podklasę, taką jak TextMessage. Jeśli typ komunikatu nie jest znany, aplikacja może użyć operatora `instanceof` w celu określenia typu. Zawsze zaleca się stosowanie aplikacji w celu określenia typu komunikatu przed rzutowaniem, dzięki czemu błędy mogą być obsługiwane w sposób wdzięczny.

Poniższy kod używa operatora `instanceof` i pokazuje, jak wyodrębnić dane z treści wiadomości tekstowej:

```
if (inMessage instanceof TextMessage) {
  String replyString = ((TextMessage) inMessage).getText();
  .
  .
  .
} else {
  // Print error message if Message was not a TextMessage.
  System.out.println("Reply message was not a TextMessage");
}
```

Jeśli aplikacja wysyła komunikat w ramach transakcji, komunikat nie zostanie dostarczony do miejsca docelowego, dopóki transakcja nie zostanie zatwierdzona. Oznacza to, że aplikacja nie może wysłać komunikatu i otrzymać odpowiedzi na komunikat w ramach tej samej transakcji.

Jeśli konsument komunikatów odbierze komunikaty z miejsca docelowego skonfigurowanego do odczytu z wyprzedzeniem, wszystkie nietrwałe komunikaty znajdujące się w buforze odczytu z wyprzedzeniem po zakończeniu działania aplikacji są usuwane.

W domenie publikowania/subskrybowania usługa JMS identyfikuje dwa typy konsumenta komunikatów, nietrwały subskrybent tematu i trwały subskrybent tematu, które są opisane w następujących dwóch sekcjach.

Nietrwałe subskrybenty tematów

Nietrwały subskrybent tematu otrzymuje tylko te komunikaty, które są publikowane, gdy subskrybent jest aktywny. Subskrypcja nietrwała jest uruchamiana, gdy aplikacja tworzy nietrwały subskrybent tematu i kończy się, gdy aplikacja zamknie subskrybenta, lub gdy subskrybent nie wchodzi w zakres. Jako rozszerzenie w klasach WebSphere MQ classes for JMS, subskrybent tematu nietrwałego otrzymuje również zachowane publikacje, ale nie w czasie rzeczywistym połączenia z brokerem.

Aby utworzyć nietrwały subskrybent tematów, aplikacja może użyć niezależnej metody createConsumer() domeny, określając obiekt tematu jako miejsce docelowe. Alternatywnie aplikacja może użyć metody createSubscriber() specyficznej dla domeny, tak jak przedstawiono to w poniższym przykładzie:

```
TopicSubscriber subscriber = session.createSubscriber(topic);
```

Parametr `topic` jest obiektem tematu, który został wcześniej utworzony przez aplikację.

Trwałe subskrybenty tematów

Ograniczenie: Aplikacja nie może utworzyć trwałych subskrybentów tematów podczas korzystania z połączenia w czasie rzeczywistym z brokerem.

Trwały subskrybent tematu odbiera wszystkie komunikaty opublikowane w czasie trwania trwałej subskrypcji. Komunikaty te obejmują wszystkie te komunikaty, które są publikowane, gdy subskrybent nie jest aktywny. Jako rozszerzenie w klasach WebSphere MQ classes for JMS, subskrybent trwałych tematów otrzymuje również zachowane publikacje.

Aby utworzyć trwały subskrybent tematów, aplikacja korzysta z metody createDurableSubscriber () obiektu Session, tak jak przedstawiono to w poniższym przykładzie:

```
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001");
```

W wywołaniu metody createDurableSubscriber () pierwszy parametr jest obiektem tematu, który został wcześniej utworzony przez aplikację, a drugim parametrem jest nazwa używana do identyfikowania trwałej subskrypcji.

Sesja użyta do utworzenia trwałego subskrybenta tematów musi mieć powiązany identyfikator klienta. Identyfikator klienta powiązany z sesją jest taki sam, jak identyfikator klienta dla połączenia używanego do utworzenia sesji. Identyfikator klienta można określić, ustawiając właściwość CLIENTID obiektu ConnectionFactory . Alternatywnie, aplikacja może określić identyfikator klienta, wywołując metodę setClientID () obiektu połączenia.

Nazwa używana do identyfikowania trwałej subskrypcji musi być unikalna tylko w obrębie identyfikatora klienta, a zatem identyfikator klienta stanowi część pełnego, unikalnego identyfikatora trwałej subskrypcji. Aby kontynuować korzystanie z trwałej subskrypcji, która została wcześniej utworzona, aplikacja musi utworzyć trwałą subskrybent tematu przy użyciu sesji o takim samym identyfikatorze klienta, jak ten powiązany z trwałą subskrypcją, i przy użyciu tej samej nazwy subskrypcji.

Trwała subskrypcja jest uruchamiana, gdy aplikacja tworzy trwały subskrybent tematu przy użyciu identyfikatora klienta i nazwy subskrypcji, dla której nie istnieje obecnie trwała subskrypcja. Jednak trwała subskrypcja nie kończy się, gdy aplikacja zamknie trwałą subskrybent tematu. Aby zakończyć trwałą subskrypcję, aplikacja musi wywołać metodę unsubscribe () obiektu Session, który ma taki sam identyfikator klienta, jak ten powiązany z trwałą subskrypcją. Parametrem w wywołaniu unsubscribe () jest nazwa subskrypcji, tak jak przedstawiono to w poniższym przykładzie:

```
session.unsubscribe("D_SUB_000001");
```

Zasięgiem trwałej subskrypcji jest menedżer kolejek. Jeśli w jednym menedżerze kolejek istnieje trwała subskrypcja, a aplikacja połączona z innym menedżerem kolejek tworzy trwałą subskrypcję z tym samym identyfikatorem klienta i nazwą subskrypcji, to dwie trwałe subskrypcje są całkowicie niezależne.

Selektory komunikatów

Aplikacja może określić, że tylko te komunikaty, które spełniają określone kryteria, są zwracane przez kolejne wywołania receive (). Podczas tworzenia obiektu MessageConsumer aplikacja może określić wyrażenie SQL (Structured Query Language) określające, które komunikaty są pobierane. To wyrażenie SQL jest nazywane *selektorem komunikatów*. Selektor komunikatów może zawierać nazwy pól nagłówka komunikatu JMS oraz właściwości komunikatu. Informacje na temat konstruowania selektora komunikatów zawiera sekcja “Selektory komunikatów w usłudze JMS” na stronie 830.

W poniższym przykładzie przedstawiono, w jaki sposób aplikacja może wybierać komunikaty na podstawie właściwości zdefiniowanej przez użytkownika o nazwie myProp:

```
MessageConsumer consumer;
.
consumer = session.createConsumer(destination, "myProp = 'blue'");
```

Specyfikacja JMS nie zezwala aplikacji na zmianę selektora komunikatów konsumenta komunikatów. Po utworzeniu przez aplikację konsumenta komunikatów z selektorem komunikatów selektor komunikatów pozostaje w stanie życia tego konsumenta. Jeśli aplikacja wymaga więcej niż jednego selektora komunikatów, aplikacja musi utworzyć konsumenta komunikatów dla każdego selektora komunikatów.

Należy zwrócić uwagę, że jeśli aplikacja jest połączona z menedżerem kolejek w wersji 7, właściwość MSGSELECTION fabryki połączeń nie ma wpływu. Aby zoptymalizować wydajność, wszystkie opcje wyboru komunikatów są wykonywane przez menedżer kolejek.

Pomijanie publikacji lokalnych

Aplikacja może utworzyć konsument komunikatów, który ignoruje publikacje opublikowane w ramach połączenia własnego konsumenta. Aplikacja wykonuje tę operację, ustawiając trzeci parametr w wywołaniu metody createConsumer() na wartość `true`, jak pokazano w poniższym przykładzie:

```
MessageConsumer consumer = session.createConsumer(topic, null, true);
```

W wywołaniu metody createDurableSubscriber () aplikacja wykonuje tę funkcję, ustawiając czwarty parametr na wartość `true`, jak to pokazano w poniższym przykładzie.

```
String selector = "company = 'IBM'";
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001",
                                                              selector, true);
```

Asynchroniczne dostarczanie komunikatów

Aplikacja może odbierać komunikaty asynchronicznie, rejestrując obiekt nasłuchiwania komunikatów za pomocą konsumenta komunikatów. Program nasłuchujący komunikatów ma metodę o nazwie onMessage, która jest wywoływana asynchronicznie, gdy dostępny jest odpowiedni komunikat i którego celem jest przetworzenie komunikatu. Poniższy kod ilustruje mechanizm:

```
import javax.jms.*;

public class MyClass implements MessageListener
{
  // The method that is called asynchronously when a suitable message is available
  public void onMessage(Message message)
  {
    System.out.println("Message is "+message);

    // The code to process the message
    .
    .
    .
  }
}
.
.
.
// Main program (possibly in another class)
.
// Creating the message listener
MyClass listener = new MyClass();

// Registering the message listener with a message consumer
consumer.setMessageListener(listener);

// The main program now continues with other processing
```

Aplikacja może używać sesji w celu synchronicznego odbierania komunikatów za pomocą wywołań receive () lub asynchronicznego odbierania komunikatów za pomocą programów nasłuchujących komunikatów, ale nie dla obu tych elementów. Jeśli aplikacja musi odbierać komunikaty synchronicznie i asynchronicznie, musi utworzyć oddzielne sesje.

Po ustawieniu sesji w celu asynchronicznego odbierania komunikatów nie można wywoływać następujących metod w tej sesji ani w obiektach utworzonych w tej sesji:

- MessageConsumer.receive ()
- MessageConsumer.receive (long)
- MessageConsumer.receiveNoWait ()
- Session.acknowledge()
- MessageProducer.send (miejsce docelowe, komunikat)
- MessageProducer.send (Destination, Message, int, int, long)
- MessageProducer.send (Message)
- MessageProducer.send (Message, int, int, long)
- Session.commit()
- Session.createBrowser(Kolejka)
- Session.createBrowser(Kolejka, Łańcuch)
- Session.createBytesMessage()
- Session.createConsumer(Miejsce docelowe)
- Session.createConsumer(Destination, String, boolean).
- Session.createDurableSubscriber(Topic, String)
- Session.createDurableSubscriber(Topic, String, String, boolean)
- Session.createMapMessage()
- Session.createMessage()
- Session.createObjectMessage()
- Session.createObjectMessage(Serializable)
- Session.createProducer(Miejsce docelowe)
- Session.createQueue(String).
- Session.createStreamMessage()
- Session.createTemporaryQueue()
- Session.createTemporaryTopic()
- Session.createTextMessage()
- Session.createTextMessage(String)
- Session.createTopic()
- Session.getAcknowledgeMode()
- Session.getMessageListener()
- Session.getTransacted()
- Session.rollback()
- Session.unsubscribe(String).

Jeśli zostanie wywołana dowolna z tych metod, zostanie zgłoszony wyjątek JMSException zawierający komunikat:

```
JMSCC0033: synchroniczne wywołanie metody nie jest dozwolone, gdy sesja jest używana
asynchronicznie: 'nazwa metody'
```

.

Odbieranie komunikatów trujących

Aplikacja może otrzymać komunikat, który nie może zostać przetworzony. Może istnieć kilka przyczyn, dla których komunikat nie może zostać przetworzony, na przykład komunikat może mieć niepoprawny

format. Takie komunikaty są opisywane jako nieprzetwarzalne komunikaty i wymagają specjalnej obsługi, aby zapobiec rekurencyjnemu przetwarzaniu komunikatu.

Szczegółowe informacje na temat obsługi nieprzetwarzalnych komunikatów zawiera sekcja "Obsługa komunikatów trujących w produkcie IBM WebSphere MQ classes for JMS" na stronie 914.

V 7.5.0.8 Pobieranie danych użytkownika subskrypcji

Jeśli komunikaty, które aplikacja IBM WebSphere MQ classes for JMS konsumuje z kolejki, są umieszczane przez zdefiniowaną administracyjnie trwałą subskrypcję, aplikacja musi uzyskać dostęp do informacji o danych użytkownika powiązanych z subskrypcją. Te informacje są dodawane do komunikatu jako właściwość.

W produkcie Version 7.5.0, Fix Pack 8, gdy komunikat jest pobierany z kolejki, która zawiera nagłówek RFH2 z folderem MQPS, wartość powiązana z kluczem Sud, jeśli istnieje, jest dodawana jako właściwość String do obiektu komunikatu JMS zwracanego do aplikacji IBM WebSphere MQ classes for JMS . Aby włączyć pobieranie tej właściwości z komunikatu, należy użyć stałej JMS_IBM_SUBSCRIPTION_USER_DATA w interfejsie JmsConstants z metodą javax.jms.Message.getStringProperty(java.lang.String), aby pobrać dane użytkownika subskrypcji.

W poniższym przykładzie subskrypcja trwała administracyjna jest definiowana za pomocą komendy MQSC **DEFINE SUB**:

```
DEFINE SUB('MY.SUBCRIPTION') TOPICSTR('PUBLIC') DEST('MY.SUBSCRIPTION.Q')
USERDATA('Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q')
```

Kopie komunikatów, które są publikowane w łańcuchu tematu PUBLIC , są umieszczane w kolejce, MY.SUBSCRIPTION.Q. Dane użytkownika powiązane z trwałą subskrypcją są następnie dodawane jako właściwość do wiadomości, która jest przechowywana w folderze MQPS nagłówka RFH2 z kluczem Sud.

Aplikacja IBM WebSphere MQ classes for JMS może wywołać następujące informacje:

```
javax.jms.Message.getStringProperty(JmsConstants.JMS_IBM_SUBSCRIPTION_USER_DATA);
```

Następnie zwracany jest następujący łańcuch:

```
Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q
```

.

Pojęcia pokrewne

"Nagłówek MQRFH2 i JMS" na stronie 834

Zadania pokrewne

Definiowanie subskrypcji administracyjnej

Odsyłacze pokrewne

DEFINE SUB

Interfejs JmsConstants

Zamykanie klas produktu WebSphere MQ dla aplikacji JMS

Ważne jest, aby klasy produktu WebSphere MQ dla aplikacji JMS zamykano niektóre obiekty JMS jawnie przed zatrzymaniem. Nie można wywoływać procesów finalizujących, dlatego nie należy ich używać do wolnych zasobów. Nie zezwalaj na zakończenie działania aplikacji ze skompresowanym śledzeniem.

Samo czyszczenie pamięci nie może w odpowiednim czasie zwolnić wszystkich klas produktu WebSphere MQ dla zasobów JMS i WebSphere MQ , zwłaszcza jeśli aplikacja tworzy wiele krótkotrych obiektów JMS na poziomie sesji lub niższych. Dlatego ważne jest, aby aplikacja zamykała obiekt Connection, Session, MessageConsumerlub obiekt MessageProducer , gdy nie jest już wymagany.

Jeśli aplikacja zakończy działanie bez zamykania połączenia, dla wszystkich sesji transaktywnych połączenia zostanie wykonane niejawne wycofanie zmian. Aby zapewnić, że wszystkie zmiany

wprowadzone przez aplikację zostaną zatwierdzone, należy zamknąć połączenie jawnie przed zamknięciem aplikacji.

Nie należy używać finalizatorów w aplikacji do zamykania obiektów JMS. Ponieważ nie można wywołać procesu finalizowania, zasoby mogą nie zostać zwolnione. Po zamknięciu połączenia zamyka wszystkie sesje utworzone na jego podstawie. Podobnie elementy MessageConsumers i MessageProducers utworzone na podstawie sesji są zamykane po zamknięciu sesji. Jednak należy jawnie rozważyć zamknięcie sesji, MessageConsumersi MessageProducers w celu zapewnienia, że zasoby są zwalniane w odpowiednim czasie.

Jeśli kompresja śledzenia jest aktywowana, wyłączenia System.Halt() i nieprawidłowe, niekontrolowane zakończenie działania maszyny JVM prawdopodobnie spowodują uszkodzenie pliku śledzenia. Jeśli to możliwe, wyłącz narzędzie śledzenia, gdy zebrano potrzebne informacje śledzenia. W przypadku śledzenia aplikacji aż do nieprawidłowego zakończenia należy użyć nieskompresowanych danych wyjściowych śledzenia.

Obsługa komunikatów trujących w produkcie IBM WebSphere MQ classes for JMS

Komunikat nieprzetwarzalny to komunikat, który nie może być przetwarzany przez odbierającą aplikację MDB. Jeśli zostanie napotkany komunikat nieprzetwarzalny, obiekty JMS MessageConsumer i ConnectionConsumer mogą je ponownie przekwalifikować zgodnie z dwiema właściwościami kolejki, BOQNAME i BOTHRESH.

Czasami w kolejce pojawia się źle sformatowany komunikat. W tym kontekście niepoprawnie sformatowany komunikat oznacza, że aplikacja odbierający nie może poprawnie przetworzyć komunikatu. Taki komunikat może spowodować, że aplikacja odbierający nie powiedzie się, a następnie wycofa się ze źle sformatowanego komunikatu. Następnie komunikat może być wielokrotnie dostarczany do kolejki wejściowej i wielokrotnie wycofany przez aplikację. Te komunikaty są nazywane *komunikatami nieprzetwarzanymi*. Obiekt MessageConsumer JMS wykrywa komunikaty nieprzetwarzalne i przekierowuje je do alternatywnego miejsca docelowego.

Menedżer kolejek produktu IBM WebSphere MQ przechowuje zapis liczby operacji, w których każda wiadomość została wycofana. Gdy ta liczba osiągnie konfigurowalną wartość progową, konsument komunikatu ponownie przeczy komunikat do nazwanej kolejki wycofania. Jeśli ponowne wykonanie nie powiedzie się z jakiegokolwiek powodu, komunikat zostanie usunięty z kolejki wejściowej i zostanie ponownie wysłany do kolejki niedostarczonych komunikatów lub usunięty. Więcej szczegółów na ten temat zawiera sekcja “Usuwanie komunikatów z kolejki w ASF” na stronie 955.

Istnieje różnica między sposobem, w jaki komunikaty nieprzetwarzalne są ponownie wysyłane przez elementy MessageConsumers i ConnectionConsumers. ConnectionConsumers są w stanie requeować komunikaty nieprzetwarzalne bez wpływu na dostarczanie komunikatów. Proces requeue jest przetwarzany poza dowolną jednostką pracy powiązaną z rzeczywistym dostarczaniu komunikatu do kodu aplikacji. Jest to możliwe ze względu na wielowątkowy charakter operacji ConnectionConsumer .

Jednak MessageConsumerssą jednowątkowe na poziomie niższym niż sesja, a każde przekwalifikowanie komunikatów nieprzetwarzalnych odbywa się w ramach bieżącej jednostki pracy. Nie wpływa to na działanie aplikacji, jednak w przypadku, gdy komunikaty nieprzetwarzalne są ponownie wykonywane w ramach sesji transakcyjnych lub sesji potwierdzania klienta, sama czynność requeue nie jest zatwierdzana do momentu zatwierdzenia bieżącej jednostki pracy przez kod aplikacji lub, w razie potrzeby, kodu kontenera aplikacji.

Obiekty JMS ConnectionConsumer obsługują komunikaty nieprzetwarzalne w ten sam sposób i korzystają z tych samych właściwości kolejki. Jeśli wiele konsumentów połączeń monitoruje tę samą kolejkę, możliwe jest, że komunikat nieprzetwarzalny może zostać dostarczony do aplikacji więcej razy niż wartość progowa, zanim nastąpi ponowne wystąpienie kolejki. Takie zachowanie jest spowodowane sposobem, w jaki klienci połączenia indywidualnego monitorują kolejki i komunikaty nieprzetwarzalne w kolejce.

Wartość progowa i nazwa kolejki zaplecza są atrybutami kolejki produktu IBM WebSphere MQ . Nazwy atrybutów to BackoutThreshold i BackoutRequeueQName. Kolejka, do której mają zastosowanie, jest następująca:

- W przypadku przesyłania komunikatów w trybie punkt z punktem jest to podstawowa kolejka lokalna. Jest to ważne, gdy konsumenci komunikatów i konsumenci połączą się z aliasami kolejek.
- W przypadku przesyłania komunikatów w trybie publikowania/subskrypcji w normalnym trybie dostawcy przesyłania komunikatów produktu IBM WebSphere MQ kolejka zarządzana tematu jest tworzona z kolejki modelowej.
- W przypadku przesyłania komunikatów publikowania/subskrypcji w trybie migracji dostawcy przesyłania komunikatów produktu IBM WebSphere MQ jest to kolejka CCSUB zdefiniowana w obiekcie fabryki TopicConnectionlub w kolejce CCDSUB zdefiniowanej w obiekcie tematu.

IBM WebSphere MQ classes for JMS odpytuje BackoutThreshold i BackoutRequeueQName kolejki. W związku z tym należy przyznać dostęp do zapytania w kolejce do użytkownika, który uruchomił aplikację.

V 7.5.0.9 Jeśli kolejka docelowa jest kolejką klastra, wymagane uprawnienia zależą od wersji używanej partycji IBM WebSphere MQ classes for JMS :

- Jeśli używany jest produkt IBM WebSphere MQ classes for JMS for Version 7.5.0, Fix Pack 9 plus poprawka tymczasowa dla raportu APAR IT26482, wymagany jest dostęp z zapytaniem.
- W przypadku wszystkich pozostałych wersji, należy uzyskać dostęp do informacji o zapytaniu, przeglądać i uzyskiwać dostęp.

Aby ustawić atrybuty nazwy QName BackoutThreshold i BackoutRequeue, należy wprowadzić następującą komendę MQSC:

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

Jeśli atrybut BackoutThreshold jest ustawiony na wartość inną niż zero, to aby uniknąć nieoczekiwanego działania, należy ustawić atrybut QName BackoutRequeuena poprawną nazwę kolejki.

W przypadku przesyłania komunikatów w trybie publikowania/subskrypcji, jeśli system tworzy kolejkę dynamiczną dla każdej subskrypcji, te wartości atrybutów są uzyskiwane z kolejki modelowej IBM WebSphere MQ classes for JMS (SYSTEM.JMS.MODEL.QUEUE. Aby zmienić te ustawienia, należy użyć następującej komendy:

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

Jeśli wartość progowa wycofania jest równa zero, obsługa komunikatów nieprzetwarzalnych jest wyłączona, a komunikaty nieprzetwarzalne pozostają w kolejce wejściowej. W przeciwnym razie, gdy liczba wycofań osiągnie wartość progową, komunikat jest wysyłany do nazwanej kolejki wycofanych komunikatów. Jeśli licznik wycofań osiągnie wartość progową, ale komunikat nie może przejść do kolejki wycofania, komunikat jest wysyłany do kolejki niedostarczonych komunikatów lub jest usuwany. Taka sytuacja występuje wtedy, gdy kolejka wycofania nie została zdefiniowana lub jeśli obiekt MessageConsumer nie może wysłać komunikatu do kolejki wycofania. Więcej informacji na ten temat zawiera sekcja “Usuwanie komunikatów z kolejki w ASF” na stronie 955.

W przypadku, gdy komunikat jest ponownie wysyłany do kolejki wycofanych komunikatów, niektóre wartości pól w deskryptorze komunikatu (MQMD) zmiany komunikatu. Szczegółowe informacje na temat formatu deskryptora MQMD zawiera sekcja MQMD-deskryptor komunikatu .

Następujące pola MQMD zmieniają wartość, gdy komunikat przechodzi do kolejki wycofanych komunikatów.

- Wartość PutDate jest aktualizowana do daty, w której przechodzi do kolejki wycofanych komunikatów.
- Wartość PutTime jest aktualizowana do czasu, w którym przechodzi do kolejki wycofanych komunikatów.
- Liczba wycofań jest resetowana do zera.
- Utrata ważności komunikatu jest aktualizowana w taki sposób, aby odzwierciedlała pozostały czas utraty ważności w momencie odebrania oryginalnego komunikatu przez aplikację JMS.

Wartości w poniższych polach pozostają takie same, gdy komunikat jest wyświetlany w kolejce wycofania:

- StructId
- Wersja
- Raport
- MessageType
- Opinie
- Kodowanie
- CodedCharSetId
- MsgId
- CorrelId
- ReplyToQ
- ReplyToQMgr
- Format
- Trwałość
- Priorytet

Wyjątki w produkcie IBM WebSphere MQ classes for JMS

Aplikacja IBM WebSphere MQ classes for JMS musi obsługiwać wyjątki zgłaszane przez wywołania interfejsu API JMS lub dostarczane do procedury obsługi wyjątków.

Produkt IBM WebSphere MQ classes for JMS zgłasza problemy w czasie wykonywania, zgłaszając wyjątki. Wyjątek JMSException jest klasą główną dla wyjątków zgłaszanych przez metody JMS, a wychwytywanie wyjątków JMSException udostępnia ogólny sposób obsługi wszystkich wyjątków związanych z JMS.

Każdy wyjątek JMSException hermetyzuje następujące informacje:

- Komunikat o wyjątku specyficznym dla dostawcy, który jest wyświetlany przez aplikację wywołując metodę Throwable.getMessage().
- Kod błędu specyficznego dla dostawcy, który jest pobierany przez wywołanie metody JMSException.getErrorCode().
- Powiązany wyjątek. Wyjątek zgłoszony przez wywołanie interfejsu API JMS jest często wynikiem problemu niższego poziomu, który jest zgłaszany przez inny wyjątek powiązany z tym wyjątkiem. Aplikacja uzyskuje powiązany wyjątek, wywołując metodę JMSException.getLinkedException() lub metodę Throwable.getCause().

Większość wyjątków zgłaszanych przez produkt IBM WebSphere MQ classes for JMS to instancje podklas wyjątku JMSException. Te podklasy implementują interfejs com.ibm.msg.client.jms.JmsExceptionDetail , który udostępnia następujące dodatkowe informacje:

- Wyjaśnienie komunikatu o wyjątku, którego dotyczy aplikacja, wywołując metodę JmsExceptionDetail.getExplanation().
- Zalecana odpowiedź użytkownika na wyjątek, który aplikacja uzyskuje, wywołując metodę JmsExceptionDetail.getUserAction().
- Klucze dla operacji wstawiania komunikatu w komunikacie o wyjątku. Aplikacja uzyskuje iterator dla wszystkich kluczy, wywołując metodę JmsExceptionDetail.getKeys().
- Komunikat zostanie wstawiony w komunikacie o wyjątku. Na przykład wstawienie komunikatu może być nazwą kolejki, która spowodowała wyjątek, i może być przydatna dla aplikacji, która ma mieć dostęp do tej nazwy. Aplikacja uzyskuje wstawiany komunikat odpowiadający określonowi kluczowi, wywołując metodę JmsExceptionDetail.getValue().

Wszystkie metody w interfejsie szczegółów JmsExceptionmogą zwracać wartość NULL, jeśli nie są dostępne żadne szczegóły.

Jeśli na przykład aplikacja próbuje utworzyć producenta komunikatów dla kolejki IBM WebSphere MQ , która nie istnieje, zgłaszany jest wyjątek z następującymi informacjami:

```
Message : JMSWMQ2008: Failed to open MQ queue 'Q_test'.
Class : class com.ibm.msg.client.jms.DetailedInvalidDestinationException
Error Code : JMSWMQ2008
Explanation : JMS attempted to perform an MQOPEN, but WebSphere MQ reported an
              error.
User Action : Use the linked exception to determine the cause of this error. Check
              that the specified queue and queue manager are defined correctly.
```

Zgłoszony wyjątek, com.ibm.msg.client.jms.DetailedInvalidDestinationException, jest podklasą klasy javax.jms.InvalidDestinationException i implementuje interfejs com.ibm.msg.client.jms.JmsExceptionDetail .

Powiązane wyjątki

Dołączony wyjątek zawiera dodatkowe informacje na temat problemu z środowiskiem wykonawczym. Dlatego dla każdego zgłoszonego wyjątku JMSException aplikacja powinna sprawdzić powiązany wyjątek. Sam powiązany wyjątek może mieć inny powiązany wyjątek, a więc połączone wyjątki tworzą łańcuch prowadzący do pierwotnego problemu bazowego. Powiązany wyjątek jest implementowany przy użyciu mechanizmu połączonych wyjątków klasy java.lang.Throwable , a aplikacja uzyskuje powiązany wyjątek, wywołując metodę Throwable.getCause(). W przypadku wyjątku JMSException metoda getLinkedException () w rzeczywistości deleguje do metody Throwable.getCause().

Na przykład, jeśli aplikacja określa niepoprawny numer portu podczas nawiązywania połączenia z menedżerem kolejek, wyjątki tworzą następujący łańcuch:

```
com.ibm.msg.client.jms.DetailIllegalStateException
   |
   +--->com.ibm.mq.MQException
           |
           +--->com.ibm.mq.jmqi.JmqiException
                   |
                   +--->java.net.ConnectionException
```

Zwykle każdy wyjątek w łańcuchu jest zgłaszany z innej warstwy w kodzie. Na przykład wyjątki w poprzedzającym łańcuchu są zgłaszane przez następujące warstwy:

- Pierwszy wyjątek, instancja podklasy wyjątku JMSException, jest zgłaszany przez wspólną warstwę w produkcie IBM WebSphere MQ classes for JMS.
- Następny wyjątek, instancja klasy com.ibm.mq.MQException, jest zgłaszany przez dostawcę przesyłania komunikatów produktu IBM WebSphere MQ .
- Następny wyjątek, instancja klasy com.ibm.mq.jmqi.JmqiException, jest zgłaszany przez wspólny interfejs Java do interfejsu MQI.
- Końcowy wyjątek, instancja klasy java.net.ConnectionException, jest zgłaszany przez bibliotekę klas Java.

Więcej informacji na temat architektury warstwowej produktu IBM WebSphere MQ classes for JMSzawiera sekcja “Klasy produktu IBM WebSphere MQ dla architektury JMS” na stronie 820.

Używając kodu podobnego do następującego kodu, aplikacja może iterować przez ten łańcuch w celu wyodrębnienia wszystkich odpowiednich informacji:

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import javax.jms.JMSException;
.
.
.
catch (JMSException je) {
    System.err.println("Caught JMSException");

    // Check for linked exceptions in JMSException
    Throwable t = je;
    while (t != null) {
```

```
        // Write out the message that is applicable to all exceptions
        System.err.println("Exception Msg: " + t.getMessage());
        // Write out the exception stack trace
        t.printStackTrace(System.err);

        // Add on specific information depending on the type of exception
        if (t instanceof JMSException) {
            JMSException je1 = (JMSException) t;
            System.err.println("JMS Error code: " + je1.getErrorCode());

            if (t instanceof JmsExceptionDetail){
                JmsExceptionDetail jed = (JmsExceptionDetail)je1;
                System.err.println("JMS Explanation: " + jed.getExplanation());
                System.err.println("JMS Explanation: " + jed.getUserAction());
            }
        } else if (t instanceof MQException) {
            MQException mqe = (MQException) t;
            System.err.println("WMQ Completion code: " + mqe.getCompCode());
            System.err.println("WMQ Reason code: " + mqe.getReason());
        } else if (t instanceof JmqiException){
            JmqiException jmqie = (JmqiException)t;
            System.err.println("WMQ Log Message: " + jmqie.getWmqLogMessage());
            System.err.println("WMQ Explanation: " + jmqie.getWmqMsgExplanation());
            System.err.println("WMQ Msg Summary: " + jmqie.getWmqMsgSummary());
            System.err.println("WMQ Msg User Response: "
                               + jmqie.getWmqMsgUserResponse());
            System.err.println("WMQ Msg Severity: " + jmqie.getWmqMsgSeverity());
        }

        // Get the next cause
        t = t.getCause();
    }
}
```

Należy pamiętać, że aplikacja powinna zawsze sprawdzać typ każdego wyjątku w łańcuchu, ponieważ typ wyjątku może się różnić, a wyjątki różnych typów hermetyzować różne informacje.

Uzyskiwanie informacji specyficznych dla produktu IBM WebSphere MQ dotyczących problemu

Instancje com.ibm.mq.MQException i com.ibm.mq.jmqi.JmqiException hermetyzują IBM WebSphere MQ konkretne informacje o problemie.

Wyjątek MQException hermetyzuje następujące informacje:

- Kod zakończenia, który aplikacja uzyskuje, wywołując metodę getCompCode ().
- Kod przyczyny, który aplikacja uzyskuje przez wywołanie metody getReason().

Wyjątek JmqiException hermetyzuje kod zakończenia i kod przyczyny. Dodatkowo jednak wyjątek JmqiException hermetykuje informacje w komunikacie AMQ*nnnn* lub CSQ*nnnn* , jeśli jest on powiązany z tym wyjątkiem. Wywołując odpowiednie metody wyjątku, aplikacja może uzyskać różne komponenty tego komunikatu, takie jak istotność, wyjaśnienie i odpowiedź użytkownika.

Przykłady użycia metod wymienionych w tej sekcji można znaleźć w przykładowym kodzie w produkcie “Powiązane wyjątki” na stronie 917.

Aktualizacja z poprzednich wersji produktu IBM WebSphere MQ classes for JMS

W porównaniu z poprzednimi wersjami produktu IBM WebSphere MQ classes for JMSwiększość kodów błędów i komunikatów o wyjątkach została zmieniona w wersji 7. Przyczyną tych zmian jest to, że produkt IBM WebSphere MQ classes for JMS ma teraz architekturę warstwową, a wyjątki są zgłaszane z różnych warstw w kodzie.

Jeśli na przykład aplikacja próbuje połączyć się z menedżerem kolejek, który nie istnieje, poprzednia wersja produktu IBM WebSphere MQ classes for JMS zgłosiła wyjątek JMSException z następującymi informacjami:

```
MQJMS2005: Failed to create MQQueueManager for 'localhost:QM_test'.
```

Ten wyjątek zawierał powiązany wyjątek MQException z następującymi informacjami:

```
MQJE001: Completion Code 2, Reason 2058
```

W porównaniu w tych samych okolicznościach w wersji 7 produktu IBM WebSphere MQ classes for JMS zgłaszany jest wyjątek JMSException z następującymi informacjami:

```
Message : JMSWMQ0018: Failed to connect to queue manager 'QM_test' with
          connection mode 'Client' and host name 'localhost'.
Class : class com.ibm.msg.client.jms.DetailedJMSException
Error Code : JMSWMQ0018
Explanation : null
User Action : Check the queue manager is started and if running in client mode,
              check there is a listener running. Please see the linked exception
              for more information.
```

Ten wyjątek zawiera powiązany wyjątek MQException z następującymi informacjami:

```
Message : JMSCMQ0001: WebSphere MQ call failed with compcode '2' ('MQCC_FAILED')
          reason '2058' ('MQRC_Q_MGR_NAME_ERROR').
Class : class com.ibm.mq.MQException
Completion Code : 2
Reason Code : 2058
```

Jeśli aplikacja analizuje lub testuje komunikaty wyjątków zwracane przez metodę Throwable.getMessage() lub kody błędów zwracane przez metodę JMSException.getErrorCode(), a wykonywana jest aktualizacja z wersji wcześniejszej niż wersja 7, aplikacja prawdopodobnie musi zostać zmodyfikowana w celu użycia wersji 7 produktu IBM WebSphere MQ classes for JMS.

Obiekty nasłuchiwania wyjątków

Aplikacja może zarejestrować obiekt nasłuchiwania wyjątków w obiekcie połączenia. Następnie, jeśli wystąpi problem, który sprawia, że połączenie nie będzie możliwe do użycia, produkt IBM WebSphere MQ classes for JMS dostarczy wyjątek do obiektu nasłuchiwania wyjątków, wywołując metodę onException(). Następnie aplikacja ma możliwość ponownego nawiązania połączenia.

V 7.5.0.8 APAR IT14820, dołączony z produktu IBM WebSphere MQ Version 7.5.0, pakiet poprawek 8, naprawiony defekt, w przypadku którego nie wywołano funkcji JMS ExceptionListener aplikacji dla wyjątków niepowiązanych z połączeniem (na przykład MQRC_GET_INHIBITED), nawet jeśli właściwość ASYNC_EXCEPTIONS fabryki połączeń JMS używana przez aplikację została ustawiona na ASYNC_EXCEPTIONS_ALL. Była to wartość domyślna wcześniejsza niż Version 7.5.0, Fix Pack 8.

V 7.5.0.8 Aby zachować zachowanie dla bieżących aplikacji JMS, które konfigurują obiekt JMS MessageListener i interfejs JMS ExceptionListener, a także aby upewnić się, że produkt IBM WebSphere MQ classes for JMS jest spójny ze specyfikacją JMS, wartość domyślna właściwości ASYNC_EXCEPTIONS JMS ConnectionFactory została zmieniona na ASYNC_EXCEPTIONS_CONNECTIONBROKEN dla IBM WebSphere MQ classes for JMS. w rezultacie domyślnie tylko wyjątki odpowiadające kodom błędu zerwanego połączenia są dostarczane do obiektu ExceptionListener JMS aplikacji.

V 7.5.0.8 W produkcie Version 7.5.0, Fix Pack 8aktualizacja produktu IBM WebSphere MQ classes for JMS została również zaktualizowana w taki sposób, że wyjątki JMSExceptions dotyczące błędów niezwiązanych z połączeniem, które występują podczas dostarczania komunikatów do asynchronicznych konsumentów komunikatów, są nadal dostarczane do zarejestrowanego obiektu ExceptionListener , gdy właściwość ConnectionFactory JMS używana przez aplikację ma właściwość ASYNC_EXCEPTIONS ustawioną na wartość ASYNC_EXCEPTIONS_ALL.

V 7.5.0.8 Więcej informacji na temat tego, co zostało zmienione dla programów nasłuchujących wyjątków dla produktu Version 7.5.0, Fix Pack 8 i dlaczego zmiany zostały wprowadzone z wcześniejszych wersji, zawiera sekcja JMS: zmiany obiektu nasłuchiwania wyjątków w wersji 7.5.

W przypadku każdego innego typu problemu wyjątek JMSException jest zgłaszany przez bieżące wywołanie interfejsu API JMS.

Jeśli aplikacja nie zarejestrują obiektu nasłuchiwania wyjątków z obiektem połączenia, wszystkie wyjątki, które zostałyby dostarczone do obiektu nasłuchiwania wyjątków, są zapisywane w dzienniku produktu IBM WebSphere MQ classes for JMS .

Odsyłacze pokrewne

WYJĄTEK ASYNCEXCEPTION

Rejestrowanie błędów w klasach produktu WebSphere MQ dla usługi JMS

Informacje o problemach środowiska wykonawczego, które mogą wymagać wykonania czynności naprawczych przez użytkownika, są zapisywane w klasach WebSphere MQ dla dziennika JMS.

Jeśli na przykład aplikacja próbuje ustawić właściwość fabryki połączeń, ale nazwa tej właściwości nie została rozpoznana, klasy WebSphere MQ classes for JMS zapisują informacje o problemie do jego dziennika.

Domyślnie plik zawierający dziennik ma nazwę mqjms.log i znajduje się w bieżącym katalogu roboczym. Można jednak zmienić nazwę i położenie pliku dziennika, ustawiając wartość właściwości com.ibm.msg.client.commonservices.log.outputName w klasach WebSphere MQ dla pliku konfiguracyjnego JMS. Więcej informacji na temat klas WebSphere MQ dla pliku konfiguracyjnego JMS zawiera sekcja “Plik konfiguracyjny IBM WebSphere MQ classes for JMS” na stronie 745, a w celu uzyskania bardziej szczegółowych informacji na temat poprawnych wartości właściwości com.ibm.msg.client.commonservices.log.outputName można znaleźć w sekcji “Rejestrowanie i IBM WebSphere MQ classes for JMS” na stronie 817.

Technologia obsługi pierwszego niepowodzenia (FFST) w klasach produktu WebSphere MQ dla usługi JMS

Jeśli w ramach klas WebSphere MQ classes for JMS wystąpi poważny błąd wewnętrzny, generowane są informacje o pierwszej awarii technologii obsługi (FFST).

Informacje FFST są zapisywane w pliku o nazwie JMSC*nnnn*.FDC, gdzie *nnnn* to czterocyfrowy numer. Ten plik znajduje się w katalogu o nazwie FFDC, który jest podkatalogiem katalogu, w którym zapisywane są dane wyjściowe śledzenia. Domyślnie dane wyjściowe śledzenia są zapisywane w bieżącym katalogu roboczym, ale można przekierować dane wyjściowe śledzenia do innego katalogu, ustawiając właściwość **com.ibm.msg.client.commonservices.trace.outputName** w klasach produktu WebSphere MQ dla pliku konfiguracyjnego JMS. Więcej informacji na temat klas produktu WebSphere MQ dla pliku konfiguracyjnego JMS zawiera sekcja “Plik konfiguracyjny IBM WebSphere MQ classes for JMS” na stronie 745.

Jeśli śledzenie jest włączone, gdy generowane są informacje FFST , informacje FFST są również zapisywane w pliku śledzenia. Więcej informacji na temat śledzenia programów JMS zawiera sekcja Śledzenie aplikacji IBM WebSphere MQ classes for JMS.

Aby wyłączyć produkcję plików FFDC, należy ustawić właściwość **com.ibm.msg.client.commonservices.ffst.suppress**w następujący sposób:

0
: Wyprowadza wszystkie pliki FFDC (domyślnie).

-1
: Dane wyjściowe tylko w przypadku pierwszych plików FFDC określonego typu.

liczba całkowita
: Pomijaj wszystkie pliki FFDC, z wyjątkiem tych, które są wielokrotnością tej liczby.

Uzyskiwanie dostępu do funkcji produktu WebSphere MQ z klas produktu WebSphere MQ dla aplikacji JMS

Klasy WebSphere MQ classes for JMS udostępniają narzędzia do korzystania z wielu funkcji produktu WebSphere MQ.

Ostrzeżenie: Te funkcje są poza specyfikacją JMS lub, w niektórych przypadkach, naruszają specyfikację JMS. W przypadku korzystania z nich aplikacja jest mało prawdopodobna, aby była

kompatybilna z innymi dostawcami JMS. Te funkcje, które nie są zgodne ze specyfikacją JMS, są oznaczane za pomocą powiadomienia alarmowego.

Odczytywanie i zapisywanie deskryptora komunikatu z klas produktu WebSphere MQ dla aplikacji JMS

Użytkownik może sterować możliwością uzyskiwania dostępu do deskryptora komunikatu (MQMD), ustawiając właściwości w miejscu docelowym i komunikacie.

Niektóre aplikacje produktu WebSphere MQ wymagają, aby określone wartości zostały ustawione w strukturze MQMD komunikatów wysyłanych do tych aplikacji. Klasy WebSphere MQ classes for JMS udostępniają atrybuty komunikatów, które umożliwiają aplikacjom JMS ustawianie pól MQMD, a więc umożliwiają aplikacjom JMS działanie aplikacji WebSphere MQ na dysku.

Wartość właściwości obiektu docelowego WMQ_MQMD_WRITE_ENABLED należy ustawić na wartość true, aby ustawienie właściwości MQMD miało jakikolwiek wpływ. Następnie można użyć metod ustawiania właściwości komunikatu (na przykład setStringProperty) w celu przypisania wartości do pól MQMD. Wszystkie pola MQMD są prezentowane z wyjątkiem pól StrucId i Version; BackoutCount można odczytać, ale nie zapisywać do.

Ten przykład powoduje, że komunikat jest umieszczany w kolejce lub w temacie MQMD.UserIdentifier jest ustawiony na wartość "JoeBloggs".

```
// Create a ConnectionFactory, connection, session, producer, message
// ...

// Create a destination
// ...

// Enable MQMD write
dest.setBooleanProperty(WMQConstants.WMQ_MQMD_WRITE_ENABLED, true);

// Optionally, set a message context if applicable for this MD field
dest.setIntProperty(WMQConstants.WMQ_MQMD_MESSAGE_CONTEXT,
  WMQConstants.WMQ_MDCTX_SET_IDENTITY_CONTEXT);

// On the message, set property to provide custom UserId
msg.setStringProperty("JMS_IBM_MQMD_UserIdentifier", "JoeBloggs");

// Send the message
// ...
```

Przed ustawieniem JMS_IBM_MQMD_UserIdentifiernależy ustawić wartość WMQ_MQMD_MESSAGE_CONTEXT. Więcej informacji na temat korzystania z właściwości WMQ_MQMD_MESSAGE_CONTEXT zawiera sekcja “Właściwości obiektu komunikatu JMS” na stronie 924.

Podobnie można wyodrębnić zawartość pól MQMD, ustawiając wartość WMQ_MQMD_READ_ENABLED na wartość true przed odebraniem komunikatu, a następnie za pomocą metod pobierania komunikatu, na przykład właściwości getString. Wszystkie otrzymane właściwości są przeznaczone tylko do odczytu.

Ten przykład powoduje, że w polu *wartość* znajduje się wartość MQMD.ApplIdentityData : komunikat został zwrócony z kolejki lub tematu.

```
// Create a ConnectionFactory, connection, session, consumer
// ...

// Create a destination
// ...

// Enable MQMD read
dest.setBooleanProperty(WMQConstants.WMQ_MQMD_READ_ENABLED, true);

// Receive a message
// ...

// Get MQMD field value using a property
String value = rcvMsg.getStringProperty("JMS_IBM_MQMD_ApplIdentityData");
```

Właściwości obiektu docelowego JMS
Dwie właściwości obiektu docelowego kontrolują dostęp do deskryptora MQMD z usługi JMS, a trzeci kontekst komunikatu steruje.

Tabela 124. Nazwy i opisy właściwości

Właściwość	**Postać krótka**	**Opis**
WMQ_MQMD_WRITE_ENABLED	MDW	Określa, czy aplikacja JMS może ustawiać wartości pól MQMD.
WMQ_MQMD_READ_ENABLED	MDR	Określa, czy aplikacja JMS może wyodrębnić wartości pól MQMD.
WMQ_MQMD_MESSAGE_ CONTEXT	MDCTX	Jaki poziom kontekstu komunikatu ma zostać ustawiony przez aplikację JMS. Aby ta właściwość była uwzględniana, aplikacja musi być uruchomiona z odpowiednim uprawnieniem kontekstu.

Tabela 125. Nazwy właściwości, wartości i metody ustawiania

Właściwość	**Poprawne wartości w narzędziu administracyjnym (wartości domyślne są pogrubione)**	**Poprawne wartości w programach**	**Ustaw metodę**
WMQ_MQMD_WRITE (_ENABLED)	• **Nie** Wszystkie właściwości JMS_IBM_MQMD* są ignorowane, a ich wartości nie są kopiowane do bazowej struktury MQMD. • YES Przetwarzane są właściwości JMS_IBM_MQMD*. Ich wartości są kopiowane do bazowej struktury MQMD.	• **Fałsz** • Prawda	setMQMDWriteWłączone

Tabela 125. Nazwy właściwości, wartości i metody ustawiania (kontynuacja)			
Właściwość	**Poprawne wartości w narzędziu administracyjnym (wartości domyślne są pogrubione)**	**Poprawne wartości w programach**	**Ustaw metodę**
WMQ_MQMD_READ__ENABLED	• **Nie** Podczas wysyłania komunikatów właściwości JMS_IBM_MQMD* wysłanego komunikatu nie są aktualizowane w celu odzwierciedlenia zaktualizowanych wartości pól w strukturze MQMD. Podczas odbierania komunikatów żadna właściwość JMS_IBM_MQMD* nie jest dostępna w odebranym komunikacie, nawet jeśli nadawca ustawił niektóre lub wszystkie z nich. • YES Podczas wysyłania komunikatów wszystkie właściwości JMS_IBM_MQMD* wysłanego komunikatu są aktualizowane w taki sposób, aby odzwierciedlały zaktualizowane wartości pól w strukturze MQMD, w tym te, które nie zostały jawnie ustawione przez nadawcę. Podczas odbierania komunikatów wszystkie właściwości JMS_IBM_MQMD* są dostępne w odebranym komunikacie (w tym także te, które nie zostały jawnie ustawione przez nadawcę).	• **Fałsz** • Prawda	setMQMDReadWłączone

Tabela 125. Nazwy właściwości, wartości i metody ustawiania (kontynuacja)			
Właściwość	**Poprawne wartości w narzędziu administracyjnym (wartości domyślne są pogrubione)**	**Poprawne wartości w programach**	**Ustaw metodę**
WMQ_MQMD _MESSAGE_CONTEXT	• **Domyślnie** Wywołanie funkcji API MQOPEN i struktura MQPMO nie określają jawnych opcji kontekstu komunikatu. • SET_IDENTITY_CONTEXT, Wywołanie funkcji API MQOPEN określa opcję kontekstu komunikatu MQOO_SET_IDENTITY_CONTEXT, a struktura MQPMO określa wartość MQPMO_SET_IDENTITY_CONTEXT. • SET_ALL_CONTEXT Wywołanie funkcji API MQOPEN określa opcję kontekstu komunikatu MQOO_SET_ALL_CONTEXT, a struktura MQPMO określa MQPMO_SET_ALL_CONTEXT.	• **WMQ_MDCTX_DEFAULT** • WMQ_MDCTX_SET_IDENTITY_CONTEXT • WMQ_MDCTX_SET_ALL_CONTEXT	Kontekst setMQMDMessage

Właściwości obiektu komunikatu JMS
Właściwości obiektu komunikatu z przedrostkiem JMS_IBM_MQMD pozwalają na ustawienie lub odczytanie odpowiedniego pola MQMD.

Wysyłanie komunikatów

Wszystkie pola MQMD z wyjątkiem StrucId i Version są reprezentowane. Te właściwości odnoszą się tylko do pól MQMD. W przypadku, gdy właściwość występuje zarówno w strukturze MQMD, jak i w nagłówku MQRFH2 , wersja w tabeli MQRFH2 nie jest ustawiona lub wyodrębniana.

Możliwe jest ustawienie dowolnej z tych właściwości, z wyjątkiem właściwości JMS_IBM_MQMD_BackoutCount. Dowolna wartość ustawiona dla JMS_IBM_MQMD_BackoutCount jest ignorowana.

Jeśli właściwość ma maksymalną długość, a użytkownik poda wartość, która jest zbyt długa, wartość jest obcinana.

W przypadku niektórych właściwości należy również ustawić właściwość WMQ_MQMD_MESSAGE_CONTEXT w obiekcie docelowym. Aby ta właściwość miała zastosowanie, aplikacja musi być uruchamiana z odpowiednimi uprawnieniami dotyczącymi kontekstu. Jeśli wartość właściwości WMQ_MQMD_MESSAGE_CONTEXT nie zostanie ustawiona na odpowiednią wartość, wartość właściwości zostanie zignorowana. Jeśli wartość WMQ_MQMD_MESSAGE_CONTEXT zostanie ustawiona na odpowiednią wartość, ale użytkownik nie ma wystarczających uprawnień kontekstowych dla menedżera kolejek, zostanie wygenerowany wyjątek JMSException. Właściwości wymagające konkretnych wartości właściwości WMQ_MQMD_MESSAGE_CONTEXT są następujące.

Następujące właściwości wymagają, aby wartość WMQ_MQMD_MESSAGE_CONTEXT została ustawiona na wartość WMQ_MDCTX_SET_IDENTITY_CONTEXT lub WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- Dane JMS_IBM_MQMD_ApplIdentity

Następujące właściwości wymagają, aby wartość WMQ_MQMD_MESSAGE_CONTEXT została ustawiona na wartość WMQ_MDCTX_SET_ALL_CONTEXT:

- Typ JMS_IBM_MQMD_PutAppl
- Nazwa JMS_IBM_MQMD_PutAppl
- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- Dane JMS_IBM_MQMD_ApplOrigin

Odbieranie komunikatów

Wszystkie te właściwości są dostępne w odebranym komunikacie, jeśli właściwość WMQ_MQMD_READ_ENABLED jest ustawiona na wartość true (prawda), bez względu na rzeczywiste właściwości, które zostały ustawione przez aplikację produkującą. Aplikacja nie może modyfikować właściwości odebranego komunikatu, chyba że wszystkie właściwości zostaną wyczyszczone jako pierwsze zgodnie ze specyfikacją JMS. Odebrany komunikat może zostać przesłany bez modyfikowania właściwości.

Ostrzeżenie: Jeśli aplikacja odbierze komunikat z miejsca docelowego z właściwością WMQ_MQMD_READ_ENABLED ustawioną na wartość true, a następnie przekazuje ją do miejsca docelowego z ustawioną wartością WMQ_MQMD_WRITE_ENABLED na wartość true, spowoduje to, że wszystkie wartości pól MQMD odebranego komunikatu są kopiowane do przekazanego komunikatu.

Tabela właściwości

W tej tabeli znajduje się lista właściwości obiektu komunikatu reprezentującego pola MQMD. Odsyłacze do pełnych opisów pól i ich dozwolonych wartości można znaleźć w sekcji odsyłaczy.

Tabela 126. Nazwy właściwości, opisy i typy

Właściwość	Opis	Typ Java	Odsyłacz do pełnego opisu
Raport JMS_IBM_MQMD_Report	Opcje dla komunikatów raportu	Liczba całkowita	Raport
JMS_IBM_MQMD_MsgType	Typ komunikatu	Liczba całkowita	MsgType
JMS_IBM_MQMD_Expiry	Czas życia komunikatu	Liczba całkowita	Utrata ważności
Sprzężenie zwrotne JMS_IBM_MQMD_Feedback	Informacja zwrotna lub kod przyczyny	Liczba całkowita	Opinie
Kodowanie JMS_IBM_MQMD_Encoding	Kodowanie numeryczne danych komunikatu	Liczba całkowita	Kodowanie
JMS_IBM_MQMD_CodedCharSetId	Identyfikator zestawu znaków danych komunikatu	Liczba całkowita	CodedCharSetId
Format JMS_IBM_MQMD_Format	Nazwa formatu danych komunikatu	Łańcuch	Formatowanie
JMS_IBM_MQMD_Priority [1]	Priorytet komunikatu	Liczba całkowita	Priorytet
Trwałość JMS_IBM_MQMD_Persistence	Trwałość komunikatu	Liczba całkowita	Trwałość
JMS_IBM_MQMD_MsgId [2]	Identyfikator komunikatu	Obiekt (byte []) [4]	MsgId

Tabela 126. Nazwy właściwości, opisy i typy (kontynuacja)

Właściwość	Opis	Typ Java	Odsyłacz do pełnego opisu
JMS_IBM_MQMD_CorrelId [3]	Identyfikator korelacji	Obiekt (byte []) [4]	CorrelId
JMS_IBM_MQMD_BackoutCount	Liczba wycofanych	Liczba całkowita	BackoutCount
JMS_IBM_MQMD_ReplyToQ	Nazwa kolejki odpowiedzi	Łańcuch	Kolejka_zwrotna
Menedżer kolejek JMS_IBM_MQMD_ReplyTo	Nazwa menedżera kolejek odpowiedzi	Łańcuch	ReplyToQMgr
JMS_IBM_MQMD_UserIdentifier	Identyfikator użytkownika	Łańcuch	UserIdentifier
JMS_IBM_MQMD_AccountingToken	Token rozliczania	Obiekt (byte []) [4]	AccountingToken
Dane JMS_IBM_MQMD_ApplIdentity	Dane aplikacji odnoszące się do tożsamości	Łańcuch	Dane_tożsamości_aplikacji
Typ JMS_IBM_MQMD_PutAppl	Typ aplikacji, która wstawiła komunikat	Liczba całkowita	Typ_aplikacji_wstawiającej
Nazwa JMS_IBM_MQMD_PutAppl	Nazwa aplikacji umieszczonej w komunikacie.	Łańcuch	Nazwa_aplikacji_wstawiającej
JMS_IBM_MQMD_PutDate	Data umieszczenia komunikatu	Łańcuch	PutDate
JMS_IBM_MQMD_PutTime	Czas umieszczenia komunikatu	Łańcuch	PutTime
Dane JMS_IBM_MQMD_ApplOrigin	Dane dotyczące wniosku dotyczące pochodzenia	Łańcuch	Dane_pochodzenia_aplikacji
JMS_IBM_MQMD_GroupId	Identyfikator grupy	Obiekt (byte []) [4]	GroupId
Numer JMS_IBM_MQMD_MsgSeq	Numer kolejny komunikatu logicznego w grupie	Liczba całkowita	Numer_kolejny_komunikatu
Przesunięcie JMS_IBM_MQMD_Offset	Przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego	Liczba całkowita	Depozycja
JMS_IBM_MQMD_MsgFlags	Flagi komunikatu	Liczba całkowita	MsgFlags
JMS_IBM_MQMD_OriginalLength	Długość oryginalnego komunikatu	Liczba całkowita	OriginalLength

Tabela 126. Nazwy właściwości, opisy i typy (kontynuacja)

Właściwość	Opis	Typ Java	Odsyłacz do pełnego opisu

1. **Ostrzeżenie:** Jeśli wartość zostanie przypisana do wartości JMS_IBM_MQMD_Priority, która nie mieści się w zakresie od 0 do 9, to narusza to specyfikację JMS.
2. **Ostrzeżenie:** Specyfikacja JMS określa, że identyfikator komunikatu musi być ustawiony przez dostawcę JMS i musi być unikalny lub ma wartość NULL. Jeśli zostanie przypisana wartość JMS_IBM_MQMD_MsgId, ta wartość zostanie skopiowana do wartości JMSMessageID. Dlatego nie jest on ustawiany przez dostawcę JMS i może nie być unikalny: narusza to specyfikację JMS.
3. **Ostrzeżenie:** Jeśli zostanie przypisana wartość JMS_IBM_MQMD_CorrelId , która rozpoczyna się od łańcucha 'ID:', to narusza to specyfikację JMS.
4. **Ostrzeżenie:** Użycie właściwości tablicy bajtów w komunikacie narusza specyfikację JMS.

Uzyskiwanie dostępu do danych komunikatu produktu IBM WebSphere MQ z aplikacji przy użyciu klas produktu WebSphere MQ dla usługi JMS

Istnieje możliwość uzyskania dostępu do pełnych danych komunikatów produktu WebSphere MQ w obrębie aplikacji przy użyciu klas IBM WebSphere MQ dla usługi JMS. Aby uzyskać dostęp do wszystkich danych, komunikat musi być JMSBytesMessage. Treść JMSBytesMessage zawiera dowolny nagłówek MQRFH2 , wszystkie inne nagłówki IBM WebSphere MQ oraz następujące dane komunikatu.

Ustaw właściwość WMQ_MESSAGE_BODY miejsca docelowego na wartość WMQ_MESSAGE_BODY_MQ, aby odebrać wszystkie dane treści komunikatu w produkcie JMSBytesMessage.

Jeśli parametr WMQ_MESSAGE_BODY jest ustawiony na wartość WMQ_MESSAGE_BODY_JMS lub WMQ_MESSAGE_BODY_UNSPECIFIED, treść komunikatu jest zwracana bez nagłówka MQRFH2 JMS, a właściwości składnika JMSBytesMessage odzwierciedlają właściwości ustawione w RFH2.

Niektóre aplikacje nie mogą korzystać z funkcji opisanych w tym temacie. Jeśli aplikacja jest połączona z menedżerem kolejek produktu WebSphere MQ V6 lub jeśli dla aplikacji została ustawiona wartość PROVIDERVERSION na 6, funkcje te nie są dostępne.

Wysyłanie wiadomości

Podczas wysyłania komunikatów właściwość miejsca docelowego (WMQ_MESSAGE_BODY) ma pierwszeństwo przed wartością WMQ_TARGET_CLIENT.

Jeśli parametr WMQ_MESSAGE_BODY jest ustawiony na wartość WMQ_MESSAGE_BODY_JMS, klasy produktu WebSphere MQ dla usługi JMS automatycznie generują nagłówek MQRFH2 na podstawie ustawień właściwości i pól nagłówka produktu JMSMessage .

Jeśli parametr WMQ_MESSAGE_BODY jest ustawiony na wartość WMQ_MESSAGE_BODY_MQ, do treści komunikatu nie jest dodawany żaden dodatkowy nagłówek.

Jeśli parametr WMQ_MESSAGE_BODY jest ustawiony na wartość WMQ_MESSAGE_BODY_UNSPECIFIED, klasy WebSphere MQ classes for JMS wysyłają nagłówek MQRFH2 , chyba że wartość WMQ_TARGET_CLIENT jest ustawiona na wartość WMQ_TARGET_DEST_MQ. Po odebraniu, ustawienie WMQ_TARGET_CLIENT na wartość WMQ_TARGET_DEST_MQ spowoduje usunięcie z treści komunikatu żadnych MQRFH2 .

Uwaga: Produkty JMSBytesMessage i JMSTextMessage nie wymagają MQRFH2, natomiast JMSStreamMessage, JMSMapMessagei JMSObjectMessage .

WMQ_MESSAGE_BODY_UNSPECIFIED jest ustawieniem domyślnym dla WMQ_MESSAGE_BODY, a WMQ_TARGET_DEST_JMS jest domyślnym ustawieniem WMQ_TARGET_CLIENT.

W przypadku wysłania JMSBytesMessagemożna przesłonić ustawienia domyślne dla treści komunikatu JMS po utworzeniu komunikatu WebSphere MQ . Użyj następujących właściwości:

- JMS_IBM_Format lub JMS_IBM_MQMD_Format: ta właściwość określa format nagłówka lub ładunku aplikacji produktu WebSphere MQ , który uruchamia treść komunikatu JMS, jeśli nie ma poprzedniego nagłówka produktu Websphere MQ .
- JMS_IBM_Character_Set lub JMS_IBM_MQMD_CodedCharSetId: ta właściwość określa CCSID ładunku nagłówka lub aplikacji produktu WebSphere MQ , który uruchamia treść komunikatu JMS, jeśli nie ma poprzedniego nagłówka produktu Websphere MQ .
- JMS_IBM_Encoding lub JMS_IBM_MQMD_Encoding: ta właściwość określa kodowanie ładunku nagłówka lub aplikacji produktu WebSphere MQ , które uruchamia treść komunikatu JMS, jeśli nie ma poprzedniego nagłówka produktu Websphere MQ .

Jeśli określono oba typy właściwości, właściwości JMS_IBM_MQMD_* przesłaniają odpowiednie właściwości produktu JMS_IBM_* , o ile właściwość docelowa WMQ_MQMD_WRITE_ENABLED jest ustawiona na wartość true.

Różnice w skutkach między ustawieniem właściwości komunikatu za pomocą JMS_IBM_MQMD_* i JMS_IBM_* są znaczące:

1. Właściwości produktu JMS_IBM_MQMD_* są specyficzne dla dostawcy JMS produktu IBM WebSphere MQ .
2. Właściwości JMS_IBM_MQMD_* są ustawiane tylko w MQMD. Właściwości produktu JMS_IBM_* są ustawiane w produkcie MQMD tylko wtedy, gdy komunikat nie ma nagłówka JMS produktu MQRFH2 . W przeciwnym razie są one ustawiane w nagłówku RFH2 JMS.
3. Właściwości JMS_IBM_MQMD_* nie mają wpływu na kodowanie tekstu i liczb zapisanych w JMSMessage.

   Aplikacja odbierający prawdopodobnie przyjmie wartości parametrów MQMD.Encoding i MQMD.CodedCharSetId odpowiadają kodowaniu i zestawem znaków liczb i tekstu w treści komunikatu. Jeśli używane są właściwości produktu JMS_IBM_MQMD_* , za pomocą aplikacji wysyłającej należy ją wykonać. Kodowanie i zestaw znaków liczb i tekstu w treści komunikatu są ustawiane za pomocą właściwości JMS_IBM_* .

   Źle zakodowany fragment kodu w produkcie Rysunek 163 na stronie 928 wysyła komunikat zakodowany w zestawie znaków 1208 z zestawem MQMD.CodedCharSetId ustawionym na wartość 37.

   a. Wyślij niesłusznie zakodowaną wiadomość

   ```
   TextMessage tmo = session.createTextMessage();
   ((MQDestination) destination).setMessageBodyStyle
                   (WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ((MQDestination)destination).setMQMDWriteEnabled(true);
   tmo.setIntProperty(WMQConstants.JMS_IBM_MQMD_CODEDCHARSETID, 37);
   tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 1208);
   tmo.setText("String one");
   producer.send(tmo);
   ```

   b. Odbierając komunikat, opierając się na wartości JMS_IBM_CHARACTER_SET ustawionej na podstawie wartości MQMD.CodedCharSetId:

   ```
   TextMessage tmi = (TextMessage) cons.receive();
   System.out.println("Message is \"" + tmi.getText() + "\"");
   ```

   c. Wynikowe dane wyjściowe:

   ```
   Message is "ëÈÊ'>...??>?"
   ```

   Rysunek 163. Niekonsekwentnie zakodowane dane MQMD i komunikaty

Jeden z fragmentów kodu w produkcie Rysunek 164 na stronie 929 powoduje umieszczenie komunikatu w kolejce lub temacie z jego treścią zawierającą ładunek aplikacji bez dodawania automatycznie wygenerowanego nagłówka MQRFH2 .

1. Ustawianie WMQ_MESSAGE_BODY_MQ:

```
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

2. Ustawianie WMQ_TARGET_DEST_MQ:

```
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);
((MQDestination) destination).
                setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
```

Rysunek 164. Wyślij komunikat z treścią komunikatu MQ .

Odbieranie komunikatu

Jeśli parametr WMQ_MESSAGE_BODY jest ustawiony na wartość WMQ_MESSAGE_BODY_JMS, typ i treść przychodzącego komunikatu JMS są określane na podstawie zawartości odebranego komunikatu produktu Websphere MQ . Typ i treść komunikatu są określane przez pola w nagłówku MQRFH2 lub w MQMD(jeśli nie ma MQRFH2).

Jeśli parametr WMQ_MESSAGE_BODY jest ustawiony na wartość WMQ_MESSAGE_BODY_MQ, przychodzącym typem komunikatu JMS jest JMSBytesMessage. Treść komunikatu JMS to dane komunikatu zwracane przez bazowe wywołanie funkcji API produktu MQGET . Długość treści komunikatu jest długością zwróconej przez wywołanie MQGET . Zestaw znaków i kodowanie danych w treści komunikatu jest określane za pomocą pól CodedCharSetId i Encoding w MQMD. Format danych w treści komunikatu jest określany na podstawie pola Format w MQMD .

Jeśli właściwość WMQ_MESSAGE_BODY jest ustawiona na wartość WMQ_MESSAGE_BODY_UNSPECIFIED, wartość domyślna IBM WebSphere MQ classes for JMS ustawia ją na wartość WMQ_MESSAGE_BODY_JMS.

Po odebraniu JMSBytesMessagemożna go zdekodować, odwołując się do następujących właściwości:

- JMS_IBM_Format lub JMS_IBM_MQMD_Format: ta właściwość określa format nagłówka lub ładunku aplikacji produktu WebSphere MQ , który uruchamia treść komunikatu JMS, jeśli nie ma poprzedniego nagłówka produktu Websphere MQ .
- JMS_IBM_Character_Set lub JMS_IBM_MQMD_CodedCharSetId: ta właściwość określa CCSID ładunku nagłówka lub aplikacji produktu WebSphere MQ , który uruchamia treść komunikatu JMS, jeśli nie ma poprzedniego nagłówka produktu Websphere MQ .
- JMS_IBM_Encoding lub JMS_IBM_MQMD_Encoding: ta właściwość określa kodowanie ładunku nagłówka lub aplikacji produktu WebSphere MQ , które uruchamia treść komunikatu JMS, jeśli nie ma poprzedniego nagłówka produktu Websphere MQ .

Poniższy fragment kodu powoduje odebraną wiadomość, która jest JMSBytesMessage. Niezależnie od treści odebranego komunikatu i pola formatu odebranego MQMD, komunikat jest JMSBytesMessage.

```
   ((MQDestination)destination).setMessageBodyStyle
                              (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

Właściwość miejsca docelowego WMQ_MESSAGE_BODY
Obiekt WMQ_MESSAGE_BODY określa, czy aplikacja JMS przetwarza obiekt MQRFH2 komunikatu WebSphere MQ jako część ładunku komunikatu (czyli jako część treści komunikatu JMS).

Tabela 127. Nazwy i opisy właściwości

Właściwość	Postać krótka	Opis
WMQ_MESSAGE_BODY	MBODY	Określa, czy aplikacja JMS przetwarza obiekt MQRFH2 komunikatu WebSphere MQ jako część ładunku komunikatu (to znaczy jako część treści komunikatu JMS).

Tabela 128. Nazwy właściwości, wartości i metody ustawiania

Właściwość	Poprawne wartości w narzędziu administracyjnym (wartości domyślne są pogrubione)	Poprawne wartości w programach	Ustaw metodę
WMQ_MESSAGE_BODY	• **Nieokreślone** Podczas wysyłania klasy WebSphere MQ classes for JMS generują lub nie generują i dołączają nagłówek MQRFH2 , w zależności od wartości właściwości WMQ_TARGET_CLIENT. Podczas odbierania działa jako wartość JMS. • JMS Podczas wysyłania, klasy WebSphere MQ classes for JMS automatycznie generują nagłówek MQRFH2 i dołączają go do komunikatu WebSphere MQ . Podczas odbierania klasy WebSphere MQ classes for JMS ustawiają właściwości komunikatu JMS zgodnie z wartościami w tabeli MQRFH2 (jeśli istnieje). Nie jest ona prezentowana jako część treści komunikatu JMS (MQRFH2). • MQ Podczas wysyłania klasy WebSphere MQ classes for JMS nie generują obiektu MQRFH2. Po odebraniu klasa WebSphere MQ classes for JMS przedstawia element MQRFH2 jako część treści komunikatu JMS.	• **WMQ_MESSAGE_ BODY_UNSPECIFIED** • WMQ_MESSAGE_BODY_JMS • WMQ_MESSAGE_BODY_MQ	setMessageBodyStyle

Trwałe komunikaty JMS

Klasy produktu WebSphere MQ dla aplikacji JMS mogą używać atrybutu kolejki produktu **NonPersistentMessageClass** w celu zapewnienia lepszej wydajności trwałych komunikatów JMS, kosztem niezawodności.

Kolejka produktu WebSphere MQ ma atrybut o nazwie **NonPersistentMessageClass**. Wartość tego atrybutu określa, czy nietrwałe komunikaty w kolejce są usuwane po restarcie menedżera kolejek.

Atrybut dla kolejki lokalnej można ustawić przy użyciu komendy WebSphere MQ Script (MQSC), DEFINE QLOCAL, z jednym z następujących parametrów:

NPMCLASS (NORMALNY)
Nietrwałe komunikaty w kolejce są usuwane po restarcie menedżera kolejek. Jest to wartość domyślna.

NPMCLASS (HIGH)
Nietrwałe komunikaty w kolejce nie są usuwane po zrestartowaniu menedżera kolejek po wygaszonym lub natychmiastowym zamknięciu. Komunikaty nietrwałe mogą jednak zostać usunięte po zawłaczającym zamknięciu systemu lub niepowodzeniu.

W tym temacie opisano, w jaki sposób klasy produktu WebSphere MQ dla aplikacji JMS mogą używać tego atrybutu kolejki w celu zapewnienia lepszej wydajności trwałych komunikatów JMS.

Właściwość PERSISTENCE dla obiektu kolejki lub tematu może mieć wartość HIGH. Aby ustawić tę wartość, można użyć narzędzia administracyjnego WebSphere MQ JMS, a aplikacja może wywołać metodę Destination.setPersistence(), przekazując wartość WMQConstants.WMQ_PER_NPHIGH jako parametr.

Jeśli aplikacja wysyła komunikat trwały JMS lub komunikat nietrwały JMS do miejsca docelowego, w którym właściwość PERSISTENCE ma wartość HIGH, a bazowa kolejka produktu WebSphere MQ jest ustawiona na NPMCLASS (HIGH), komunikat jest umieszczany w kolejce jako nietrwały komunikat WebSphere MQ . Jeśli właściwość PERSISTENCE dla miejsca docelowego nie ma wartości HIGH, lub jeśli kolejka bazowa jest ustawiona na NPMCLASS (NORMAL), komunikat trwały JMS jest umieszczany w kolejce jako trwały komunikat produktu WebSphere MQ , a nietrwały komunikat JMS jest umieszczany w kolejce jako nietrwały komunikat WebSphere MQ .

Jeśli trwały komunikat JMS jest umieszczany w kolejce jako nietrwały komunikat produktu WebSphere MQ , a użytkownik chce upewnić się, że komunikat nie zostanie odrzucony po wygaszonym lub natychmiastowym zamknięciu menedżera kolejek, wszystkie kolejki, przez które komunikat może być kierowany, muszą być ustawione na NPMCLASS (HIGH). W domenie publikowania/subskrybowania kolejki te obejmują kolejki subskrybenta. Aby wyegzekwować tę konfigurację, klasy WebSphere MQ classes for JMS zgłaszają wyjątek InvalidDestination, jeśli aplikacja próbuje utworzyć konsument komunikatów dla miejsca docelowego, w którym właściwość PERSISTENCE ma wartość HIGH, a bazowa kolejka WebSphere MQ jest ustawiona na NPMCLASS (NORMAL).

Ustawienie właściwości PERSISTENCE dla miejsca docelowego na wartość HIGH nie ma wpływu na sposób odbierania komunikatu z tego miejsca docelowego. Komunikat wysyłany jako trwały komunikat JMS jest odbierany jako trwały komunikat JMS, a komunikat wysyłany jako nietrwały komunikat JMS jest odbierany jako komunikat nietrwały JMS.

Gdy aplikacja wysyła pierwszy komunikat do miejsca docelowego, w którym właściwość PERSISTENCE ma wartość HIGH, lub gdy aplikacja tworzy pierwszy konsument komunikatów dla miejsca docelowego, w którym właściwość PERSISTENCE ma wartość HIGH, klasy WebSphere MQ dla usługi JMS wywołują wywołanie MQINQ w celu określenia, czy parametr NPMCLASS (HIGH) jest ustawiony w bazowej kolejce produktu WebSphere MQ . W związku z tym aplikacja musi mieć uprawnienia do wykonywania zapytań w kolejce. Dodatkowo klasy produktu WebSphere MQ classes for JMS preserują wynik wywołania MQINQ do momentu usunięcia miejsca docelowego i nie wywoła więcej wywołań MQINQ. Dlatego w przypadku zmiany ustawienia NPMCLASS w kolejce bazowej, gdy aplikacja nadal korzysta z miejsca docelowego, klasy WebSphere MQ classes for JMS nie zauważają nowego ustawienia.

Zezwalając na umieszczanie trwałych komunikatów JMS w kolejkach WebSphere MQ jako nietrwałych komunikatów produktu WebSphere MQ , uzyskujesz wydajność kosztem pewnej niezawodności. Jeśli

w przypadku trwałych komunikatów JMS wymagana jest maksymalna niezawodność, nie należy wysyłać komunikatów do miejsca docelowego, w którym właściwość PERSISTENCE ma wartość HIGH.

Warstwa JMS może używać systemu SYSTEM.JMS.TEMPQ.MODEL, zamiast SYSTEM.DEFAULT.MODEL.QUEUE. SYSTEM.JMS.TEMPQ.MODEL tworzy trwałe kolejki dynamiczne, które akceptują komunikaty trwałe, ponieważ SYSTEM.DEFAULT.MODEL.QUEUE nie można zaakceptować trwałych komunikatów. Aby używać kolejek tymczasowych do akceptowania trwałych komunikatów, należy użyć systemu SYSTEM.JMS.TEMPQ.MODELlub zmień kolejkę modelową na wybraną przez siebie kolejkę alternatywną.

Korzystanie z protokołu SSL (Secure Sockets Layer) z klasami produktu WebSphere MQ dla usługi JMS

Klasy produktu WebSphere MQ dla aplikacji JMS mogą używać szyfrowania SSL. W tym celu wymagane jest, aby dostawca JSSE był wymagany.

Klasy WebSphere MQ classes for JMS connections using TRANSPORT (CLIENT) obsługują szyfrowanie SSL (Secure Sockets Layer). Protokół SSL zapewnia szyfrowanie komunikacji, uwierzytelnianie i integralność komunikatów. Jest on zwykle używany do zabezpieczania komunikacji między dowolnymi dwoma równorzędnymi płatnikami w sieci Internet lub w intranecie.

Klasy WebSphere MQ classes for JMS używają rozszerzenia Java Secure Socket Extension (JSSE) do obsługi szyfrowania SSL i dlatego wymaga dostawcy JSSE. Maszyny JVM JSE v1.4 mają wbudowaną dostawcę JSSE. Szczegółowe informacje na temat zarządzania certyfikatami i ich przechowywania mogą być różne od dostawcy. Aby uzyskać informacje na ten temat, zapoznaj się z dokumentacją dostawcy JSSE.

W tej sekcji założono, że dostawca JSSE jest poprawnie zainstalowany i skonfigurowany oraz że odpowiednie certyfikaty zostały zainstalowane i udostępnione dla dostawcy JSSE. Teraz można użyć obiektu JMSAdmin, aby ustawić liczbę właściwości administracyjnych.

Jeśli klasy produktu WebSphere MQ dla aplikacji JMS korzystają z tabeli definicji kanału klienta (CCDT) w celu nawiązania połączenia z menedżerem kolejek, należy zapoznać się z “Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM WebSphere MQ classes for JMS” na stronie 941.

Właściwość obiektu SSLCIPHERSUITE

Ustaw SSLCIPHERSUITE, aby włączyć szyfrowanie SSL dla obiektu ConnectionFactory .

Aby włączyć szyfrowanie SSL dla obiektu ConnectionFactory , należy użyć obiektu JMSAdmin w celu ustawienia właściwości SSLCIPHERSUITE na pakiet CipherSuite obsługiwany przez dostawcę JSSE. Musi to być zgodne ze specyfikacją CipherSpec ustawioną na kanale docelowym. Jednak pakiety CipherSuites różnią się od specyfikacji CipherSpecs i dlatego mają różne nazwy. Produkt “Specyfikacje szyfrowania CipherSpecs i CipherSuites w usłudze JMS” na stronie 935 zawiera tabelę odwzorowania specyfikacji CipherSpecs obsługiwaną przez produkt WebSphere MQ na równoważną wartość parametru CipherSuites w postaci znanej dla JSSE. Więcej informacji na temat specyfikacji CipherSpecs i CipherSuites w produkcie WebSphere MQzawiera sekcja Zabezpieczenia.

Na przykład, aby skonfigurować obiekt ConnectionFactory , którego można użyć do utworzenia połączenia przez kanał MQI z włączoną obsługą SSL z atrybutem CipherSpec o wartości RC4_MD5_EXPORT, należy wprowadzić następującą komendę do obiektu JMSAdmin:

```
ALTER CF(my.cf) SSLCIPHERSUITE(SSL_RSA_EXPORT_WITH_RC4_40_MD5)
```

Tę opcję można również ustawić za pomocą aplikacji, używając metody setSSLCipherSuite () w obiekcie MQConnectionFactory .

Dla wygody, jeśli właściwość CipherSpec jest określona we właściwości SSLCIPHERSUITE, JMSAdmin próbuje odwzorować wartość CipherSpec na odpowiedni zestaw CipherSuite i generuje ostrzeżenie. Ta próba odwzorowania nie jest podejmowana, jeśli właściwość jest określona przez aplikację.

Aby zmienić wartość, należy użyć tabeli definicji kanału klienta (CCDT). Więcej informacji na ten temat zawiera sekcja “Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM WebSphere MQ classes for JMS” na stronie 941.

Właściwość obiektu SSLFIPSREQUIRED
Jeśli wymagane jest połączenie z użyciem pakietu CipherSuite obsługiwanego przez dostawcę IBM Java JSSE FIPS (IBMJSSEFIPS), ustaw właściwość SSLFIPSREQUIRED fabryki połączeń na wartość YES.

Wartością domyślną tej właściwości jest NO, co oznacza, że połączenie może używać dowolnego zestawu CipherSuite , który jest obsługiwany przez produkt WebSphere MQ.

Jeśli aplikacja używa więcej niż jednego połączenia, wartość atrybutu SSLFIPSREQUIRED używana, gdy aplikacja tworzy pierwsze połączenie, określa wartość używaną podczas tworzenia kolejnego połączenia przez aplikację. Oznacza to, że wartość właściwości SSLFIPSREQUIRED fabryki połączeń używanej do tworzenia kolejnego połączenia jest ignorowana. Aby użyć innej wartości parametru SSLFIPSREQUIRED, należy zrestartować aplikację.

Aplikacja może ustawić tę właściwość, wywołując metodę setSSLFipsRequired () obiektu ConnectionFactory . Właściwość ta jest ignorowana, jeśli nie jest ustawiona opcja CipherSuite .

Zadania pokrewne
Określanie, że w czasie wykonywania w kliencie MQI są używane tylko specyfikacje CipherSpecs z certyfikatem FIPS
Odsyłacze pokrewne
Standardy FIPS (Federal Information Processing Standards) dla systemów UNIX, Linux i Windows

Właściwość obiektu SSLPEERNAME
Użyj funkcji SSLPEERNAME, aby określić wzorzec nazwy wyróżniającej, aby upewnić się, że aplikacja JMS łączy się z poprawnym menedżerem kolejek.

Aplikacja JMS może się upewnić, że łączy się z poprawnym menedżerem kolejek, określając wzorzec nazwy wyróżniającej (DN). Połączenie powiedzie się tylko wtedy, gdy menedżer kolejek przedstawia nazwę wyróżniającą zgodną z wzorcem. Więcej informacji na temat formatu tego wzorca można znaleźć w tematach pokrewnych.

Nazwa wyróżniająca (DN) jest ustawiana przy użyciu właściwości SSLPEERNAME obiektu ConnectionFactory . Na przykład następująca komenda JMSAdmin ustawia obiekt ConnectionFactory , aby oczekiwać, że menedżer kolejek zidentyfikuje się ze wspólną nazwą rozpoczynającą się od znaków `QMGR.` i z co najmniej dwiema nazwami jednostek organizacyjnych, z których pierwszym musi być IBM i drugim WEBSPHERE:

```
ALTER CF(my.cf) SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Podczas sprawdzania nie jest rozróżniana wielkość liter, a w miejsce przecinków mogą być używane średniki. Parametr SSLPEERNAME można również ustawić z aplikacji przy użyciu metody setSSLPeerName () w obiekcie MQConnectionFactory . Jeśli ta właściwość nie jest ustawiona, sprawdzanie nie jest wykonywane na podstawie nazwy wyróżniającej podanej przez menedżer kolejek. Ta właściwość jest ignorowana, jeśli nie jest ustawiona opcja CipherSuite .

Właściwość obiektu SSLCERTSTORES
Użyj komendy SSLCERTSTORES, aby określić listę serwerów LDAP, które mają być używane na potrzeby sprawdzania listy odwołań certyfikatów (CRL).

Często używana jest lista odwołań certyfikatów (CRL) do identyfikowania certyfikatów, które nie są już zaufane. Listy CRL są zwykle udostępniane na serwerach LDAP. Usługa JMS umożliwia określenie serwera LDAP na potrzeby sprawdzania listy CRL w środowisku Java 2 v1.4 lub nowszym. Następujący przykład JMSAdmin powoduje, że usługa JMS ma używać listy CRL udostępnianej na serwerze LDAP o nazwie crl1.ibm.com:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com)
```

Uwaga: Aby pomyślnie użyć narzędzia CertStore z listą CRL udostępnianą na serwerze LDAP, należy upewnić się, że pakiet Java Software Development Kit (SDK) jest kompatybilny z CRL. Niektóre pakiety

SDK wymagają, aby lista CRL była zgodna z dokumentem RFC 2587, który definiuje schemat dla LDAP v2. Większość serwerów LDAP v3 używa zamiast niego RFC 2256.

Jeśli serwer LDAP nie jest uruchomiony na domyślnym porcie 389, można określić port, dodając dwukropek (:) i numer portu do nazwy hosta. Jeśli certyfikat prezentowany przez menedżera kolejek znajduje się na liście CRL udostępnionej na stronie crl1.ibm.com, połączenie nie zostanie zakończone. Aby uniknąć pojedynczego punktu awarii, usługa JMS zezwala na dostarczanie wielu serwerów LDAP, dostarczając listę serwerów LDAP rozdzielonych znakiem spacji. Oto przykład:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com ldap://crl2.ibm.com)
```

Jeśli określono wiele serwerów LDAP, JMS próbuje każdy z nich na kolei, dopóki nie znajdzie serwera, z którym będzie mógł pomyślnie zweryfikować certyfikat menedżera kolejek. Każdy serwer musi zawierać identyczne informacje.

Łańcuch w tym formacie może być dostarczony przez aplikację w metodzie MQConnectionFactory.setSSLCertStores (). Alternatywnie aplikacja może utworzyć jedną lub więcej obiektów java.security.cert.CertStore , umieścić je w odpowiednim obiekcie kolekcji i dostarczyć ten obiekt kolekcji do metody setSSLCertStores (). W ten sposób aplikacja może dostosować sprawdzanie list CRL. Szczegółowe informacje na temat konstruowania i używania obiektów CertStore można znaleźć w dokumentacji JSSE.

Poprawność certyfikatu prezentowanego przez menedżera kolejek przy ustawionym połączeniu jest sprawdzana w następujący sposób:

1. Pierwszy obiekt CertStore w kolekcji identyfikowanej przez sslCertStores służy do identyfikacji serwera CRL.
2. Podjęto próbę skontaktowania się z serwerem CRL.
3. Jeśli próba zakończy się pomyślnie, serwer jest wyszukiwany pod kątem zgodności z certyfikatem.
   a. Jeśli certyfikat zostanie unieważniony, proces wyszukiwania zakończy się, a żądanie nawiązania połączenia nie powiedzie się i zostanie odebrany kod przyczyny MQRC_SSL_CERTIFICATE_ODWOŁANE.
   b. Jeśli certyfikat nie zostanie znaleziony, proces wyszukiwania zostanie nadany, a połączenie może być kontynuowane.
4. Jeśli próba skontaktowania się z serwerem nie powiedzie się, następny obiekt CertStore jest używany do identyfikacji serwera CRL, a proces jest powtarzany z kroku 2.

   Jeśli był to ostatni element CertStore w kolekcji lub jeśli kolekcja nie zawiera obiektów CertStore , proces wyszukiwania nie powiódł się i żądanie połączenia nie powiedzie się i kod przyczyny MQRC_SSL_CERT_STORE_ERROR.

Obiekt Collection określa kolejność, w jakiej używane są CertStores .

Jeśli aplikacja korzysta z funkcji setSSLCertStores () w celu ustawienia kolekcji obiektów CertStore , obiekt MQConnectionFactory nie może być już powiązany z przestrzenią nazw JNDI. Próba wykonania tego zadania powoduje wystąpienie wyjątku. Jeśli właściwość sslCertStores nie jest ustawiona, sprawdzanie odwołań nie jest wykonywane na certyfikacie udostępnionym przez menedżer kolejek. Ta właściwość jest ignorowana, jeśli nie jest ustawiona opcja CipherSuite .

Właściwość obiektu SSLRESETCOUNT
Ta właściwość reprezentuje łączną liczbę bajtów wysłanych i odebranych przez połączenie, zanim klucz tajny używany do szyfrowania jest ponownie negocjowany.

Liczba wysłanych bajtów jest liczbą przed zaszyfrowaniem, a liczba odebranych bajtów jest liczbą po deszyfrowaniu. Liczba bajtów obejmuje również informacje sterujące wysłane i odebrane przez klasy produktu WebSphere MQ dla usługi JMS.

Na przykład, aby skonfigurować obiekt ConnectionFactory , którego można użyć do utworzenia połączenia przez kanał MQI z włączoną obsługą SSL przy użyciu klucza tajnego, który jest ponownie negocjowany po 4 MB danych, należy wprowadzić następującą komendę do obiektu JMSAdmin:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Aplikacja może ustawić tę właściwość, wywołując metodę setSSLResetCount () obiektu ConnectionFactory .

Jeśli wartość tej właściwości jest równa zero, która jest wartością domyślną, klucz tajny nigdy nie jest ponownie negocjowany. Właściwość ta jest ignorowana, jeśli nie jest ustawiona opcja CipherSuite .

Właściwość obiektu SSLSocketFactory
Aby dostosować inne aspekty połączenia SSL dla aplikacji, należy utworzyć fabrykę SSLSocketFactory i skonfigurować ją w taki sposób, aby korzystała z niej.

Użytkownik może dostosować inne aspekty połączenia SSL dla aplikacji. Na przykład można zainicjować sprzęt szyfrujący lub zmienić używany magazyn kluczy i magazyn zaufanych certyfikatów. W tym celu aplikacja musi najpierw utworzyć obiekt javax.net.ssl.SSLSocketFactory , który jest odpowiednio dostosowany. Informacje na temat tego, jak to zrobić, można znaleźć w dokumentacji JSSE, ponieważ konfigurowalne składniki różnią się od dostawcy. Po uzyskaniu odpowiedniego obiektu SSLSocketFactory należy użyć metody MQConnectionFactory.setSSLSocketFactory (), aby skonfigurować usługę JMS w taki sposób, aby korzystała z dostosowanego obiektu SSLSocketFactory .

Jeśli aplikacja korzysta z metody setSSLSocketFactory () w celu ustawienia dostosowanego obiektu SSLSocketFactory , obiekt MQConnectionFactory nie może być już powiązany z przestrzenią nazw JNDI. Próba wykonania tego zadania powoduje wystąpienie wyjątku. Jeśli ta właściwość nie jest ustawiona, zostanie użyty domyślny obiekt SSLSocketFactory . Szczegółowe informacje na temat zachowania domyślnego obiektu SSLSocketFactory można znaleźć w dokumentacji rozszerzenia JSSE. Ta właściwość jest ignorowana, jeśli nie jest ustawiona opcja CipherSuite .

Ważne: Nie należy zakładać, że użycie właściwości SSL zapewnia bezpieczeństwo, gdy obiekt ConnectionFactory jest pobierany z przestrzeni nazw JNDI, która nie jest sama w sobie zabezpieczona. W szczególności standardowa implementacja protokołu LDAP w interfejsie JNDI nie jest zabezpieczona. Atakujący może naśladować serwer LDAP, wprowadzając w błąd aplikację JMS do łączenia się z niewłaściwym serwerem bez zauważania. Dzięki odpowiednim uzgodnieniom zabezpieczeń inne implementacje interfejsu JNDI (takie jak implementacja protokołu fscontext) są zabezpieczone.

Wprowadzanie zmian w magazynie kluczy JSSE lub magazynie zaufanych certyfikatów
W przypadku wprowadzenia zmian w magazynie kluczy lub magazynie zaufanych certyfikatów należy podjąć określone działania, aby zmiany zostały pobrane.

Jeśli zawartość magazynu kluczy lub magazynu zaufanych certyfikatów JSSE zostanie zmieniona lub zostanie zmieniona lokalizacja pliku kluczy lub pliku zaufanych certyfikatów, klasy WebSphere MQ dla aplikacji JMS, które są uruchomione w danym momencie, nie pobierają automatycznie zmian. Aby zmiany zostały uwzględnione, muszą zostać wykonane następujące działania:

- Aplikacje muszą zamknąć wszystkie połączenia i zniszczyć wszelkie nieużywane połączenia w pulach połączeń.
- Jeśli dostawca JSSE buforuje informacje z magazynu kluczy i magazynu zaufanych certyfikatów, te informacje muszą zostać odświeżone.

Po wykonaniu tych czynności aplikacje mogą następnie ponownie utworzyć połączenia.

W zależności od sposobu zaprojektowania aplikacji oraz funkcji udostępnianej przez dostawcę JSSE możliwe jest wykonanie tych działań bez zatrzymywania i restartowania aplikacji. Jednak zatrzymywanie i restartowanie aplikacji może być najprostszym rozwiązaniem.

Specyfikacje szyfrowania CipherSpecs i CipherSuites w usłudze JMS
Specyfikacje CipherSpecs obsługiwane przez produkt WebSphere MQ i odpowiadające im pakiety CipherSuites.

Tabela 129 na stronie 936 zawiera listę specyfikacji CipherSpecs obsługiwanych przez produkt WebSphere MQ oraz ich odpowiedniki CipherSuites. Jeśli właściwość ConnectionFactory SSLFIPSREQUIRED ma ustawioną wartość NO, klasy produktu WebSphere MQ dla aplikacji JMS

mogą łączyć się z menedżerem kolejek, jeśli na końcu kanału MQI określono dowolną obsługiwaną wartość CipherSpec , a na końcu klienta jest określony odpowiednik CipherSuite . Jeśli parametr SSLFIPSREQUIRED jest ustawiony na wartość YES, to kombinacja atrybutów CipherSpec i CipherSuite określa, czy aplikacja może nawiązać połączenie z menedżerem kolejek.

Na końcu serwera kanału MQI nazwa CipherSpec może zostać określona jako wartość parametru SSLCIPH w komendzie DEFINE CHANNEL CHLTYPE (SVRCONN). Na końcu kanału MQI klienta nazwa CipherSuite może zostać określona w następujący sposób:

- Aplikacja może wywołać metodę setSSLCipherSuite () obiektu ConnectionFactory .
- Za pomocą narzędzia administracyjnego JMS produktu WebSphere MQ można ustawić właściwość SSLCIPHERSUITE obiektu ConnectionFactory .

Tabela 129. CipherSpecs obsługiwane przez produkt WebSphere MQ i ich odpowiedniki CipherSuites

CipherSpec	**Odpowiednik CipherSuite**	**Połączenie można nawiązać, jeśli wartość SFIPS [1] jest ustawiona na YES?**
NULL_MD5	SSL_RSA_WITH_NULL_MD5	Nie
NULL_SHA	SSL_RSA_WITH_NULL_SHA	Nie
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5	Nie
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	Nie
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA	Nie
RC2_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	Nie
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA	Nie
RC4_56_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA	Nie
DES_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA	Nie
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Nie
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	Nie [7]
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	Tak [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	Tak [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	Tak [5] [7]
AES_SHA_US[2]		
TLS_RSA_WITH_DES_CBC_SHA[8] [9]	SSL_RSA_WITH_DES_CBC_SHA	Nie[3]
TLS_RSA_WITH_3DES_EDE_CBC_SHA[8]	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Tak
FIPS_WITH_DES_CBC_SHA	SSL_RSA_FIPS_WITH_DES_CBC_SHA	Nie[4]
FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA	Nie[6]

Uwagi:

1. Jeśli używane jest narzędzie administracyjne WebSphere MQ JMS, SFIPS jest skróconą nazwą właściwości ConnectionFactory SSLFIPSREQUIRED.

2. Ta właściwość CipherSpec nie ma odpowiednika CipherSuite.
3. Ta specyfikacja CipherSpec miała certyfikat FIPS 140-2 przed 19th maja 2007.
4. Ta specyfikacja CipherSpec miała certyfikat FIPS 140-2 przed 19th maja 2007. Nazwa FIPS_WITH_DES_CBC_SHA jest historyczna i odzwierciedla fakt, że ta specyfikacja CipherSpec była poprzednio (ale nie jest już) zgodna ze standardem FIPS. Ta specyfikacja szyfrowania jest nieaktualna i jej użycie nie jest zalecane.
5. Tych specyfikacji CipherSpecs (TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) nie można użyć do zabezpieczenia połączenia z programu WebSphere MQ Explorer do menedżera kolejek, chyba że odpowiednie nieograniczone pliki strategii są stosowane do środowiska JRE używanego przez eksplorator.

   Więcej informacji na temat plików strategii zawiera sekcja Informacje o zabezpieczeniach .
6. Nazwa FIPS_WITH_3DES_EDE_CBC_SHA jest historyczna i odzwierciedla fakt, że ta specyfikacja CipherSpec była poprzednio (ale nie jest już) zgodna ze standardem FIPS. Ta specyfikacja szyfrowania jest nieaktualna i jej użycie nie jest zalecane.
7. Te CipherSpecs (TLS_RSA_WITH_NULL_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) wymagają środowisk IBM 6.0 SR13 FP2 , 7.0 SR4 FP2 lub nowszego.
8. Te CipherSpecs (TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_DES_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA256) mogą używać zarówno protokołu SSLv3 , jak i TLS. Domyślnie, gdy tryb FIPS nie jest włączony, używany jest protokół SSLv3 . Aby użyć protokołu TLS, ustaw właściwość systemową Java **com.ibm.mq.cfg.preferTLS** na wartość `true`.
9. Ta specyfikacja CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA jest nieaktualna. Jednak może być ona nadal używana do przesyłania do 32 GB danych, zanim połączenie zostanie zakończone z błędem AMQ9288. Aby uniknąć tego błędu, należy unikać używania potrójnego algorytmu szyfrowania DES lub włączyć resetowanie klucza tajnego podczas korzystania z tej specyfikacji CipherSpec.

Informacje pokrewne

Określanie, że w czasie wykonywania w kliencie MQI są używane tylko specyfikacje CipherSpecs z certyfikatem FIPS

Standardy FIPS (Federal Information Processing Standards) dla systemów UNIX, Linux i Windows

Zapisywanie wyjść kanału w języku Java dla klas produktu WebSphere MQ dla usługi JMS

Wyjścia kanału można utworzyć, definiując klasy Java, które implementują określone interfejsy.

W pakiecie com.ibm.mq.exits są zdefiniowane trzy interfejsy:

- WMQSendExit, dla wyjścia wysyłania
- WMQReceiveExit, dla wyjścia odbierania
- WMQSecurityExit, dla wyjścia zabezpieczeń

Poniższy przykładowy kod definiuje klasę, która implementuje wszystkie trzy interfejsy:

```
public class MyMQExits implements
WMQSendExit, WMQReceiveExit, WMQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method implements the send exit interface
  public ByteBuffer channelSendExit(
MQCXP channelExitParms,
                                    MQCD channelDefinition,
                                    ByteBuffer agentBuffer)
  {
    // Complete the body of the send exit here
  }
    // This method implements the receive exit interface
```

```
  public ByteBuffer channelReceiveExit(
MQCXP channelExitParms,
                                      MQCD channelDefinition,
                                      ByteBuffer agentBuffer)
  {
    // Complete the body of the receive exit here
  }
    // This method implements the security exit interface
  public ByteBuffer channelSecurityExit(
MQCXP channelExitParms,
                                       MQCD channelDefinition,
                                       ByteBuffer agentBuffer)
  {
    // Complete the body of the security exit here
  }
}
```

Każde wyjście otrzymuje jako parametry obiekt MQCXP i obiekt MQCD. Obiekty te reprezentują struktury MQCXP i MQCD zdefiniowane w interfejsie proceduralnym.

Gdy wywoływane jest wyjście wysyłania, parametr agentBuffer zawiera dane, które mają zostać wysłane do menedżera kolejek serwera. Parametr długości nie jest wymagany, ponieważ wyrażenie agentBuffer.limit () udostępnia długość danych. Wyjście wysyłania zwraca jako wartość dane, które mają zostać wysłane do menedżera kolejek serwera. Jeśli jednak wyjście wysyłania nie jest ostatnim wyjściem wysyłania w sekwencji wysyłania wyjść, zwrócone dane są przekazywane do następnego wyjścia wysyłania w sekwencji. Wyjście wysyłania może zwrócić zmodyfikowaną wersję danych, które otrzymuje w parametrze agentBuffer , lub może zwrócić dane bez zmian. Najprostszym możliwym organem wyjściowym jest zatem:

```
{ return agentBuffer; }
```

Po wywołaniu wyjścia odbierania parametr agentBuffer zawiera dane, które zostały odebrane z menedżera kolejek serwera. Wyjście odbierania zwraca jako wartość dane, które mają być przekazane do aplikacji przez klasy WebSphere MQ dla usługi JMS. Jeśli jednak wyjście odbierania nie jest ostatnim wyjściem odbioru w sekwencji wyjść odbierania, zwrócone dane są przekazywane do następnego wyjścia odbierania w sekwencji.

Po wywołaniu wyjścia zabezpieczeń parametr agentBuffer zawiera dane, które zostały odebrane w przepływie zabezpieczeń od wyjścia zabezpieczeń na końcu serwera połączenia. Wyjście zabezpieczeń zwraca jako wartość dane, które mają zostać wysłane w przepływie zabezpieczeń do wyjścia zabezpieczeń serwera.

Wyjścia kanału są wywoływane z buforem, w którym znajduje się tablica zapasowa. Aby uzyskać najlepszą wydajność, wyjście powinno zwrócić bufor z tablicą bazową.

Do wyjścia kanału można przekazać do 32 znaków danych użytkownika, gdy jest on wywoływany. Wyjście uzyskuje dostęp do danych użytkownika, wywołując metodę getExitData () obiektu MQCXP. Chociaż wyjście może zmienić dane użytkownika, wywołując metodę setExitData (), dane użytkownika są odświeżane za każdym razem, gdy wywoływane jest wyjście. W związku z tym wszelkie zmiany wprowadzone w danych użytkownika są tracone. Wyjście może jednak przekazywać dane z jednego wywołania do następnego przy użyciu obszaru użytkownika wyjścia obiektu MQCXP. Wyjście uzyskuje dostęp do obszaru użytkownika wyjścia przez odwołanie, wywołując metodę getExitUserArea().

Każda klasa wyjścia musi mieć konstruktor. Konstruktorem może być konstruktor domyślny, tak jak pokazano to w poprzednim przykładzie, lub konstruktor z parametrem łańcuchowym. Konstruktor jest wywoływany w celu utworzenia instancji klasy wyjścia dla każdego wyjścia zdefiniowanego w klasie. Dlatego w poprzednim przykładzie instancja klasy MyMQExits jest tworzona dla wyjścia wysyłania, inna instancja jest tworzona dla wyjścia odbierania, a dla wyjścia zabezpieczeń tworzona jest trzecia instancja. Gdy wywoływany jest konstruktor z parametrem łańcuchowym, parametr ten zawiera te same dane użytkownika, które są przekazywane do wyjścia kanału, dla którego tworzona jest instancja. Jeśli klasa wyjścia ma zarówno konstruktor domyślny, jak i konstruktor pojedynczego parametru, pierwszeństwo ma konstruktor pojedynczego parametru.

Nie zamykać połączenia z poziomu wyjścia kanału.

Po wysłaniu danych do końca połączenia z serwerem, szyfrowanie SSL jest wykonywane *po* wywołaniu dowolnego wyjścia kanału. Podobnie, gdy dane są odbierane z końca połączenia serwera, deszyfrowanie SSL jest wykonywane *przed* wywołaniem dowolnego wyjścia kanału.

W wersjach produktu WebSphere MQ classes for JMS wcześniejszych niż wersja 7.0programy obsługi wyjść kanału zostały zaimplementowane przy użyciu interfejsów MQSendExit, MQReceiveExiti MQSecurityExit. Nadal można używać tych interfejsów, ale nowe interfejsy są preferowane w celu poprawy funkcji i wydajności.

Konfigurowanie produktu IBM WebSphere MQ classes for JMS do korzystania z wyjść kanału

Aplikacja IBM WebSphere MQ classes for JMS może używać zabezpieczeń kanału, wysyłania i odbierania wyjść z kanału MQI, który jest uruchamiany, gdy aplikacja łączy się z menedżerem kolejek. Aplikacja może używać wyjść napisanych w Java, C lub C + +. Aplikacja może również używać sekwencji wyjść wysyłania lub odbierania, które są uruchamiane w ramach dziedziczenia.

Następujące właściwości są używane przez określenie wyjścia wysyłania lub sekwencji wyjść wysyłania używanych przez połączenie JMS :

- Właściwość **SENDEXIT** obiektu MQConnectionFactory .
- Właściwość **sendexit** w specyfikacji aktywowania używanej przez adapter zasobów produktu IBM WebSphere MQ na potrzeby komunikacji przychodzącej,
- Właściwość **sendexit** w obiekcie ConnectionFactory używanym przez adapter zasobów IBM WebSphere MQ do komunikacji wyjściowej.

Wartość właściwości jest łańcuchem składowym, który składa się z co najmniej jednego elementu rozdzielonego przecinkami. Każdy element identyfikuje wyjście wysyłania w jeden z następujących sposobów:

- Nazwa klasy, która implementuje interfejs `WMQSendExit` dla wyjścia wysyłania napisanego w produkcie Java.
- Łańcuch w formacie *libraryName* (*entryPointnazwa*) dla wyjścia wysyłania napisanego w języku C lub C + +.

W podobny sposób, następujące właściwości określają wyjście odbierania lub sekwencję wyjść odbierania, używane przez połączenie:

- Właściwość **RECEXIT** obiektu MQConnectionFactory .
- Właściwość **receiveexit** w specyfikacji aktywowania używanej przez adapter zasobów produktu IBM WebSphere MQ na potrzeby komunikacji przychodzącej,
- Właściwość **receiveexit** w obiekcie ConnectionFactory używanym przez adapter zasobów IBM WebSphere MQ do komunikacji wyjściowej.

Następujące właściwości określają wyjście zabezpieczeń używane przez połączenie:

- Właściwość **SECEXIT** obiektu MQConnectionFactory .
- Właściwość **securityexit** w specyfikacji aktywowania używanej przez adapter zasobów produktu IBM WebSphere MQ na potrzeby komunikacji przychodzącej,
- Właściwość **securityexit** w obiekcie ConnectionFactory używanym przez adapter zasobów IBM WebSphere MQ do komunikacji wyjściowej.

W przypadku właściwości MQConnectionFactoriesmożna ustawić właściwości **SENDEXIT**, **RECEXIT** i **SECEXIT** , używając narzędzia administracyjnego IBM WebSphere MQ JMS lub IBM WebSphere MQ Explorer. Alternatywnie, aplikacja może ustawić właściwości, wywołując metody `setSendExit()`, `setReceiveExit()`i `setSecurityExit()` .

Wyjścia kanału są ładowane przez własny program ładujący klasy. Aby znaleźć wyjście kanału, program ładujący klasy przeszukuje następujące lokalizacje w podanej kolejności.

1. Ścieżka klasy określona przez właściwość **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** lub przez atrybut **JavaExitsClassPath** w sekcji Kanały w pliku konfiguracyjnym klienta IBM WebSphere MQ .
2. Ścieżka klasy określona przez właściwość systemową Java **com.ibm.mq.exitClasspath**. Należy pamiętać, że ta właściwość jest już nieaktualna.
3. Katalog programu IBM WebSphere MQ kończy działanie, jak pokazano na rysunku Tabela 130 na stronie 940. Program ładujący klasy najpierw wyszukuje w katalogu pliki klas, które nie są spakowane w plikach archiwum Java (JAR). Jeśli wyjście kanału nie zostanie znalezione, program ładujący klasy przeszukuje następnie pliki JAR w katalogu.

Tabela 130. Katalog wyjść programu IBM WebSphere MQ

Platforma	**Katalog**
UNIX and Linux	`/var/mqm/exits` (32-bitowe wyjścia kanału) `/var/mqm/exits64` (64-bitowe wyjścia kanału)
Windows	*katalog_danych_instalacji*\exits gdzie *katalog_danych_instalacji* to katalog, który został wybrany dla plików danych programu IBM WebSphere MQ podczas instalacji. Katalog domyślny to `C:\Program Files\IBM\WebSphere MQ`.

Uwaga: Jeśli wyjście kanału istnieje w więcej niż jednej lokalizacji, program IBM WebSphere MQ classes for JMS ładuje pierwszą instancję, która zostanie znaleziona.

Elementem nadrzędnym programu ładującego klasy jest program ładujący klasy, który jest używany do ładowania IBM WebSphere MQ classes for JMS. Dlatego program ładujący klasy macierzyste może załadować wyjście kanału, jeśli nie można go znaleźć w żadnej z poprzednich lokalizacji. Jeśli jednak produkt IBM WebSphere MQ classes for JMS jest używany w środowisku, takim jak serwer aplikacji JEE , prawdopodobnie nie ma możliwości wywierania wpływu na wybór nadrzędnego programu ładującego klasy, dlatego program ładujący klasy powinien zostać skonfigurowany przez ustawienie właściwości systemowej Java **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** na serwerze aplikacji.

Jeśli aplikacja jest uruchamiana z włączoną obsługą programu Java Security Manager, plik konfiguracyjny strategii używany przez środowisko wykonawcze produktu Java , w którym działa aplikacja, musi mieć uprawnienia do ładowania klasy wyjścia kanału. Więcej informacji na ten temat zawiera sekcja Uruchamianie klas produktu IBM MQ dla aplikacji JMS w ramach menedżera zabezpieczeń Java.

Interfejsy `MQSendExit`, `MQReceiveExit`i `MQSecurityExit` dostarczane z wersjami produktu IBM WebSphere MQ wcześniejszymi niż Version 7.0 są nadal obsługiwane. Jeśli używane są wyjścia kanału, które implementują te interfejsy, produkt `com.ibm.mq.jar` musi być obecny w ścieżce klasy.

Informacje na temat pisania wyjść kanału w języku C zawiera sekcja “Programy obsługi wyjścia kanału dla kanałów przesyłania komunikatów” na stronie 406. Należy zapisać programy obsługi wyjścia kanału napisane w języku C lub C + + w katalogu podanym w sekcji Tabela 130 na stronie 940.

Jeśli aplikacja korzysta z tabeli definicji kanału klienta (CCDT) w celu nawiązania połączenia z menedżerem kolejek, należy zapoznać się z “Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM WebSphere MQ classes for JMS” na stronie 941.

Określanie danych użytkownika, które mają być przekazywane do wyjść kanału podczas korzystania z klas WebSphere MQ dla usługi JMS

Do wyjścia kanału można przekazać do 32 znaków danych użytkownika, gdy jest on wywoływany.

Właściwość SENDEXITINIT obiektu MQConnectionFactory określa dane użytkownika, które są przekazywane do każdego wyjścia wysyłania podczas jego wywołania. Wartość właściwości to łańcuch składający się z co najmniej jednego elementu danych użytkownika oddzielonych przecinkami. Pozycja każdego elementu danych użytkownika w łańcuchu określa, które wyjście wysyłania, w sekwencji wysyłania wyjść, dane użytkownika są przekazywane do. Na przykład pierwsza pozycja danych

użytkownika w łańcuchu jest przekazywana do pierwszego wyjścia wysyłania w sekwencji wysyłania wyjść.

Właściwość SENDEXITINIT można ustawić, korzystając z narzędzia administracyjnego WebSphere MQ JMS lub programu WebSphere MQ Explorer. Alternatywnie, aplikacja może ustawić właściwość, wywołując metodę setSendExitInit().

W podobny sposób właściwość RECEXITINIT obiektu ConnectionFactory określa dane użytkownika, które są przekazywane do każdego wyjścia odbierania, a właściwość SECEXITINIT określa dane użytkownika przekazywane do wyjścia zabezpieczeń. Te właściwości można ustawić za pomocą narzędzia administracyjnego WebSphere MQ JMS lub programu WebSphere MQ Explorer. Alternatywnie, aplikacja może ustawić właściwości, wywołując metody setReceiveExitInit() i setSecurityExitInit().

Podczas określania danych użytkownika, które są przekazywane do wyjść kanału, należy pamiętać o następujących regułach:

- Jeśli liczba elementów danych użytkownika w łańcuchu jest większa niż liczba wyjść w sekwencji, nadmiarowe pozycje danych użytkownika są ignorowane.
- Jeśli liczba elementów danych użytkownika w łańcuchu jest mniejsza niż liczba wyjść w sekwencji, każdy nieokreślony element danych użytkownika jest ustawiany na pusty łańcuch. Dwa przecinki w spadku w łańcuchu lub przecinek na początku łańcucha oznaczają również nieokreślony element danych użytkownika.

Jeśli aplikacja korzysta z tabeli definicji kanału klienta (CCDT) w celu nawiązania połączenia z menedżerem kolejek, wszystkie dane użytkownika określone w definicji kanału połączenia klienta są przekazywane do wyjść kanału, gdy są wywoływane. Więcej informacji na temat korzystania z tabeli definicji kanału klienta zawiera sekcja “Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM WebSphere MQ classes for JMS” na stronie 941.

Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM WebSphere MQ classes for JMS

Klasy produktu IBM WebSphere MQ dla aplikacji JMS mogą korzystać z definicji kanału połączenia klienta zapisanych w tabeli definicji kanału klienta (CCDT). W celu użycia tabeli definicji kanału klienta należy skonfigurować obiekt ConnectionFactory . Istnieją pewne ograniczenia dotyczące jego stosowania.

Alternatywą dla tworzenia definicji kanału połączenia klienta przez ustawienie określonych właściwości obiektu ConnectionFactory aplikacja IBM WebSphere MQ classes for JMS może używać definicji kanału połączenia klienta zapisanych w tabeli definicji kanału klienta. Definicje te są tworzone za pomocą komend IBM WebSphere MQ Script Commands (MQSC) lub IBM WebSphere MQ Programmable Command Format (PCF). Gdy aplikacja tworzy obiekt połączenia, program IBM WebSphere MQ classes for JMS przeszukuje tabelę definicji kanału klienta pod kątem odpowiedniej definicji kanału połączenia klienckiego i używa definicji kanału w celu uruchomienia kanału MQI. Więcej informacji na temat tabel definicji kanału klienta i sposobu ich tworzenia zawiera sekcja Tabela definicji kanału klienta.

Aby można było używać tabeli definicji kanału klienta, właściwość CCDTURL obiektu ConnectionFactory musi być ustawiona na obiekt URL. Obiekt URL hermetyzuje jednolity wskaźnik zasobów (URL), który identyfikuje nazwę i położenie pliku zawierającego tabelę definicji kanału klienta i określa sposób dostępu do pliku. Właściwość CCDTURL można ustawić za pomocą narzędzia administracyjnego JMS produktu IBM WebSphere MQ lub można ustawić właściwość przez utworzenie obiektu adresu URL i wywołanie metody setCCDTURL() obiektu ConnectionFactory .

Jeśli na przykład plik ccdt1.tab zawiera tabelę definicji kanału klienta i jest przechowywany w tym samym systemie, w którym działa aplikacja, to aplikacja może ustawić właściwość CCDTURL w następujący sposób:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
factory.setCCDTURL(chanTab1);
```

Inny przykład: przypuśćmy, że plik ccdt2.tab zawiera tabelę definicji kanału klienta i jest przechowywany w systemie innym niż ten, w którym działa aplikacja. Jeśli dostęp do pliku można uzyskać za pomocą protokołu FTP, aplikacja może ustawić właściwość CCDTURL w następujący sposób:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
factory.setCCDTURL(chanTab2);
```

Oprócz ustawienia właściwości CCDTURL obiektu ConnectionFactory , właściwość QMANAGER tego samego obiektu musi być ustawiona na jedną z następujących wartości:

- Nazwa menedżera kolejek
- Gwiazdka (*), po której następuje nazwa grupy menedżerów kolejek
- Gwiazdka (*)
- Pusty łańcuch lub łańcuch zawierający wszystkie puste znaki

Są to te same wartości, których można użyć dla parametru *QMgrName* w wywołaniu MQCONN wywoływanym przez aplikację kliencką korzystaniową z interfejsu kolejki komunikatów (Message Queue Interface-MQI). Więcej informacji na temat znaczenia tych wartości znajduje się w sekcji MQCONN. Właściwość QMANAGER można ustawić za pomocą narzędzia administracyjnego WebSphere MQ JMS lub programu IBM WebSphere MQ Explorer. Alternatywnie, aplikacja może ustawić właściwość, wywołując metodę setQueueManager () obiektu ConnectionFactory .

Jeśli aplikacja utworzy obiekt połączenia z obiektu ConnectionFactory , program IBM WebSphere MQ classes for JMS uzyskuje dostęp do tabeli definicji kanału klienta identyfikowanej przez właściwość CCDTURL, korzysta z właściwości QMANAGER w celu wyszukania odpowiedniej definicji kanału połączenia klienckiego, a następnie używa definicji kanału w celu uruchomienia kanału MQI w menedżerze kolejek.

Należy zauważyć, że właściwości CCDTURL i CHANNEL obiektu ConnectionFactory nie mogą być ustawione w obu przypadkach, gdy aplikacja wywołuje metodę createConnection(). Jeśli ustawione są obie właściwości, metoda zgłasza wyjątek. Właściwość CCDTURL lub CHANNEL jest uznawana za ustawioną, jeśli jej wartością jest dowolna wartość inna niż null, pusty łańcuch lub łańcuch zawierający wszystkie puste znaki.

Gdy program IBM WebSphere MQ classes for JMS znajdzie odpowiednią definicję kanału połączenia klienta w tabeli definicji kanału klienta, używa ona tylko informacji wyodrębnionych z tabeli w celu uruchomienia kanału MQI. Wszystkie właściwości powiązane z kanałem obiektu ConnectionFactory są ignorowane.

W przypadku korzystania z protokołu SSL (Secure Sockets Layer) należy w szczególności zwrócić uwagę na następujące kwestie:

- Kanał MQI używa protokołu SSL tylko wtedy, gdy definicja kanału wyodrębniona z tabeli definicji kanału klienta określa nazwę partycji CipherSpec obsługiwanej przez produkt IBM WebSphere MQ classes for JMS.
- Tabela definicji kanału klienta zawiera również informacje na temat położenia serwerów LDAP (Lightweight Directory Access Protocol), które przechowują listy odwołań certyfikatów (CRL). Produkt IBM WebSphere MQ classes for JMS używa tylko tych informacji w celu uzyskania dostępu do serwerów LDAP, które przechowują listy CRL.
- Tabela definicji kanału klienta może również zawierać położenie odpowiadającego protokołu OCSP (Online Certificate Status Protocol). Produkt IBM WebSphere MQ classes for JMS nie może używać informacji OCSP w pliku tabeli definicji kanału klienta. Można jednak skonfigurować protokół OCSP w sposób opisany w sekcji Korzystanie z protokołu Online Certificate Protocol.

Więcej informacji na temat używania protokołu SSL z tabelą definicji kanału klienta zawiera sekcja Korzystanie z rozszerzonego klienta transakcyjnego z kanałami SSL.

Jeśli używane są wyjścia kanału, należy pamiętać również o następujących punktach:

- Kanał MQI korzysta tylko z wyjść kanału i powiązanych danych użytkownika określonych przez definicję kanału wyodrębnioną z tabeli definicji kanału klienta.
- Definicja kanału wyodrębniona z tabeli definicji kanału klienta może określać wyjścia kanału, które są zapisywane w języku Java. Oznacza to, na przykład, że parametr SCYEXIT w komendzie DEFINE CHANNEL w celu utworzenia definicji kanału połączenia klienta może określać nazwę klasy implementujący interfejs WMQSecurityExit . Podobnie, parametr SENDEXIT może określać nazwę

klasy implementujący interfejs WMQSendExit , a parametr RCVEXIT może określać nazwę klasy implementujący interfejs WMQReceiveExit . Więcej informacji na temat pisania wyjścia kanału w języku Java zawiera sekcja “Zapisywanie wyjść kanału w języku Java dla klas produktu WebSphere MQ dla usługi JMS” na stronie 937.

Obsługiwane jest również użycie wyjść kanału napisanych w języku innym niż Java. Więcej informacji na temat określania parametrów SCYEXIT, SENDEXIT i RCVEXIT w komendzie DEFINE CHANNEL dla wyjść kanału napisanych w innym języku zawiera sekcja DEFINE CHANNEL(DEFINIOWANIE KANAŁU).

Automatyczne ponowne połączenie klienta JMS

Skonfiguruj klienta JMS w taki sposób, aby ponownie nawiązał połączenie po awarii sieci, menedżera kolejek lub serwera.

Właściwości `CONNECTIONNAMELIST` i `CLIENTRECONNECTOPTIONS` klasy `MQConnectionFactory` umożliwiają skonfigurowanie połączenia klienta w celu automatycznego ponownego nawiązania połączenia po awarii połączenia lub żądania administracyjnego w celu ponownego nawiązania połączenia z aplikacjami klienckim po zatrzymaniu menedżera kolejek.

Pełna lista nazw połączeń znajdujących się na liście connectionNamejest dostępna tylko dla metod set/getconnectionNameList, które mogą obsługiwać listę nazw połączeń. Metody, takie jak get/setHostname , które nie obsługują list nazw, uzyskują dostęp do pierwszej nazwy z listy.

Po nawiązaniu połączenia automatycznie ponownie połączalne połączenia klienta stają się ponownie połączalne.

To, czy aplikacja nadal działa poprawnie po ponownym połączeniu, zależy od jego projektu. Zapoznaj się z tematami pokrewnymi, aby dowiedzieć się, jak projektować klienty z możliwością ponownego połączenia. Niektóre istniejące klienty mogą działać poprawnie bez modyfikacji po automatycznym ponownym połączeniu.

Klasy WebSphere MQ classes for Java nie obsługują automatycznego nawiązywania ponownego połączenia przez klient.

W celu uniknięcia ponownego połączenia wszystkich klientów przyłączonych do menedżera kolejek, które nie powiodły się, próby ponownego połączenia są opóźniane z częstotliwością częściowo stałą, a częściowo losową.

Domyślnie próby ponownego połączenia są podejmowane w następujących odstępach czasu:

1. Pierwsza próba jest podejmowana po początkowym opóźnieniu jednej sekundy, plus losowy element nawet do 250 milisekund.
2. Druga próba to dwie sekundy, plus przypadkowy odstęp czasu do 500 milisekund, po pierwszej próbie awarii.
3. Trzecia próba to cztery sekundy, plus losowy odstęp czasu do jednej sekundy, po drugiej próbie nie powiedzie się.
4. Czwarta próba to osiem sekund, plus przypadkowy odstęp czasu do dwóch sekund, po trzeciej próbie awarii.
5. Piąta próba jest wykonana 16 sekund, plus przypadkowy odstęp czasu do czterech sekund, po czwartej próbie nie udaje się.
6. Szósta próba, a wszystkie kolejne próby są wykonane 25 sekund, plus przypadkowy odstęp czasu do sześciu sekund i 250 milisekund, po poprzedniej próbie awarii.

Proces ponownego połączenia jest kontynuowany, dopóki klient nie zostanie pomyślnie ponownie podłączony do menedżera kolejek lub do momentu, gdy upłynie maksymalny przedział czasu ponownego połączenia.

Jeśli konieczne jest zwiększenie wartości domyślnych, w celu dokładniejszego odzwierciedlenia czasu wymaganego do odtworzenia menedżera kolejek lub aktywowania rezerwowego menedżera kolejek, należy zmienić wartości opóźnienia w MQCLIENT.INI , korzystając z atrybutu **ReconDelay** .

Pojęcia pokrewne

Zautomatyzowane ponowne połączenie klienta

Zadania pokrewne

Konfigurowanie klienta przy użyciu pliku konfiguracyjnego

Współużytkowanie połączenia TCP/IP w produkcie IBM WebSphere MQ classes for JMS

W celu współużytkowania pojedynczego połączenia TCP/IP można utworzyć wiele instancji kanału MQI.

Aplikacje działające w tym samym środowisku wykonawczym Java, które używają adaptera zasobów IBM WebSphere MQ classes for JMS lub IBM WebSphere MQ do nawiązywania połączenia z menedżerem kolejek przy użyciu transportu CLIENT, mogą być udostępniane do współużytkowania tej samej instancji kanału.

Między instancjami kanału a połączeniami TCP/IP istnieje relacja jeden do jednego. Dla każdej instancji kanału tworzone jest jedno połączenie TCP/IP.

Jeśli kanał jest zdefiniowany z parametrem **SHARECNV** ustawionym na wartość większą niż 1, to ta liczba konwersacji może współużytkować instancję kanału. Aby włączyć fabrykę połączeń lub specyfikację aktywowania w celu użycia tej funkcji, należy ustawić właściwość **SHARECONVALLOWED** na wartość YES.

Każde połączenie JMS i sesja JMS, które są tworzone przez aplikację JMS, tworzy własną konwersację z menedżerem kolejek.

Po uruchomieniu specyfikacji aktywowania klasa IBM WebSphere MQ dla adaptera zasobów JMS rozpoczyna konwersację z menedżerem kolejek w celu użycia specyfikacji aktywowania. Każda sesja serwera w puli sesji serwera, która jest powiązana ze specyfikacją aktywowania, rozpoczyna również konwersację z menedżerem kolejek.

Atrybut SHARECNV jest najlepszym sposobem podejścia do współużytkowania połączeń. W związku z tym, jeśli wartość SHARECNV większa niż 0 jest używana z klasami IBM WebSphere MQ classes for JMS, nie ma gwarancji, że nowe żądanie połączenia zawsze będzie współużytkował nawiązane już połączenie.

Obliczanie liczby instancji kanału

Aby określić maksymalną liczbę instancji kanałów utworzonych przez aplikację, należy użyć następujących formuł:

Specyfikacje aktywowania

Number of channel instances = (<**maxPoolDepth**> + 1)/<**SHARECNV**>

Gdzie <**maxPoolDepth**> to wartość właściwości **maxPoolDepth** , a <**SHARECNV**> to wartość właściwości **SHARECNV** w kanale, która jest używana przez specyfikację aktywowania.

Inne aplikacje JMS

Liczba instancji kanału = (< połączenia JMS > + < nazwa_sesji_JMS >)/<**SHARECNV**>

Gdzie < połączenia JMS > to liczba połączeń utworzonych przez aplikację, gdzie < sesje JMS > to liczba sesji JMS utworzonych przez aplikację, a <**SHARECNV**> to wartość właściwości **SHARECNV** w kanale, która jest używana przez specyfikację aktywowania.

Przykłady

W poniższych przykładach przedstawiono sposób użycia formuł w celu obliczenia liczby instancji kanału, które są tworzone w menedżerze kolejek przez aplikacje przy użyciu adaptera zasobów IBM WebSphere MQ classes for JMS lub IBM WebSphere MQ classes for JMS .

Przykład aplikacji JMS

Połączenie aplikacji JMS łączy się z menedżerem kolejek przy użyciu transportu CLIENT i tworzy połączenie JMS i trzy sesje JMS. Kanał, za pomocą którego aplikacja używa do połączenia z menedżerem kolejek, ma właściwość **SHARECNV** ustawioną na wartość 10. Gdy aplikacja jest uruchomiona, między aplikacją a menedżerem kolejek i jedną instancją kanału są dostępne cztery konwersacje. Wszystkie cztery konwersacje współużytkują instancję kanału.

Przykład specyfikacji aktywowania

Specyfikacja aktywowania łączy się z menedżerem kolejek przy użyciu transportu CLIENT. Specyfikacja aktywowania jest skonfigurowana z właściwością **maxPoolDepth** ustawioną na wartość 10. Kanał, którego konfiguracja aktywowania jest skonfigurowana do użycia, ma właściwość **SHARECNV** ustawioną na wartość 10. Jeśli specyfikacja aktywowania jest uruchomiona i jednocześnie przetwarza 10 komunikatów, liczba konwersacji między specyfikacją aktywowania a menedżerem kolejek wynosi 11 (10 konwersacji dla sesji serwera, a jedna dla specyfikacji aktywowania). Liczba instancji kanału, które są używane przez specyfikację aktywowania to 2.

Przykład specyfikacji aktywowania

Specyfikacja aktywowania łączy się z menedżerem kolejek przy użyciu transportu CLIENT. Specyfikacja aktywowania jest skonfigurowana z właściwością **maxPoolDepth** ustawioną na wartość 5. Kanał, który jest skonfigurowany do używania specyfikacji aktywowania, ma właściwość **SHARECNV** ustawioną na 0. Gdy specyfikacja aktywowania jest uruchomiona i jednocześnie przetwarza 5 komunikatów, liczba konwersacji między specyfikacją aktywowania a menedżerem kolejek wynosi 6 (pięć konwersacji dla sesji serwera, a jedna dla specyfikacji aktywowania). Liczba instancji kanału, które są używane przez specyfikację aktywowania to 6, ponieważ właściwość **SHARECNV** w kanale jest ustawiona na 0, każda konwersacja korzysta z własnej instancji kanału.

Określanie zakresu portów dla połączeń klienckich w klasach produktu WebSphere MQ dla usługi JMS

Użyj właściwości LOCALADDRESS, aby określić zakres portów, z którymi aplikacja może się wiązać.

Gdy klasy produktu WebSphere MQ dla aplikacji JMS podejmują próbę nawiązania połączenia z menedżerem kolejek produktu WebSphere MQ w trybie klienta, firewall może zezwalać tylko na połączenia, które pochodzą z określonych portów lub z zakresu portów. W takiej sytuacji można użyć właściwości LOCALADDRESS obiektu fabryki połączeń ConnectionFactory, QueueConnectionlub obiektu fabryki TopicConnectionw celu określenia portu lub zakresu portów, z którymi aplikacja może się wiązać.

Właściwość LOCALADDRESS można ustawić za pomocą narzędzia administracyjnego JMS produktu WebSphere MQ lub wywołując metodę setLocalAddress () w aplikacji JMS. Poniżej przedstawiono przykład ustawiania właściwości z poziomu aplikacji:

```
mqConnectionFactory.setLocalAddress("192.0.2.0(2000,3000)");
```

Gdy aplikacja nawiąże połączenie z menedżerem kolejek, aplikacja wiąże się z lokalnym adresem IP i numerem portu z zakresu od 192.0.2.0(2000) do 192.0.2.0(3000).

W systemie z więcej niż jednym interfejsem sieciowym można również użyć właściwości LOCALADDRESS, aby określić, który interfejs sieciowy musi być używany dla połączenia.

W przypadku połączenia w czasie rzeczywistym z brokerem właściwość LOCALADDRESS ma znaczenie tylko wtedy, gdy używana jest funkcja rozsyłania grupowego. W takim przypadku można użyć tej właściwości w celu określenia, który lokalny interfejs sieciowy musi być używany dla połączenia, ale wartość właściwości nie może zawierać numeru portu ani zakresu numerów portów.

W przypadku ograniczenia zakresu portów mogą wystąpić błędy połączenia. Jeśli wystąpi błąd, zgłaszany jest wyjątek JMSException z osadzonym wyjątkiem MQException, który zawiera kod przyczyny WebSphere MQ MQRC_Q_MGR_NOT_AVAILABLE i następujący komunikat:

```
Próba nawiązania połączenia przez gniazdo nie powiodła się ze względu na ograniczenia właściwości
LOCAL_ADDRESS_PROPERTY
```

Błąd może wystąpić, jeśli używane są wszystkie porty w podanym zakresie lub jeśli podany adres IP, nazwa hosta lub numer portu nie są poprawne (na przykład jest to ujemny numer portu).

Ponieważ klasy WebSphere MQ classes for JMS mogą tworzyć połączenia inne niż te, które są wymagane przez aplikację, należy zawsze rozważyć określenie zakresu portów. Na ogół każda sesja utworzona przez aplikację wymaga jednego portu, a klasy produktu WebSphere MQ dla usługi JMS mogą wymagać trzech lub czterech dodatkowych portów. Jeśli wystąpi błąd połączenia, należy zwiększyć zakres portów.

Zestawianie połączeń, które jest używane domyślnie w klasach produktu WebSphere MQ dla usługi JMS, może mieć wpływ na szybkość, z jaką porty mogą być ponownie wykorzystywane. W związku z tym może wystąpić błąd połączenia, gdy porty są zwalniane.

Kompresja kanału w klasach produktu WebSphere MQ dla usługi JMS

Klasy produktu WebSphere MQ dla aplikacji JMS mogą używać narzędzi WebSphere MQ do kompresowania nagłówka komunikatu lub danych.

Kompresowanie danych, które przepływa przez kanał WebSphere MQ , może poprawić wydajność kanału i zmniejszyć ruch w sieci. Za pomocą funkcji dostarczonej z produktem WebSphere MQmożna kompresować dane, które przepływają w kanałach komunikatów i kanałach MQI. W każdym z tych typów kanałów można kompresować dane nagłówka i dane komunikatów niezależnie od siebie. Domyślnie żadne dane nie są kompresowane w kanale.

Aplikacja klasy WebSphere MQ dla aplikacji JMS określa techniki, które mogą być używane do kompresowania nagłówka lub danych komunikatu w połączeniu przez utworzenie obiektu java.util.Collection . Każda technika kompresji jest obiektem typu Integer w kolekcji, a kolejność, w jakiej aplikacja dodaje techniki kompresji do kolekcji, jest to kolejność, w jakiej techniki kompresji są negocjowane z menedżerem kolejek, gdy aplikacja tworzy połączenie. Aplikacja może następnie przekazać kolekcję do obiektu ConnectionFactory , wywołując metodę setHdrCompList(), dla danych nagłówka lub metodę setMsgCompList(), w celu uzyskania danych komunikatu. Gdy aplikacja jest gotowa, może utworzyć połączenie.

Opisane poniżej fragmenty kodu ilustrują opisane podejście. Pierwszy fragment kodu pokazuje, jak zaimplementować kompresję danych nagłówka:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(WMQConstants.WMQ_COMPHDR_SYSTEM));
.
.
.
((MQConnectionFactory) cf).setHdrCompList(headerComp);
.
.
.
connection = cf.createConnection();
```

Drugi fragment kodu przedstawia sposób implementowania kompresji danych komunikatu:

```
Collection msgComp = new Vector();
msgComp.add(new Integer(WMQConstants.WMQ_COMPMSG_RLE));
msgComp.add(new Integer(WMQConstants.WMQ_COMPMSG_ZLIBHIGH));
.
.
.
((MQConnectionFactory) cf).setMsgCompList(msgComp);
.
.
.
connection = cf.createConnection();
```

W drugim przykładzie techniki kompresji są negocjowane w kolejności RLE, a następnie ZLIBHIGH, kiedy to połączenie jest tworzone. Wybrana technika kompresji nie może zostać zmieniona w czasie życia obiektu połączenia. Aby można było używać kompresji w połączeniu, przed utworzeniem obiektu połączenia należy wywołać metodę setHdrCompList() i metody setMsgCompList().

Asynchronicznie umieszczanie komunikatów w klasach produktu IBM WebSphere MQ dla usługi JMS

Zwykle, gdy aplikacja wysyła komunikaty do miejsca docelowego, aplikacja musi czekać na menedżera kolejek, aby upewnić się, że przetworzono żądanie. W niektórych sytuacjach wydajność przesyłania komunikatów można zwiększyć, wybierając opcję asynchronicznie umieszczając komunikaty. Gdy aplikacja umieszcza komunikat asynchronicznie, menedżer kolejek nie zwraca powodzenia lub niepowodzenia każdego wywołania, ale można okresowo sprawdzać, czy nie wystąpiły błędy.

Niezależnie od tego, czy miejsce docelowe zwraca sterowanie do aplikacji, bez określania, czy menedżer kolejek odebrał komunikat bezpiecznie, zależy od następujących właściwości:

- Docelowa właściwość JMS PUTSYNCALLOWED (krótka nazwa-PAALD).

PUTASYNCALLOWED określa, czy aplikacje JMS mogą umieszczać komunikaty asynchronicznie, czy kolejka bazowa lub temat, który reprezentuje miejsce docelowe JMS, umożliwia tę opcję.

- Właściwość tematu lub kolejki produktu IBM WebSphere MQ DEFPRESP (Domyślny typ odpowiedzi dla umieszczenia).

  Funkcja DEFPRESP określa, czy aplikacje, które wstawiają komunikaty do kolejki, czy publikują komunikaty w temacie, mogą korzystać z funkcji put asynchronicznego.

W poniższej tabeli przedstawiono możliwe wartości dla właściwości PUTASYNCALLOWED i DEFPRESP oraz jakie wartości są wymagane dla funkcji put asynchronicznego, która ma być włączona:

Tabela 131. Właściwości PUTASYNCALLOWED i DEFPRESP określające, czy komunikaty są umieszczane asynchronicznie.

Właściwość kolejki produktu WebSphere MQ	**PUTASYNCALLOWED = NIE**	**PUTASYNCALLOWED = TAK**	**PUTASYNCALLOWED = AS_DEST lub AS_Q_DEF lub AS_T_DEF**
DEFPRESP=SYNC	Funkcja asynchronicznej funkcji put nie została włączona	Włączona funkcja asynchronicznej funkcji put	Funkcja asynchronicznej funkcji put nie została włączona
DEFPRESP=ASYNC	Funkcja asynchronicznej funkcji put nie została włączona	Włączona funkcja asynchronicznej funkcji put	Włączona funkcja asynchronicznej funkcji put

W przypadku komunikatów wysyłanych w sesji transakcyjnych aplikacja ostatecznie określa, czy menedżer kolejek odebrał komunikaty w sposób bezpieczny, gdy wywołuje program `commit()`.

Jeśli aplikacja wysyła komunikaty trwałe w ramach sesji transakcyjnej, a co najmniej jeden komunikat nie jest odbierany bezpiecznie, transakcja nie zostanie zatwierdzona i zostanie zgłoszony wyjątek. Jeśli jednak aplikacja wysyła nietrwałe komunikaty w ramach sesji transakcyjnej, a co najmniej jeden komunikat nie zostanie odebrany bezpiecznie, transakcja zostanie pomyślnie wykonana. Aplikacja nie otrzymuje żadnych informacji zwrotnych, że komunikaty nietrwałe nie dotarły bezpiecznie.

W przypadku komunikatów nietrwałych wysłanych w sesji, która nie jest transakowana, właściwość SENDCHECKCOUNT obiektu *ConnectionFactory* określa liczbę komunikatów, które mają zostać wysłane, zanim klasy produktu IBM WebSphere MQ dla usługi JMS sprawdzą, czy menedżer kolejek odebrał komunikaty bezpiecznie.

Jeśli sprawdzenie wykryje, że co najmniej jeden komunikat nie został odebrany bezpiecznie, a aplikacja zarejestrowało obiekt nasłuchiwania wyjątków z połączeniem, klasy IBM WebSphere MQ classes for JMS wywołują metodę `onException()` obiektu nasłuchiwania wyjątków w celu przekazania wyjątku JMS do aplikacji.

Wyjątek JMS ma kod błędu JMSWMQ0028 , a ten kod wyświetla następujący komunikat:

```
At least one asynchronous put message failed or gave a warning.
```

Wyjątek JMS ma także powiązany wyjątek, który udostępnia więcej szczegółów. Wartość domyślna właściwości SENDCHECKCOUNT wynosi zero, co oznacza, że nie są wykonywane żadne sprawdzenia.

Ta optymalizacja jest najbardziej korzystna dla aplikacji, która łączy się z menedżerem kolejek w trybie klienta i wymaga wysyłania sekwencji komunikatów w szybkim dziedziczeniu, ale nie wymaga natychmiastowych informacji zwrotnych od menedżera kolejek dla każdego wysłanego komunikatu. Jednak aplikacja może nadal korzystać z tej optymalizacji nawet wtedy, gdy łączy się z menedżerem kolejek w trybie powiązań, ale oczekiwane korzyści z wydajności nie są tak duże.

Korzystanie z odczytu z wyprzedzeniem z klasami produktu WebSphere MQ dla usługi JMS

Funkcja odczytu z wyprzedzeniem udostępniana przez produkt WebSphere MQ umożliwia wysyłanie nietrwałych komunikatów, które są odbierane poza transakcją, do IBM WebSphere MQ classes for JMS przed ich żądaniami. Program IBM WebSphere MQ classes for JMS zapisuje komunikaty w buforze wewnętrznym i przekazuje je do aplikacji, gdy aplikacja zapyta o te komunikaty.

Aplikacje produktu IBM WebSphere MQ classes for JMS , które używają produktu `MessageConsumers` lub `MessageListeners` do odbierania komunikatów z miejsca docelowego poza transakcją, mogą

korzystać z funkcji odczytu z wyprzedzeniem. Funkcja odczytu z wyprzedzeniem umożliwia aplikacjom korzystaniowym z tych obiektów korzystanie z większej wydajności podczas odbierania komunikatów.

To, czy aplikacja, która używa produktu `MessageConsumers` lub `MessageListeners` , może używać odczytu z wyprzedzeniem, zależy od następujących właściwości:

- Docelowa właściwość JMS READAHEADALLOWED (krótka nazwa-RAALD). READAHEADALLOWED określa, czy aplikacje JMS mogą korzystać z odczytu z wyprzedzeniem podczas pobierania lub przeglądania nietrwałych komunikatów poza transakcją, jeśli kolejka bazowa lub temat, który reprezentuje miejsce docelowe JMS, pozwala na korzystanie z tej opcji.
- Kolejka IBM WebSphere MQ lub właściwość tematu DEFREADA (domyślnie odczyt z wyprzedzeniem). DEFREADA określa, czy aplikacje, które odbierają lub przeglądają nietrwałe komunikaty poza transakcją, mogą korzystać z odczytu z wyprzedzeniem.

W poniższej tabeli przedstawiono możliwe wartości dla właściwości READAHEADALLOWED i DEFREADA oraz jakie wartości są wymagane dla funkcji odczytu z wyprzedzeniem, która ma być włączona:

Tabela 132. Właściwości READAHEADALLOWED i DEFREADA określające, czy odczyt z wyprzedzeniem jest używany podczas odbierania lub przeglądania nietrwałych komunikatów poza transakcją.

Właściwość miejsca docelowego produktu WebSphere MQ	**READAHEADALLOWED = YES**	**READAHEADALLOWED = NIE**	**AS_DEST lub AS_Q_DEF lub AS_T_DEF**
Właściwość kolejki produktu WebSphere MQ			
DEFREADA = NIE	Funkcja odczytu z wyprzedzeniem włączona	Funkcja odczytu z wyprzedzeniem nie jest włączona	Funkcja odczytu z wyprzedzeniem nie jest włączona
DEFREADA = TAK	Funkcja odczytu z wyprzedzeniem włączona	Funkcja odczytu z wyprzedzeniem nie jest włączona	Funkcja odczytu z wyprzedzeniem włączona
DEFREADA = WYŁĄCZONE	Funkcja odczytu z wyprzedzeniem nie jest włączona	Funkcja odczytu z wyprzedzeniem nie jest włączona	Funkcja odczytu z wyprzedzeniem nie jest włączona

Jeśli funkcja odczytu z wyprzedzeniem jest włączona, gdy aplikacja `MessageConsumer` lub `MessageListener` jest tworzona przez aplikację, IBM WebSphere MQ classes for JMS tworzy wewnętrzny bufor dla miejsca docelowego, które jest monitorowane przez produkt `MessageConsumer` lub `MessageListener` . Dla każdego `MessageConsumer` lub `MessageListener`istnieje jeden bufor wewnętrzny. Menedżer kolejek uruchamia wysyłanie nietrwałych komunikatów do partycji IBM WebSphere MQ classes for JMS , gdy aplikacja wywołuje jedną z następujących metod:

- `MessageConsumer.receive()`
- `MessageConsumer.receive(long timeout)`
- `MessageConsumer.receiveNoWait()`
- `Session.setMessageListener(MessageListener listener)`

Program IBM WebSphere MQ classes for JMS automatycznie zwraca pierwszy komunikat z powrotem do aplikacji przy użyciu wywołania metody, które zostało wykonane przez aplikację. Inne nietrwałe komunikaty są zapisywane przez IBM WebSphere MQ classes for JMS w wewnętrznym buforze utworzonym dla miejsca docelowego. Gdy aplikacja zażąda przetworzenia następnego komunikatu do przetworzenia, program IBM WebSphere MQ classes for JMS zwróci następny komunikat w buforze wewnętrznym.

Jeśli bufor wewnętrzny jest pusty, program IBM WebSphere MQ classes for JMS żąda bardziej nietrwałych komunikatów z menedżera kolejek.

Bufor wewnętrzny używany przez składnik IBM WebSphere MQ classes for JMS jest usuwany, gdy aplikacja zamknie `MessageConsumer`lub sesja JMS, z którą powiązany jest `MessageListener` .

W przypadku systemu `MessageConsumers`wszystkie nieprzetworzone komunikaty w buforze wewnętrznym są tracone.

W przypadku korzystania z produktu `MessageListeners`, co dzieje się z komunikatami w buforze wewnętrznym, zależy od właściwości miejsca docelowego JMS READAHEADCLOSEPOLICY (nazwa skrócona-RACP). Wartością domyślną właściwości jest DELIVER_ALL, co oznacza, że sesja JMS użyta do utworzenia partycji `MessageListener` nie jest zamknięta, dopóki wszystkie komunikaty w buforze wewnętrznym nie zostaną dostarczone do aplikacji. Jeśli właściwość jest ustawiona na DELIVER_CURRENT, sesja JMS zostanie zamknięta po przetworzeniu bieżącego komunikatu przez aplikację, a wszystkie pozostałe komunikaty w buforze wewnętrznym zostaną usunięte.

Zachowane publikacje w klasach produktu WebSphere MQ dla usługi JMS

Klasy produktu WebSphere MQ dla klienta JMS można skonfigurować w taki sposób, aby używały zachowywanych publikacji.

Publikator może określić, że kopia publikacji musi być zachowana, aby można ją było wysłać do przyszłych subskrybentów, którzy zarejestrują zainteresowanie tematem. Jest to wykonywane w klasach WebSphere MQ classes for JMS, ustawiając właściwość całkowitoliczbową JMS_IBM_RETAIN na wartość 1. Stałe zostały zdefiniowane dla tych wartości w interfejsie com.ibm.msg.client.jms.JmsConstants . Na przykład, jeśli został utworzony komunikat *msg*, aby ustawić go jako zachowaną publikację, należy użyć następującego kodu:

```
// set as a retained publication
msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION);
```

Teraz można wysłać wiadomość w normalny sposób. Zapytanie JMS_IBM_RETAIN może być również wysyłane w odebranym komunikacie. W związku z tym możliwe jest sprawdzenie, czy odebrany komunikat jest zachowaną publikacją.

Obsługa interfejsu XA w klasach produktu WebSphere MQ dla usługi JMS

Usługa JMS obsługuje transakcje zgodne z interfejsem XA w powiązaniach i trybach klienta z obsługiwanym menedżerem transakcji.

Jeśli wymagane jest korzystanie z funkcji XA w środowisku serwera aplikacji, należy odpowiednio skonfigurować aplikację. Informacje na temat sposobu konfigurowania aplikacji do korzystania z transakcji rozproszonych można znaleźć w dokumentacji własnej serwera aplikacji.

Korzystanie z połączenia w czasie rzeczywistym z brokerem produktu WebSphere Event Broker lub WebSphere Message Broker

Klasy produktu WebSphere MQ dla aplikacji JMS mogą korzystać z połączenia w czasie rzeczywistym z brokerem produktu WebSphere Event Broker lub WebSphere Message Broker na potrzeby przesyłania komunikatów w trybie publikowania/subskrypcji. Zarówno broker, jak i klasy produktu WebSphere MQ dla usługi JMS muszą być skonfigurowane w taki sposób, aby umożliwić połączenie w czasie rzeczywistym.

Gdy aplikacja korzysta z połączenia w czasie rzeczywistym z brokerem produktu WebSphere Event Broker lub WebSphere Message Broker, to aplikacja i komunikaty wymiany brokera są wysyłane za pomocą produktu WebSphere MQ Real-Time Transport. W zależności od konfiguracji komunikaty mogą być również dostarczane do aplikacji przy użyciu produktu WebSphere MQ Multicast Transport.

Informacje na temat sposobu nawiązywania połączenia aplikacji z menedżerem kolejek produktu WebSphere MQ i używania produktu WebSphere MQ Enterprise Transport w celu wymiany komunikatów z brokerem produktu WebSphere Event Broker lub WebSphere Message Broker, Zapoznaj się z dokumentacją poprzednich wersji klas WebSphere MQ dla usługi JMS. Należy zwrócić uwagę, że aby można było używać produktu WebSphere MQ Enterprise Transport, aplikacja musi łączyć się z menedżerem kolejek przy użyciu fabryki połączeń działającej w trybie migracji dostawcy przesyłania komunikatów produktu WebSphere MQ .

Konfigurowanie brokera produktu WebSphere Event Broker lub WebSphere Message Broker na potrzeby połączenia w czasie rzeczywistym

W przypadku klas WebSphere MQ dla aplikacji JMS w celu użycia połączenia w czasie rzeczywistym z brokerem produktu WebSphere Event Broker lub WebSphere Message Broker należy skonfigurować broker, tworząc i wdrażając przepływ komunikatów w celu odczytania komunikatów z portu TCP/IP, na którym broker nasłuchuje i publikuje komunikaty. W zależności od wymagań może być konieczne skonfigurowanie brokera na dodatkowe sposoby.

Aby skonfigurować broker, należy utworzyć i wdrożyć jeden z następujących przepływów komunikatów:

- Przepływ komunikatów, który zawiera węzeł przetwarzania komunikatów przepływu rzeczywistegotimeOptimized.
- Przepływ komunikatów, który zawiera węzeł przetwarzania komunikatów Real-timeInput i węzeł przetwarzania komunikatów publikacji.

Należy skonfigurować węzeł Real-timeOptimizedFlow lub Real-timeInput , aby nasłuchiwać na porcie TCP/IP używanym do połączeń w czasie rzeczywistym. Domyślnie numerem portu dla połączeń w czasie rzeczywistym jest 1506.

Broker musi również zostać skonfigurowany, jeśli użytkownik ma dowolne z następujących wymagań:

- Jeśli aplikacja ma łączyć się z brokerem przy użyciu protokołu SSL (Secure Sockets Layer)
- Jeśli aplikacja ma łączyć się z brokerem przy użyciu tunelowania HTTP
- Jeśli komunikaty mają być dostarczane do konsumenta komunikatów przy użyciu rozsyłania grupowego

Informacje na temat konfigurowania brokera można znaleźć w sekcji *Dokumentacja produktu WebSphere Event Broker* lub *Dokumentacja produktu WebSphere Message Broker*.

Konfigurowanie klas produktu WebSphere MQ dla usługi JMS na potrzeby połączenia w czasie rzeczywistym z brokerem produktu WebSphere Event Broker lub WebSphere Message Broker

Aby aplikacja WebSphere MQ classes for JMS używała połączenia w czasie rzeczywistym z brokerem produktu WebSphere Event Broker lub WebSphere Message Broker, klasy WebSphere MQ classes for JMS należy skonfigurować, ustawiając określone właściwości fabryki połączeń. W zależności od wymagań, klasy produktu WebSphere MQ dla usługi JMS mogą wymagać skonfigurowania na dodatkowe sposoby.

Aby skonfigurować klasy produktu WebSphere MQ dla usługi JMS, muszą zostać ustawione następujące właściwości fabryki połączeń:

- Właściwość TRANSPORT musi być ustawiona na wartość DIRECT.

  Jednak w przypadku aplikacji w celu nawiązania połączenia przy użyciu tunelowania HTTP właściwość TRANSPORT musi być ustawiona na wartość DIRECTHTTP. Patrz “Korzystanie z tunelowania HTTP” na stronie 952.
- Właściwość HOSTNAME musi być ustawiona na nazwę hosta lub adres IP systemu, na którym jest uruchomiony broker.
- Właściwość PORT musi być ustawiona na wartość numeru portu, na którym broker nasłuchuje połączeń w czasie rzeczywistym.

Aplikacja może ustawić te właściwości dynamicznie w czasie wykonywania, korzystając z rozszerzeń IBM JMS lub rozszerzeń JMS produktu WebSphere MQ . Alternatywnie, jeśli fabryka połączeń jest administrowany obiekt, administrator może ustawić te właściwości za pomocą narzędzia administracyjnego WebSphere MQ JMS lub programu WebSphere MQ Explorer.

Więcej informacji na temat właściwości oraz metod używanych przez aplikacje do ustawiania ich wartości zawiera sekcja Właściwości obiektów produktu IBM WebSphere MQ classes for JMS. Więcej informacji na temat korzystania z narzędzia administracyjnego WebSphere MQ JMS zawiera sekcja “Korzystanie z narzędzia administracyjnego JMS produktu WebSphere MQ” na stronie 960. Informacje na temat korzystania z programu WebSphere MQ Explorer można znaleźć w pomocy dostarczonej z programem WebSphere MQ Explorer.

Jeśli użytkownik ma dowolne z następujących wymagań, klasy produktu WebSphere MQ dla usługi JMS wymagają dodatkowej konfiguracji:

- Jeśli aplikacja ma łączyć się z brokerem przy użyciu protokołu SSL (Secure Sockets Layer)
- Jeśli aplikacja ma łączyć się z brokerem przy użyciu tunelowania HTTP
- Jeśli aplikacja ma łączyć się z brokerem za pośrednictwem serwera proxy
- Jeśli komunikaty mają być dostarczane do konsumenta komunikatów przy użyciu rozsyłania grupowego

W poniższych sekcjach opisano sposób konfigurowania klas produktu WebSphere MQ dla usługi JMS dla każdego z tych wymagań.

Korzystanie z uwierzytelniania SSL (Secure Sockets Layer)

Uwierzytelnianie SSL może być używane w czasie rzeczywistym do połączenia z brokerem. Dla tego typu połączenia obsługiwane jest tylko uwierzytelnianie. Za pomocą protokołu SSL nie można szyfrować i deszyfrować danych komunikatów, które przepływają między aplikacją a brokerem lub w celu wykrycia ingerowania w te dane.

Należy zwrócić uwagę na różnicę między tą sytuacją a sytuacją, w której aplikacja łączy się z menedżerem kolejek w trybie klienta. W tym drugim przypadku można użyć obsługi protokołu SSL produktu WebSphere MQ do szyfrowania i deszyfrowania danych komunikatów, które przepływają między aplikacją a menedżerem kolejek oraz do wykrywania manipulacji danymi, a także do udostępniania uwierzytelniania.

Aby zabezpieczyć dane komunikatu w czasie rzeczywistym połączenia z brokerem, można użyć funkcji udostępnionej przez broker. Do każdego tematu z komunikatami, które mają być chronione, można przypisać wartość ochrony (QoP). W związku z tym można wybrać inny poziom ochrony komunikatów dla każdego tematu. Więcej informacji na temat zabezpieczeń komunikatów udostępnianych przez brokera można znaleźć w sekcji *Dokumentacja produktu WebSphere Event Broker* lub *Dokumentacja produktu WebSphere Message Broker*.

Aby używać uwierzytelniania SSL w czasie rzeczywistym do połączenia z brokerem, właściwość DIRECTAUTH fabryki połączeń musi być ustawiona na CERTIFICATE.

Jeśli do uwierzytelniania wzajemnego ma być używany protokół SSL, właściwość Typ protokołu uwierzytelniania brokera musi określać opcję R dla symetrycznego protokołu SSL. Jeśli do uwierzytelniania brokera ma być używany tylko protokół SSL, właściwość Typ protokołu uwierzytelniania brokera musi określać opcję S dla asymetrycznego protokołu SSL. Jednak w tym przypadku aplikacja musi łączyć się z brokerem, wywołując metodę createConnection() z identyfikatorem użytkownika i hasłem jako parametrami, jak w następującym przykładzie:

```
factory.createConnection("user1", "user1pw");
```

Broker używa następnie identyfikatora użytkownika i hasła, a nie SSL, w celu uwierzytelnienia aplikacji. Więcej informacji na temat konfigurowania brokera do uwierzytelniania SSL zawiera publikacja *Dokumentacja produktu WebSphere Event Broker* lub *Dokumentacja produktu WebSphere Message Broker*.

Uwagi:

1. Wartość właściwości DIRECTAUTH określa, czy uwierzytelnianie SSL jest używane w czasie rzeczywistym do połączenia z brokerem, a nie wartość właściwości SSLCIPHERSUITE.
2. Jeśli uwierzytelnianie SSL jest używane w połączeniu w czasie rzeczywistym do brokera, właściwości SSLPEERNAME i SSLCRL są używane do wykonywania tych samych operacji sprawdzania, co te wykonywane, gdy aplikacja łączy się z menedżerem kolejek w trybie klienta.
3. Klasy WebSphere MQ classes for JMS mogą używać tej samej konfiguracji magazynu kluczy i magazynu zaufanych certyfikatów Java Secure Socket Extension (JSSE), aby zapewnić obsługę protokołu SSL w jednej z następujących sytuacji:
   - Gdy aplikacja korzysta z połączenia w czasie rzeczywistym z brokerem
   - Gdy aplikacja łączy się z menedżerem kolejek w trybie klienta

Korzystanie z tunelowania HTTP

Klasy produktu WebSphere MQ dla aplikacji JMS mogą łączyć się z brokerem przy użyciu tunelowania HTTP, co oznacza, że aplikacja łączy się z brokerem przy użyciu protokołu HTTP, jak w przypadku łączenia się z serwisem WWW.

Aby można było używać tunelowania HTTP w czasie rzeczywistym połączenia z brokerem, właściwość TRANSPORT fabryki połączeń musi być ustawiona na wartość DIRECTHTTP.

Tunelowanie HTTP nie może być używane w połączeniu z uwierzytelnianiem SSL, nawiązywaniem połączenia przez serwer proxy lub dostarczaniem komunikatów przy użyciu rozsyłania grupowego. Obsługiwana wersja protokołu HTTP to 1.0. Wersja HTTP 1.1 nie jest obsługiwana.

Nawiązanie połączenia przez serwer proxy

Klasy produktu WebSphere MQ dla aplikacji JMS mogą korzystać z połączenia w czasie rzeczywistym z brokerem, łącząc się za pośrednictwem serwera proxy. Klasy WebSphere MQ classes for JMS łączy się bezpośrednio z serwerem proxy i używa protokołu internetowego zdefiniowanego w dokumencie RFC 2817, aby zażądać od serwera proxy przesłania żądania połączenia do brokera.

Aby połączyć się z brokerem przez serwer proxy, należy ustawić następujące właściwości fabryki połączeń:

- Właściwość PROXYHOSTNAME musi być ustawiona na nazwę hosta lub adres IP systemu, na którym działa serwer proxy.
- Właściwość PROXYPORT musi być ustawiona na wartość numeru portu, na którym nasłuchuje serwer proxy.

Jeśli właściwość PROXYHOSTNAME nie jest ustawiona lub jest ustawiona na pusty łańcuch, klasy WebSphere MQ classes for JMS podejmą próbę nawiązania bezpośredniego połączenia z brokerem przy użyciu tylko właściwości HOSTNAME i PORT i nie podejmuje próby nawiązania połączenia za pośrednictwem serwera proxy.

Dostarczanie komunikatów przy użyciu rozsyłania grupowego

Korzystając z połączenia w czasie rzeczywistym z brokerem, komunikaty mogą być dostarczane do konsumenta komunikatów przy użyciu rozsyłania grupowego.

Aby włączyć rozsyłanie grupowe, właściwość MULTICAST obiektu tematu musi być ustawiona na wymaganą opcję rozsyłania grupowego. Alternatywnie, jeśli właściwość MULTICAST obiektu tematu jest ustawiona na wartość ASCF, właściwość MULTICAST fabryki połączeń musi być ustawiona na wymaganą opcję rozsyłania grupowego.

Klasy WebSphere MQ classes for JMS obsługują protokoły rozsyłania grupowego (Packet Transfer Layer-PTL) i Pragmatic General Multicast (PGM), a także obsługują zarówno implementacje protokołów PGM, PGM/IP i PGM UDP hermetyzowanych. Jednak obsługa PGM/IP jest dostępna tylko na następujących platformach:

- AIX (tylko wersja 32-bitowa)
- Linux (platformax86)
- Linux (platforma zSeries , tylko 32-bitowa)
- Solaris SPARC (tylko 32-bit)
- Windows (tylko wersja 32-bitowa)
- z/OS

Klasy produktu WebSphere MQ dla obiektów serwera aplikacji JMS

W tym temacie opisano, w jaki sposób klasy WebSphere MQ classes for JMS implementują klasę ConnectionConsumer oraz zaawansowaną funkcjonalność w klasie Session. Podsumowuje on również funkcję puli sesji serwera.

Klasy WebSphere MQ classes for JMS obsługują narzędzia ASF (Application Server Facilities) określone w specyfikacji *Java Message Service Specification, wersja 1.1* (patrz serwis WWW Sun w serwisie WWW https://java.sun.com). Ta specyfikacja identyfikuje trzy role w ramach tego modelu programowania:

- **Dostawca JMS** udostępnia funkcje ConnectionConsumer i zaawansowane funkcje sesji.
- **Serwer aplikacji** udostępnia funkcje puli ServerSessionoraz funkcji ServerSession .
- **Aplikacja kliencka** korzysta z funkcji dostarczanej przez dostawcę JMS i serwer aplikacji.

Informacje zawarte w tym temacie nie mają zastosowania, jeśli aplikacja korzysta z połączenia w czasie rzeczywistym z brokerem.

Interfejs JMS ConnectionConsumer

Interfejs ConnectionConsumer udostępnia wysokowydajną metodę dostarczania komunikatów współbieżnie do puli wątków.

Specyfikacja JMS umożliwia serwerowi aplikacji ścisłą integrację z implementacją JMS przy użyciu interfejsu `ConnectionConsumer` . Ten składnik udostępnia współbieżne przetwarzanie komunikatów. Zwykle serwer aplikacji tworzy pulę wątków, a implementacja JMS udostępnia komunikaty do tych wątków. Serwer aplikacji zorientowany na usługi JMS (na przykład serwer WebSphere Application Server) może korzystać z tej funkcji w celu udostępnienia funkcji przesyłania komunikatów wysokiego poziomu, takich jak komponenty bean sterowane komunikatami.

W normalnych aplikacjach nie jest używany obiekt ConnectionConsumer, ale mogą być używane przez klientów JMS typu expert. W przypadku takich klientów ConnectionConsumer udostępnia wysokowydajną metodę dostarczania komunikatów współbieżnie do puli wątków. Gdy komunikat dociera do kolejki lub tematu, usługa JMS wybiera wątek z puli i dostarcza do niego partię komunikatów. W tym celu usługa JMS uruchamia powiązaną metodę `onMessage()` obiektu MessageListener.

Ten sam efekt można osiągnąć, konstruując wiele obiektów sesji i obiektów MessageConsumer , a każdy z nich ma zarejestrowaną wartość MessageListener. Jednak ConnectionConsumer zapewnia lepszą wydajność, mniejsze wykorzystanie zasobów i większą elastyczność. W szczególności wymagane jest mniej obiektów sesji.

Planowanie aplikacji z ASF

W tej sekcji opisano, jak zaplanować aplikację, w tym:

- “Ogólne zasady przesyłania komunikatów w trybie punkt z punktem przy użyciu narzędzia ASF” na stronie 953
- “Ogólne zasady przesyłania komunikatów w trybie publikowania/subskrypcji przy użyciu narzędzia ASF” na stronie 954
- “Usuwanie komunikatów z kolejki w ASF” na stronie 955
- Obsługa komunikatów nieprzetwarzalnych w ASF. Patrz “Obsługa komunikatów trujących w produkcie IBM WebSphere MQ classes for JMS” na stronie 914.

Ogólne zasady przesyłania komunikatów w trybie punkt z punktem przy użyciu narzędzia ASF

Ten temat zawiera ogólne informacje na temat przesyłania komunikatów w trybie punkt z punktem przy użyciu narzędzia ASF.

Gdy aplikacja tworzy obiekt ConnectionConsumer z obiektu QueueConnection , określa on obiekt kolejki JMS oraz łańcuch selektora. Następnie element ConnectionConsumer rozpoczyna udostępnianie komunikatów do sesji w powiązanej puli ServerSession. Komunikaty docierają do kolejki, a jeśli są zgodne z selektorem, są dostarczane do sesji w powiązanej puli ServerSession.

W terminach WebSphere MQ obiekt kolejki odwołuje się do QLOCAL lub QALIAS w lokalnym menedżerze kolejek. Jeśli jest to QALIAS, to QALIAS musi odwoływać się do QLOCAL. W pełni rozpoznany produkt WebSphere MQ QLOCAL jest znany jako *bazowy QLOCAL*. Parametr ConnectionConsumer ma wartość *active* (aktywny), jeśli nie jest zamknięty, a jego element nadrzędny QueueConnection jest uruchomiony.

Możliwe jest, aby wiele ConnectionConsumers, każdy z różnymi selektorami, było uruchamiane względem tego samego bazowego systemu QLOCAL. Aby zachować wydajność, niepożądane komunikaty nie mogą być gromadzone w kolejce. Niechciane komunikaty to te, dla których żaden aktywny element ConnectionConsumer nie ma zgodnego selektora. Fabrykę QueueConnectionmożna ustawić w taki sposób, aby te niechciane komunikaty zostały usunięte z kolejki (szczegółowe informacje zawiera sekcja “Usuwanie komunikatów z kolejki w ASF” na stronie 955). To zachowanie można ustawić na jeden z dwóch sposobów:

- Użyj narzędzia administracyjnego JMS, aby ustawić fabrykę QueueConnectionna wartość MRET (NO).
- W programie należy użyć:

```
MQQueueConnectionFactory.setMessageRetention(WMQConstants.WMQ_MRET_NO)
```

Jeśli to ustawienie nie zostanie zmienione, wartością domyślną będzie zachowanie takich niechcianych komunikatów w kolejce.

Podczas konfigurowania menedżera kolejek produktu WebSphere MQ należy wziąć pod uwagę następujące kwestie:

- Dla danych wejściowych współużytkowanych musi być włączona bazowa wartość QLOCAL. Aby to zrobić, należy użyć następującej komendy MQSC:

```
ALTER QLOCAL(your.qlocal.name) SHARE GET(ENABLED)
```

- Menedżer kolejek musi mieć włączoną kolejkę niedostarczonych komunikatów. Jeśli obiekt ConnectionConsumer wystąpi z problemem, gdy umieszcza komunikat w kolejce niedostarczonych komunikatów, dostarczenie komunikatu od bazowego zatrzymania QLOCAL zostanie zatrzymane. Aby zdefiniować kolejkę niedostarczonych komunikatów, należy użyć następującej komendy:

```
ALTER QMGR DEADQ(your.dead.letter.queue.name)
```

- Użytkownik, który uruchamia obiekt ConnectionConsumer , musi mieć uprawnienia do wykonywania operacji MQOPEN z MQOO_SAVE_ALL_CONTEXT i MQOO_PASS_ALL_CONTEXT. Szczegółowe informacje na ten temat zawiera dokumentacja produktu WebSphere MQ dla konkretnej platformy.
- Jeśli w kolejce pozostawiane są niechciane komunikaty, pogarszają one wydajność systemu. Dlatego należy zaplanować selektory komunikatów w taki sposób, aby między nimi ConnectionConsumers usunował wszystkie komunikaty z kolejki.

Szczegółowe informacje na temat komend MQSC zawiera sekcja Informacje dodatkowe dotyczące komend MQSC.

Ogólne zasady przesyłania komunikatów w trybie publikowania/subskrypcji przy użyciu narzędzia ASF

ConnectionConsumers odbiera komunikaty dla określonego tematu. Element ConnectionConsumer może być trwały lub nie jest trwały. Należy określić kolejkę lub kolejki, których używa ConnectionConsumer .

Gdy aplikacja tworzy obiekt ConnectionConsumer z obiektu TopicConnection , określa on obiekt tematu i łańcuch selektora. Następnie element ConnectionConsumer rozpoczyna odbieranie komunikatów zgodnych z selektorem w tym temacie, w tym wszystkie zachowane publikacje dotyczące subskrybowanego tematu.

Alternatywnie aplikacja może utworzyć trwałą wartość ConnectionConsumer , która jest powiązana z konkretną nazwą. Ten obiekt ConnectionConsumer odbiera komunikaty, które zostały opublikowane w ramach tematu, ponieważ trwały obiekt ConnectionConsumer był ostatnio aktywny. Odbiera on wszystkie takie komunikaty, które są zgodne z selektorem w temacie. Jeśli jednak obiekt ConnectionConsumer korzysta z funkcji odczytu z wyprzedzeniem, może utracić nietrwałe komunikaty, które znajdują się w buforze klienta po jego zamknięciu.

Jeśli klasy produktu WebSphere MQ classes for JMS znajdują się w trybie migracji dostawcy przesyłania komunikatów produktu WebSphere MQ , dla nietrwałych subskrypcji ConnectionConsumer używana jest oddzielna kolejka.Konfigurowalna opcja CCSUB w fabryce TopicConnectionokreśla kolejkę, która ma być używana. Normalnie CCSUB określa pojedynczą kolejkę do użycia przez wszystkie ConnectionConsumers , które korzystają z tej samej fabryki TopicConnection. Jednak możliwe jest, aby każdy obiekt ConnectionConsumer wygenerował kolejkę tymczasową, określając przedrostek nazwy kolejki, po którym następuje gwiazdka (*).

Jeśli klasy produktu WebSphere MQ classes for JMS znajdują się w trybie migracji dostawcy przesyłania komunikatów produktu WebSphere MQ , właściwość CCDSUB tematu określa kolejkę, która ma być używana dla trwałych subskrypcji. Ponownie może to być kolejka, która już istnieje, lub przedrostek nazwy kolejki, po którym następuje gwiazdka (*). Jeśli zostanie określona kolejka, która już istnieje, wszystkie trwałe elementy ConnectionConsumers , które subskrybują ten temat, używają tej kolejki. Jeśli zostanie określony przedrostek nazwy kolejki, po którym następuje gwiazdka (*), kolejka jest generowana po raz pierwszy, gdy zostanie utworzona trwała wartość ConnectionConsumer o określonej nazwie. Ta kolejka jest ponownie wykorzystywana później, gdy zostanie utworzona trwała wartość ConnectionConsumer o tej samej nazwie.

Podczas konfigurowania menedżera kolejek produktu WebSphere MQ należy wziąć pod uwagę następujące kwestie:

- Menedżer kolejek musi mieć włączoną kolejkę niedostarczonych komunikatów. Jeśli obiekt ConnectionConsumer wystąpi z problemem, gdy umieszcza komunikat w kolejce niedostarczonych komunikatów, dostarczenie komunikatu od bazowego zatrzymania QLOCAL zostanie zatrzymane. Aby zdefiniować kolejkę niedostarczonych komunikatów, należy użyć następującej komendy:

  ```
  ALTER QMGR DEADQ(your.dead.letter.queue.name)
  ```

- Użytkownik, który uruchamia obiekt ConnectionConsumer , musi mieć uprawnienia do wykonywania operacji MQOPEN z MQOO_SAVE_ALL_CONTEXT i MQOO_PASS_ALL_CONTEXT. Szczegółowe informacje można znaleźć w dokumentacji produktu WebSphere MQ dla używanej platformy.
- Wydajność pojedynczego obiektu ConnectionConsumer można zoptymalizować, tworząc osobną, dedykowaną dla niego kolejkę. Jest to koszt dodatkowego wykorzystania zasobów.

Usuwanie komunikatów z kolejki w ASF

Gdy aplikacja korzysta z opcji ConnectionConsumers, może być konieczne usunięcie komunikatów z kolejki w wielu sytuacjach.

Są to następujące sytuacje:

Źle sformatowany komunikat
Może dojść do komunikatu, że usługa JMS nie może przeanalizować.

Wiadomość nieprzetwarzalna
Komunikat może osiągnąć próg wycofania, ale element ConnectionConsumer nie może ponownie umieścić go w kolejce wycofanych komunikatów.

Brak zainteresowania ConnectionConsumer
W przypadku przesyłania komunikatów w trybie punkt z punktem, gdy fabryka QueueConnectionjest ustawiona w taki sposób, że nie zachowuje niechcianych komunikatów, pojawia się komunikat niepożądany przez dowolny z elementów ConnectionConsumers(Złącza połączenia).

W takich sytuacjach element ConnectionConsumer próbuje usunąć komunikat z kolejki. Opcje rozporządzania w polu raportu deskryptora MQMD komunikatu ustawiają dokładne zachowanie. Są to następujące opcje:

MQRO_DEAD_LETTER_Q
Komunikat ten jest ponownie wysyłany do kolejki niedostarczonych komunikatów menedżera kolejek. Jest to opcja domyślna.

MQRO_DISCARD_MSG
Komunikat jest odrzucany.

Opcja ConnectionConsumer generuje również komunikat raportu, a to zależy również od pola raportu MQMD komunikatu. Ten komunikat jest wysyłany do kolejki ReplyTokomunikatu w menedżerze kolejek ReplyTo. Jeśli podczas wysyłania komunikatu o raporcie wystąpi błąd, komunikat jest wysyłany do kolejki niedostarczonych komunikatów. Opcje raportu o wyjątkach w polu raportu w szczegółach MQMD komunikatu ustawiają szczegóły komunikatu raportu. Są to następujące opcje:

MQRO_EXCEPTION
Generowany jest komunikat raportu, który zawiera deskryptor MQMD oryginalnego komunikatu. Nie zawiera żadnych danych treści komunikatu.

MQRO_EXCEPTION_WITH_DATA
Generowany jest komunikat raportu, który zawiera deskryptor MQMD, wszystkie nagłówki MQ i 100 bajtów danych treści.

MQRO_EXCEPTION_WITH_FULL_DATA
Generowany jest komunikat raportu, który zawiera wszystkie dane z oryginalnego komunikatu.

default
Nie jest generowany żaden komunikat raportu.

Gdy generowane są komunikaty raportów, uhonorowane są następujące opcje:

- MQRO_NEW_MSG_ID
- MQRO_PASS_MSG_ID
- MQRO_COPY_MSG_ID_TO_CORREL_ID (Identyfikator CORREL_ID)
- MQRO_PASS_CORREL_ID

Jeśli komunikat nieprzetwarzalny nie może być ponownie wysłany, być może dlatego, że kolejka niedostarczonych komunikatów jest pełna lub autoryzacja została błędnie określona, to co się stanie, zależy od trwałości komunikatu. Jeśli komunikat jest nietrwały, komunikat jest odrzucany i nie jest generowany żaden komunikat raportu. Jeśli komunikat jest trwały, dostarczanie komunikatów do wszystkich konsumentów połączeń nasłuchujących na tym miejscu docelowym jest zatrzymywane. Tego typu połączenia muszą zostać zamknięte, a problem rozwiązany, zanim możliwe będzie ponowne utworzenie i ponowne uruchomienie dostarczania komunikatów.

Ważne jest, aby zdefiniować kolejkę niedostarczonych komunikatów i regularnie sprawdzać, czy nie występują żadne problemy. W szczególności zadbaj o to, aby kolejka niedostarczonych komunikatów nie osiągnęła maksymalnej głębokości, a jej maksymalna wielkość komunikatu jest wystarczająco duża dla wszystkich komunikatów.

Gdy komunikat jest ponownie wysyłany do kolejki niedostarczonych komunikatów, jest on poprzedzony nagłówkiem WebSphere MQ dead-letter (MQDLH). Szczegółowe informacje na temat formatu MQDLH można znaleźć w sekcji MQDLH-Dead-letter header (Nagłówek niewysłanych wiadomości MQDLH). Użytkownik może zidentyfikować komunikaty, które ConnectionConsumer umieli w kolejce niedostarczonych komunikatów, lub komunikaty raportów, które wygenerował ConnectionConsumer , za pomocą następujących pól:

- PutApplTyp: `MQAT_JAVA (0x1C)`
- PutApplNazwa: "`MQ JMS ConnectionConsumer`"

Pola te znajdują się w komunikatach MQDLH komunikatów w kolejce niedostarczonych komunikatów oraz w deskrypcie MQMD komunikatów raportu. Pole informacji zwrotnej deskryptora MQMD oraz pole Przyczyna komunikatu MQDLH zawiera kod opisujący błąd. Szczegółowe informacje na temat tych kodów zawiera sekcja “Kody przyczyny i sprzężenia zwrotnego w ASF” na stronie 957. Inne pola są opisane w sekcji Nagłówek MQDLH-Dead-letter.

Obsługa komunikatów nieprzetwarzalnych w ASF

W obiektach serwera aplikacji obsługa komunikatów nieprzetwarzalnych jest nieco inna niż w innych klasach produktu WebSphere MQ dla usługi JMS.

Więcej informacji na temat obsługi komunikatów nieprzetwarzalnych w klasach WebSphere MQ dla usługi JMS zawiera sekcja “Obsługa komunikatów trujących w produkcie IBM WebSphere MQ classes for JMS” na stronie 914.

Jeśli używane są narzędzia Application Server Facilities (ASF), ConnectionConsumer, a nie MessageConsumer, przetwarza komunikaty nieprzetwarzalne. Właściwości ConnectionConsumer są requiem komunikatów zgodnie z właściwościami QName BackoutThreshold i BackoutRequeuekolejki.

Jeśli aplikacja korzysta z opcji ConnectionConsumers, okoliczności, w których jest tworzona kopia zapasowa komunikatu, zależą od sesji, którą udostępnia serwer aplikacji:

- Gdy sesja jest nietransakowana, z AUTO_ACKNOWLEDGE lub DUPS_OK_ACKNOWLEDGE, komunikat jest wycofany tylko po wystąpieniu błędu systemowego lub nieoczekiwanie kończy działanie aplikacji.
- Gdy sesja nie jest transakowana za pomocą funkcji CLIENT_ACKNOWLEDGE, niepotwierdzone komunikaty mogą być wycofane przez serwer aplikacji wywołując komendę Session.recover().

  Zazwyczaj implementacja klienta MessageListener lub serwer aplikacji wywołuje metodę Message.acknowledge(). Plik Message.acknowledge() potwierdza wszystkie komunikaty dostarczone do tej pory w sesji.
- Gdy sesja jest transacted, niepotwierdzone komunikaty mogą być wycofane przez serwer aplikacji wywołując komendę Session.rollback().
- Jeśli serwer aplikacji dostarcza XASession, komunikaty są zatwierdzane lub wycofane w zależności od transakcji rozproszonej. Serwer aplikacji bierze odpowiedzialność za wykonanie transakcji.

Wbudowany dostawca JMS w produkcie WebSphere Application Server, wersja 5.0 i wersja 5.1 , obsługuje komunikaty nieprzetwarzalne w inny sposób niż opisany w przypadku klas WebSphere MQ dla usługi JMS. Informacje na temat sposobu, w jaki osadzony dostawca JMS obsługuje komunikaty nieprzetwarzalne, można znaleźć w odpowiedniej dokumentacji produktu WebSphere Application Server.

Obsługa błędów

Ta sekcja obejmuje różne aspekty obsługi błędów, w tym “Odtwarzanie po wystąpieniu błędów w ASF” na stronie 957 i “Kody przyczyny i sprzężenia zwrotnego w ASF” na stronie 957.

Odtwarzanie po wystąpieniu błędów w ASF

Jeśli ConnectionConsumer doświadcza poważnego błędu, dostarczenie komunikatu do wszystkich elementów ConnectionConsumers z zainteresowaniem w tych samych zatrzymań QLOCAL. W takiej sytuacji każdy obiekt ExceptionListener , który jest zarejestrowany w dotkniętym połączeniu, jest powiadamiany o tym. Istnieją dwa sposoby, w których aplikacja może odtworzyć dane z tych warunków.

Zwykle poważny błąd o tej naturze występuje wtedy, gdy ConnectionConsumer nie może przekwalifikować komunikatu do kolejki niedostarczonych komunikatów, albo wystąpi błąd podczas odczytywania komunikatów z kolejki QLOCAL.

Ponieważ każdy obiekt ExceptionListener , który jest zarejestrowany w dotkniętym połączeniu, jest powiadamiany, można użyć ich w celu zidentyfikowania przyczyny problemu. W niektórych przypadkach administrator systemu musi interweniować, aby rozwiązać ten problem.

Użyj jednej z następujących technik, aby odtworzyć dane z następujących warunków błędu:

- Wywołaj komendę `close()` dla wszystkich ConnectionConsumers. Aplikacja może utworzyć nowe ConnectionConsumers tylko wtedy, gdy wszystkie ConnectionConsumers zostaną zamknięte, a wszystkie problemy z systemem zostaną rozwiązane.
- Wywołaj `stop()` dla wszystkich połączeń, których dotyczy problem. Po zatrzymaniu wszystkich połączeń i rozwiązaniu ewentualnych problemów z systemem aplikacja może pomyślnie `start()` jej połączenia.

Kody przyczyny i sprzężenia zwrotnego w ASF

Użyj kodów przyczyny i informacji zwrotnych, aby określić przyczynę błędu. W tym miejscu podano wspólne kody przyczyny wygenerowane przez ConnectionConsumer .

Aby określić przyczynę błędu, należy użyć następujących informacji:

- Kod opinii we wszystkich komunikatach raportu
- Kod przyczyny w MQDLH wszystkich komunikatów w kolejce niedostarczonych komunikatów.

ConnectionConsumers generuje następujące kody przyczyny.

MQRC_BACKOUT_THRESHOLD_REACHED (0x93A; 2362)

Przyczyna
Komunikat osiągnął próg wycofania zdefiniowany w systemie QLOCAL, ale nie zdefiniowano kolejki wycofanych komunikatów.

Na platformach, na których nie można zdefiniować kolejki wycofanych komunikatów, komunikat osiągnął próg wycofania zdefiniowany przez JMS na podstawie wartości 20.

Działanie
Jeśli nie jest to wymagane, zdefiniuj kolejkę wycofania dla odpowiedniej kolejki QLOCAL. Poszukaj także przyczyny wielu wycofań.

MQRC_MSG_NOT_MATCHED (0x93B; 2363)

Przyczyna
W przesyłaniu komunikatów w trybie punkt z punktem istnieje komunikat, który nie jest zgodny z żadnym z selektorów dla elementu ConnectionConsumers , który monitoruje kolejkę. Aby zachować wydajność, komunikat jest ponownie wysyłany do kolejki niedostarczonych komunikatów.

Działanie
Aby uniknąć takiej sytuacji, należy upewnić się, że opcja ConnectionConsumers przy użyciu kolejki udostępnia zestaw selektorów, które zajmują się wszystkimi komunikatami, lub ustaw fabrykę QueueConnectionw celu zachowania komunikatów.

Można również zbadać źródło komunikatu.

MQRC_JMS_FORMAT_ERROR (0x93C; 2364)

Przyczyna
Usługa JMS nie może zinterpretować komunikatu w kolejce.

Działanie
Sprawdź pochodzenie komunikatu. Usługa JMS zwykle dostarcza komunikaty o nieoczekiwanym formacie jako BytesMessage lub TextMessage. Zdarza się, że błąd ten nie powiedzie się, jeśli komunikat jest bardzo źle sformatowany.

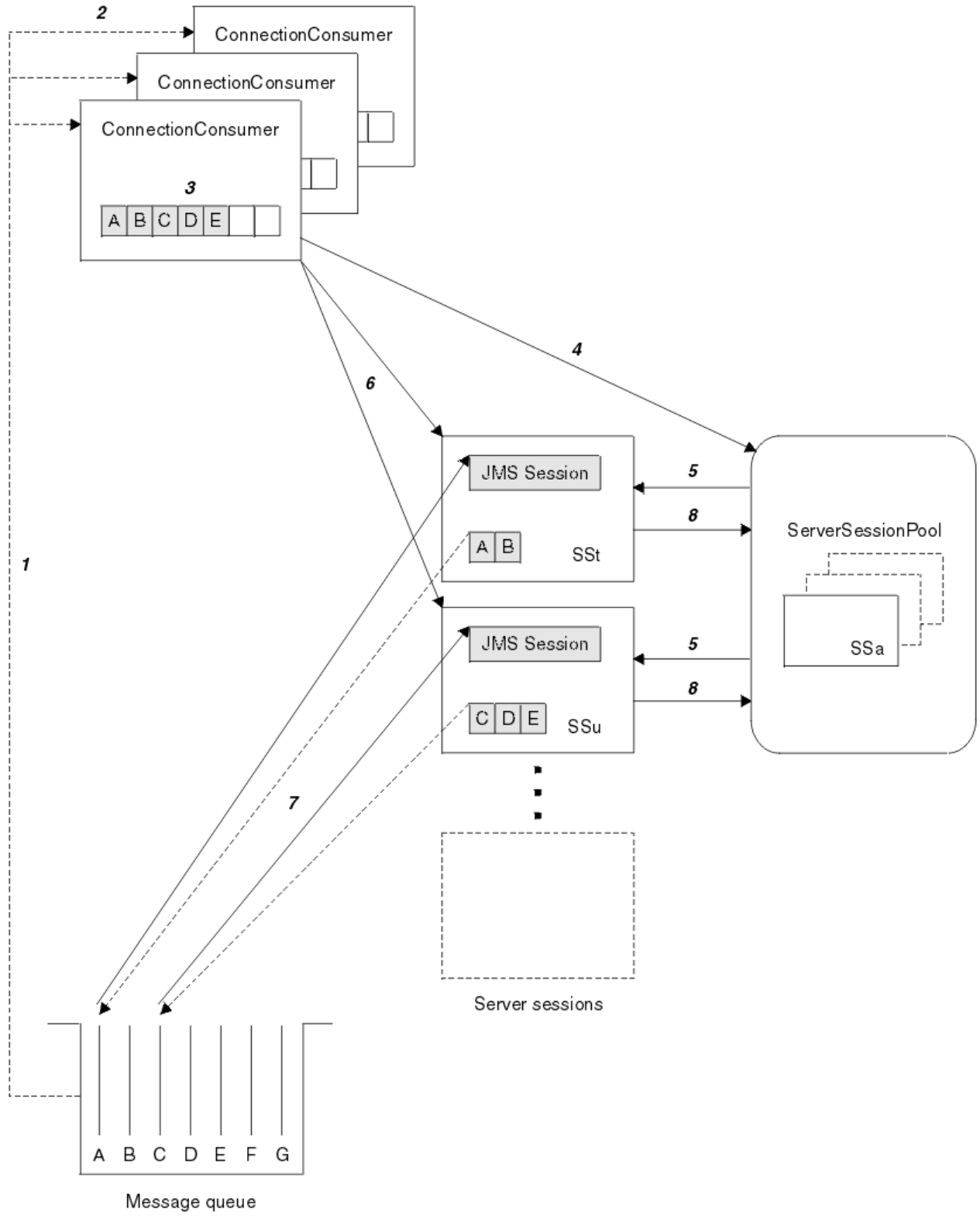
Inne kody, które pojawiają się w tych polach, są spowodowane przez nieudane próby ponownego umieszczenia komunikatu w kolejce wycofania. W tej sytuacji kod opisuje przyczynę niepowodzenia ponownego wykonania żądania. Aby zdiagnozować przyczynę występowania tych błędów, należy zapoznać się z kodami przyczyny funkcji API.

Jeśli komunikat raportu nie może zostać umieszczony w kolejce ReplyTo, jest on umieszczany w kolejce niedostarczonych komunikatów. W tej sytuacji pole sprzężenia zwrotnego deskryptora MQMD zostało zakończone zgodnie z opisem w tym temacie. Pole przyczyny w tabeli MQDLH wyjaśnia, dlaczego komunikat raportu nie może zostać umieszczony w kolejce ReplyTo.

Funkcja puli sesji serwera w systemie AFS

W tej sekcji przedstawiono podsumowanie funkcji puli sesji serwera.

Rysunek 165 na stronie 959 podsumowuje zasady działania puli ServerSessioni funkcji ServerSession .



Rysunek 165. Funkcje ServerSessionPool i ServerSession

1. Element ConnectionConsumers umożliwia pobranie odwołań do komunikatów z kolejki.
2. Każdy element ConnectionConsumer wybiera konkretne odwołania do komunikatów.
3. W buforze ConnectionConsumer znajdują się wybrane odwołania do komunikatów.
4. Opcja ConnectionConsumer żąda co najmniej jednej sesji ServerSessions z puli ServerSession.

5. ServerSessions (Sesje serwera) są przydzielane z puli ServerSession.
6. Obiekt ConnectionConsumer przypisuje odwołania do komunikatu do serwera ServerSessions i uruchamia wątki ServerSession działające.
7. Każda sesja ServerSession pobiera z niej przywoływane komunikaty. Przekazuje je do metody onMessage z obiektu MessageListener , który jest powiązany z sesją JMS.
8. Po zakończeniu przetwarzania wartość ServerSession jest zwracana do puli.

Serwer aplikacji zwykle dostarcza pulę ServerSessioni funkcję ServerSession .

Korzystanie z narzędzia administracyjnego JMS produktu WebSphere MQ

Za pomocą narzędzia administracyjnego można zdefiniować właściwości ośmiu typów klas produktu WebSphere MQ dla obiektu JMS i zapisać je w przestrzeni nazw JNDI. Aplikacje mogą następnie korzystać z interfejsu JNDI w celu pobrania tych administrowanych obiektów z przestrzeni nazw.

Obiekty JMS klasy WebSphere MQ , którymi można administrować za pomocą tego narzędzia to:

- MQConnectionFactory
- Fabryka MQQueueConnection
- Fabryka MQTopicConnection
- MQQUEUE
- Temat MQTopic
- MQXAConnectionFactory
- Fabryka MQXAQueueConnection
- Fabryka MQXATopicConnection

Szczegółowe informacje na temat tych obiektów zawiera sekcja “Administrowanie obiektami JMS” na stronie 965 .

Typy i wartości właściwości, które są wymagane do użycia tego narzędzia, są wymienione w sekcji Właściwości obiektów produktu IBM WebSphere MQ classes for JMS.

Narzędzie pozwala również administratorom na manipulowanie podkontekstami przestrzeni nazw katalogów w ramach interfejsu JNDI. Patrz sekcja “Manipulowanie podkontekstami za pomocą narzędzia administracyjnego WebSphere MQ JMS” na stronie 964.

Obiekty administrowane JMS można również tworzyć i konfigurować przy użyciu programu WebSphere MQ Explorer.

Wywoływanie narzędzia administracyjnego IBM WebSphere MQ classes for JMS

Narzędzie administracyjne ma interfejs wiersza komend. Tego interaktywnego można użyć lub użyć do uruchomienia procesu wsadowego.

Tryb interaktywny udostępnia wiersz komend, w którym można wprowadzić komendy administracyjne. W trybie wsadowym komenda uruchamiający narzędzie zawiera nazwę pliku zawierającego skrypt komend administracyjnych.

tryb interaktywny

Aby uruchomić narzędzie w trybie interaktywnym, wprowadź komendę:

```
JMSAdmin [-t] [-v] [-cfg config_filename]
```

gdzie:

-t
Włącza śledzenie (domyślnie śledzenie jest wyłączone)

Plik śledzenia jest generowany w systemie "%MQ_JAVA_DATA_PATH%"\errors (Windows) lub /var/mqm/trace (UNIX). Nazwa pliku śledzenia ma postać:

mqjms_*PID*.trc

gdzie *PID* jest identyfikatorem procesu maszyny JVM.

-v
Generuje szczegółowe dane wyjściowe (wartość domyślna to terse wyjście)

-cfg nazwa_pliku_konfiguracyjnego
Nazwij alternatywny plik konfiguracyjny. Jeśli ten parametr zostanie pominięty, zostanie użyty domyślny plik konfiguracyjny JMSAdmin.config. (Patrz sekcja “Konfigurowanie narzędzia administracyjnego JMS” na stronie 961).

Zostanie wyświetlona zachęta wiersza komend, która wskazuje, że narzędzie jest gotowe do akceptowania komend administracyjnych. Ten monit początkowo pojawia się jako:

```
InitCtx>
```

Wskazuje, że kontekst bieżący (czyli kontekst JNDI, do którego odwołuje się obecnie wszystkie operacje nazewnictwa i katalogowe) jest początkowym kontekstem zdefiniowanym w parametrze konfiguracyjnym PROVIDER_URL (patrz “Konfigurowanie narzędzia administracyjnego JMS” na stronie 961).

W miarę przechodzenia przestrzeni nazw katalogu, monit zmienia się, aby odzwierciedlić to, tak aby w wierszu komend zawsze wyświetlany był bieżący kontekst.

Tryb wsadowy

Aby uruchomić narzędzie w trybie wsadowym, wprowadź komendę:

```
JMSAdmin <test.scp
```

gdzie *test.scp* jest plikiem skryptowym, który zawiera komendy administracyjne (patrz “Komendy administracyjne w narzędziu administracyjnym JMS produktu WebSphere MQ” na stronie 963). Ostatnią komendą w pliku musi być komenda END .

Konfigurowanie narzędzia administracyjnego JMS

Narzędzie WebSphere MQ JMS Administration korzysta z pliku konfiguracyjnego w celu ustawienia wartości określonych właściwości. Dostarczany jest przykładowy plik, który można dostosować do używanego systemu.

Plik konfiguracyjny jest plikiem tekstowym z tekstem, który składa się z zestawu par klucz-wartość oddzielonych znakiem równości (=). Pokazano to w następującym przykładzie:

```
#Set the service provider
    INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
#Set the initial context
    PROVIDER_URL=ldap://polaris/o=ibm_us,c=us
#Set the authentication type
    SECURITY_AUTHENTICATION=none
```

(# w pierwszej kolumnie wiersza wskazuje komentarz, lub wiersz, który nie jest używany).

Przykładowy plik konfiguracyjny jest dostarczany razem z produktem WebSphere MQ. Plik nosi nazwę JMSAdmin.configi znajduje się w katalogu <MQ_JAVA_INSTALL_PATH>/bin . Zmodyfikuj ten plik, aby dostosować go do konfiguracji systemu.

Skonfiguruj narzędzie administracyjne z wartościami dla następujących właściwości:

INITIAL_CONTEXT_FACTORY
Dostawca usług, z którego korzysta narzędzie. Obsługiwane wartości dla tej właściwości są następujące:

- com.sun.jndi.ldap.LdapCtxFactory (dla LDAP)
- com.sun.jndi.fscontext.RefFSContextFactory (dla kontekstu systemu plików)

Istnieje również możliwość użycia fabryki InitialContext, która nie znajduje się na poprzedniej liście. Więcej szczegółów na ten temat zawiera sekcja “Korzystanie z niewymienionej fabryki InitialContextz narzędziem administracyjnym WebSphere MQ JMS” na stronie 962.

PROVIDER_URL
Adres URL kontekstu początkowego sesji. Katalog główny wszystkich operacji JNDI wykonywanych przez narzędzie. Obsługiwane są dwie formy tej właściwości:

- ldap://nazwa_hosta/nazwa_kontekstu
- file: [dysk:] /ścieżka

Format adresu URL LDAP może być różny w zależności od dostawcy LDAP. Więcej informacji na ten temat zawiera dokumentacja LDAP.

UWIERZYTELNIANIE SECURITY_AUTHENTICATION
Określa, czy interfejs JNDI przekazuje informacje autoryzacyjne zabezpieczeń do dostawcy usług. Ta właściwość jest używana tylko wtedy, gdy używany jest dostawca usług LDAP. Ta właściwość może przyjmować jedną z trzech wartości:

- brak (uwierzytelnianie anonimowe)
- proste (uwierzytelnianie proste)
- CRAM-MD5 (mechanizm uwierzytelnianiaCRAM-MD5)

Jeśli nie zostanie podana poprawna wartość, wartością domyślną właściwości będzie none. Więcej informacji na temat zabezpieczeń za pomocą narzędzia administracyjnego można znaleźć w sekcji “Konfigurowanie zabezpieczeń dla narzędzia administracyjnego JMS” na stronie 963 .

Te właściwości są ustawiane w pliku konfiguracyjnym. Po wywołaniu narzędzia można określić tę konfigurację, korzystając z parametru wiersza komend `-cfg` , zgodnie z opisem w sekcji “Wywoływanie narzędzia administracyjnego IBM WebSphere MQ classes for JMS” na stronie 960. Jeśli nie zostanie określona nazwa pliku konfiguracyjnego, narzędzie podejmie próbę załadowania domyślnego pliku konfiguracyjnego (`JMSAdmin.config`). Ten plik jest wyszukiwaniem najpierw w bieżącym katalogu, a następnie w katalogu `<MQ_JAVA_INSTALL_PATH>/bin` , gdzie `<MQ_JAVA_INSTALL_PATH>` jest ścieżką do klas produktu WebSphere MQ dla instalacji JMS.

Korzystanie z niewymienionej fabryki InitialContextz narzędziem administracyjnym WebSphere MQ JMS

Obsługiwane są dwie wartości fabryczne InitialContext. Innych kontekstów JNDI można użyć, ustawiając parametry w pliku konfiguracyjnym administrowania JMS.

Za pomocą narzędzia administracyjnego można łączyć się z kontekstami JNDI innymi niż wymienione w programie “Konfigurowanie narzędzia administracyjnego JMS” na stronie 961 , korzystając z trzech parametrów zdefiniowanych w pliku konfiguracyjnym JMSAdmin.

Aby użyć innej fabryki InitialContext, wykonaj następujące czynności:

1. Ustaw właściwość INITIAL_CONTEXT_FACTORY na wymaganą nazwę klasy.
2. Należy zdefiniować zachowanie fabryki InitialContextprzy użyciu właściwości USE_INITIAL_DIR_CONTEXT, NAME_PREFIX i NAME_READABILITY_MARKER.

Ustawienia dla tych właściwości są opisane w przykładowych komentarzach do pliku konfiguracyjnego.

Jeśli używana jest jedna z obsługiwanych wartości INITIAL_CONTEXT_FACTORY, nie ma potrzeby definiowania trzech wymienionych tutaj właściwości. Można jednak nadać im wartości, aby przesłonić wartości domyślne systemu. Jeśli zostanie pominięty co najmniej jeden z trzech właściwości fabryki InitialContext, narzędzie administracyjne udostępnia odpowiednie wartości domyślne w oparciu o wartości innych właściwości.

Konfigurowanie zabezpieczeń dla narzędzia administracyjnego JMS

Użyj właściwości SECURITY_AUTHENTICATION, aby określić, czy referencje zabezpieczeń są przekazywane do dostawcy usług.

Właściwość SECURITY_AUTHENTICATION jest opisana w sekcji "Konfigurowanie narzędzia administracyjnego JMS" na stronie 961. Jego działanie jest następujące:

- Jeśli ten parametr zostanie ustawiony na wartość none, interfejs JNDI nie przekazuje żadnych referencji zabezpieczeń do dostawcy usług, a *anonimowe uwierzytelnianie* jest wykonywane.
- Jeśli parametr zostanie ustawiony na wartość `simple` lub `CRAM-MD5`, referencje zabezpieczeń są przekazywane za pośrednictwem interfejsu JNDI do bazowego dostawcy usług. Te referencje zabezpieczeń mają postać nazwy wyróżniającej użytkownika (nazwa wyróżniająca użytkownika) i hasła.

Jeśli referencje zabezpieczeń są wymagane, użytkownik zostanie poproszony o ich zainicjowanie. Należy tego uniknąć, ustawiając właściwości PROVIDER_USERDN i PROVIDER_PASSWORD w pliku konfiguracyjnym JMSAdmin.

Uwaga: Jeśli te właściwości nie zostaną użyte, wprowadzony tekst, *wraz z hasłem*, będzie wyświetlany na ekranie. Może to mieć wpływ na bezpieczeństwo.

Narzędzie nie samo uwierzytelnia się; zadanie jest delegowane do serwera LDAP. Administrator serwera LDAP musi skonfigurować i obsługiwać uprawnienia dostępu do różnych części katalogu. Więcej informacji na ten temat zawiera dokumentacja LDAP. Jeśli uwierzytelnianie nie powiedzie się, narzędzie wyświetli odpowiedni komunikat o błędzie i kończy działanie.

Więcej szczegółowych informacji na temat zabezpieczeń i interfejsu JNDI znajduje się w dokumentacji na stronie Sun's Java (https://java.sun.com).

Komendy administracyjne w narzędziu administracyjnym JMS produktu WebSphere MQ

Narzędzie administracyjne akceptuje komendy składające się z komendy administracyjnej i jej odpowiednich parametrów.

Po wyświetleniu zachęty wiersza komend narzędzie jest gotowe do akceptowania komend. Komendy administracyjne mają zwykle następującą postać:

```
verb [param]*
```

gdzie **verb** jest jednym z komend administracyjnych wymienionych w sekcji Tabela 133 na stronie 963. Wszystkie poprawne polecenia zawierają jeden czasownik, który pojawia się na początku polecenia w jego standardowym lub krótkim formacie.

Parametry czasownika mogą być uzależnione od czasownika. Na przykład komenda END nie może przyjmować żadnych parametrów, ale komenda DEFINE może przyjmować dowolną liczbę parametrów. Szczegółowe informacje na temat czasowników, które zawierają co najmniej jeden parametr, są omówione w tematach pokrewnych.

Tabela 133. Czasowniki administracyjne

Czasownik	**Postać krótka**	**Opis**
Zmień	ALT	Zmień co najmniej jedną z właściwości administrowanego obiektu
Definiowanie	DEF	Tworzenie i zapisywanie administrowanego obiektu lub tworzenie podkontekstu
WYŚWIETL	DIS	Wyświetlanie właściwości jednego lub większej liczby składowanych obiektów administrowanych lub zawartości bieżącego kontekstu
USUŃ	Del	Usuń jeden lub więcej administrowanych obiektów z przestrzeni nazw lub usuń pusty podkontekst

Tabela 133. Czasowniki administracyjne (kontynuacja)		
Czasownik	**Postać krótka**	**Opis**
CHANGE	chg	Zmień bieżący kontekst, umożliwiając użytkownikowi przeglądanie przestrzeni nazw katalogu w dowolnym miejscu poniżej kontekstu początkowego (w oczekiwaniu na dopuszczenie zabezpieczeń).
COPY	CP	Utwórz kopię przechowywanego obiektu administrowanego, przechowując go pod alternatywną nazwą
PRZENIEŚ	MV	Zmień nazwę, pod którą administrowany obiekt jest składowany
KONIEC		Zamknij narzędzie administracyjne

W nazwach komend nie jest rozróżniana wielkość liter.

Zazwyczaj w celu zakończenia komend należy nacisnąć klawisz powrotu karetki. Można jednak przesłonić tę wartość, wpisując znak plus (+) bezpośrednio przed powrotem karetki. Pozwala to na wprowadzanie komend wielowierszowych, jak pokazano w poniższym przykładzie:

```
DEFINE Q(BookingsInputQueue) +
       QMGR(QM.POLARIS.TEST) +
       QUEUE(BOOKINGS.INPUT.QUEUE) +
       PORT(1415) +
       CCSID(437)
```

Wiersze rozpoczynające się od dowolnego z następujących znaków są traktowane jako komentarze i są ignorowane: * #/.

Manipulowanie podkontekstami za pomocą narzędzia administracyjnego WebSphere MQ JMS

Za pomocą komend **CHANGE**, **DEFINE**, **DISPLAY** i **DELETE** można manipulować podkontekstami przestrzeni nazw katalogów.

Korzystanie z tych komend jest opisane w sekcji Tabela 134 na stronie 964.

Tabela 134. Składnia i opis komend używanych do manipulowania podkontekstami	
Składnia komend	**Opis**
DEFINE CTX (ctxName)	Podejmuje próbę utworzenia podrzędnego kontekstu bieżącego kontekstu o nazwie ctxName. Nie powiedzie się, jeśli istnieje naruszenie zabezpieczeń, jeśli podkontekst już istnieje lub jeśli podana nazwa jest niepoprawna.
WYŚWIETL CTX	Wyświetla zawartość bieżącego kontekstu. Administrowane obiekty są opatrzone adnotacjami a, podkontekstami z `[D]`. Wyświetlany jest również typ Java każdego obiektu.
DELETE CTX (ctxName)	Podejmuje próbę usunięcia kontekstu potomnego bieżącego kontekstu o nazwie ctxName. Niepowodzenie, jeśli kontekst nie zostanie znaleziony, nie jest pusty lub jeśli istnieje naruszenie zabezpieczeń.

Tabela 134. Składnia i opis komend używanych do manipulowania podkontekstami (kontynuacja)	
Składnia komend	**Opis**
CHANGE CTX (ctxName)	Zmienia bieżący kontekst w taki sposób, że odwołuje się on teraz do kontekstu potomnego o nazwie ctxName. Można podać jedną z dwóch wartości specjalnych ctxName : **= WŁĄCZONY** przenosi do elementu nadrzędnego bieżącego kontekstu **= INIT** przesuwa się bezpośrednio do kontekstu początkowego Nie powiedzie się, jeśli podany kontekst nie istnieje, lub jeśli istnieje naruszenie zabezpieczeń.

Administrowanie obiektami JMS

W tej sekcji opisano osiem typów obiektów, które można obsłużyć za pomocą narzędzia administracyjnego. Zawiera ona szczegółowe informacje o każdej konfigurowalnej właściwości oraz czasowników, które mogą nimi manipulować.

Obiekty administrowane JMS można również tworzyć i konfigurować przy użyciu programu WebSphere MQ Explorer.

Typy obiektów JMS

W tabeli przedstawiono osiem typów obiektów administrowanych.

W kolumnie Słowa kluczowe są wyświetlane łańcuchy, które można zastąpić *TYPE* w komendach przedstawionych w sekcji Tabela 136 na stronie 966.

Tabela 135. Typy obiektów JMS obsługiwane przez narzędzie administracyjne		
Typ obiektu	**Słowo kluczowe**	**Opis**
MQConnectionFactory	CF	Implementacja interfejsu JMS ConnectionFactory w produkcie WebSphere MQ . Jest to obiekt fabryczny do tworzenia połączeń zarówno w domenach typu punkt z punktem, jak i publikowania/subskrypcji.
Fabryka MQQueueConnection	QCF	Implementacja interfejsu JMS QueueConnectionproduktu WebSphere MQ . Reprezentuje on obiekt fabryczny do tworzenia połączeń w domenie typu punkt z punktem.
Fabryka MQTopicConnection	TCF	Implementacja interfejsu JMS TopicConnectionproduktu WebSphere MQ . Reprezentuje on obiekt fabryczny do tworzenia połączeń w domenie publikowania/subskrypcji.
MQQUEUE	Q	Implementacja interfejsu kolejki JMS produktu WebSphere MQ . Jest to miejsce docelowe dla komunikatów w domenie typu punkt z punktem.

Tabela 135. Typy obiektów JMS obsługiwane przez narzędzie administracyjne (kontynuacja)		
Typ obiektu	**Słowo kluczowe**	**Opis**
Temat MQTopic	T	Implementacja interfejsu tematu JMS produktu WebSphere MQ . Reprezentuje miejsce docelowe dla komunikatów w domenie publikowania/subskrybowania.
MQXAConnectionFactory"1" na stronie 966	XACF	Implementacja interfejsu JMS XAConnectionFactory w produkcie WebSphere MQ . Jest to obiekt fabryczny do tworzenia połączeń zarówno w domenach typu punkt z punktem, jak i publikowania/ subskrybowania, a także w przypadku, gdy połączenia korzystają z wersji XA klas interfejsu JMS.
MQXAQueueConnectionFactory"1" na stronie 966	XAQCF	Implementacja interfejsu JMS XAQueueConnectionproduktu WebSphere MQ . Reprezentuje on obiekt fabryczny do tworzenia połączeń w domenie typu punkt z punktem, w których używane są klasy XA klasy JMS.
MQXATopicConnectionFactory"1" na stronie 966	XATCF	Implementacja interfejsu JMS XATopicConnectionproduktu WebSphere MQ . Reprezentuje on obiekt fabryczny do tworzenia połączeń w domenie publikowania/subskrypcji, w których używane są klasy interfejsu XA interfejsu XA.

Uwaga:

1. Klasy te są udostępniane do użytku przez dostawców serwerów aplikacji. Jest mało prawdopodobne, aby były one bezpośrednio przydatne dla programistów aplikacji.

Czasowniki używane z obiektami JMS

W celu manipulowania administrowanymi obiektami w przestrzeni nazw katalogu można użyć komend `ALTER`, `DEFINE`, `DISPLAY`, `DELETE`, `COPY`i `MOVE` .

Tabela 136 na stronie 966 podsumowuje użycie tych komend. Zastąp *TYPE* słowem kluczowym, które reprezentuje wymagany obiekt administrowany, zgodnie z wykazanymi w sekcji Tabela 135 na stronie 965.

Tabela 136. Składnia i opis komend używanych do manipulowania administrowanymi obiektami	
Składnia komend	**Opis**
ALTER *TYP*(nazwa) [właściwość] *	Podejmuje próbę zaktualizowania właściwości administrowanego obiektu z dostarczonym. Nie powiedzie się, jeśli wystąpi naruszenie zabezpieczeń, jeśli określony obiekt nie może zostać znaleziony lub jeśli podane nowe właściwości są niepoprawne.

Tabela 136. Składnia i opis komend używanych do manipulowania administrowanymi obiektami (kontynuacja)

Składnia komend	**Opis**
DEFINE *TYP*(nazwa) [właściwość] *	Próbuje utworzyć obiekt administrowany typu *TYPE* z dostarczonym właściwościami, a następnie zapisać go pod nazwą name w bieżącym kontekście. Nie powiedzie się, jeśli istnieje naruszenie zabezpieczeń, jeśli podana nazwa nie jest poprawna lub istnieje obiekt o tej nazwie, lub jeśli podane właściwości są niepoprawne.
DISPLAY *TYP*(nazwa)	Wyświetla właściwości administrowanego obiektu typu *TYPE*, związanego pod nazwą name w bieżącym kontekście. Nie powiedzie się, jeśli obiekt nie istnieje, lub jeśli istnieje naruszenie zabezpieczeń.
DELETE *TYP*(nazwa)	Próbuje usunąć administrowany obiekt typu *TYPE*, o nazwie name, z bieżącego kontekstu. Nie powiedzie się, jeśli obiekt nie istnieje, lub jeśli istnieje naruszenie zabezpieczeń.
COPY *TYP*(nameA) *TYP*(nameB)	Tworzy kopię administrowanego obiektu typu *TYPE*o nazwie nameA, nazywając kopię nameB. To wszystko występuje w zasięgu bieżącego kontekstu. Nie powiedzie się, jeśli obiekt, który ma zostać skopiowany, nie istnieje, jeśli istnieje obiekt o nazwie nameB , lub jeśli istnieje naruszenie zabezpieczeń.
MOVE *TYP*(nameA) *TYP*(nameB)	Służy do przenoszenia (zmiany nazwy) obiektu administrowanego typu *TYPE*o nazwie nameAdo nameB. To wszystko występuje w zasięgu bieżącego kontekstu. Nie powiedzie się, jeśli obiekt, który ma zostać przeniesiony, nie istnieje, jeśli istnieje obiekt o nazwie nameB , lub jeśli istnieje naruszenie zabezpieczeń.

Tworzenie obiektów za pomocą narzędzia administracyjnego WebSphere MQ JMS

Za pomocą komendy DEFINE utwórz obiekty i zapisz je w przestrzeni nazw JNDI,

Użyj następującej składni komendy:

```
DEFINE TYPE(name) [property]*
```

Oznacza to, że komenda DEFINE , po której następuje odwołanie do obiektu administrowanego *TYPE* (name) , po której następuje zero lub więcej *właściwości* (patrz Właściwości obiektów IBM WebSphere MQ classes for JMS).

Uwagi dotyczące nazewnictwa LDAP dla obiektów JMS

Aby przechowywać obiekty w środowisku LDAP, należy nadać im nazwy zgodne z określonymi konwencjami. Narzędzie administracyjne może pomóc w przestrzeganiem konwencji nazewnictwa przez dodanie domyślnego przedrostka.

Jedna konwencja nazewnictwa jest taka, że nazwy obiektów i podkontekstu muszą zawierać przedrostek, taki jak cn= (nazwa zwykła), lub ou= (jednostka organizacyjna).

Narzędzie administracyjne upraszcza korzystanie z dostawców usług LDAP, umożliwiając odwoływaniu się do nazw obiektów i kontekstów bez przedrostka. Jeśli przedrostek nie zostanie podany, narzędzie automatycznie doda domyślny przedrostek do nazwy dostarczanej przez użytkownika. W przypadku katalogu LDAP jest to cn=.

Domyślny przedrostek można zmienić, ustawiając właściwość NAME_PREFIX w pliku konfiguracyjnym JMSAdmin, zgodnie z opisem w sekcji “Korzystanie z niewymienionej fabryki InitialContextz narzędziem administracyjnym WebSphere MQ JMS” na stronie 962.

Pokazano to w poniższym przykładzie.

```
InitCtx> DEFINE Q(testQueue)

InitCtx> DISPLAY CTX

    Contents of InitCtx

      a  cn=testQueue              com.ibm.mq.jms.MQQueue

     1 Object(s)
       0 Context(s)
       1 Binding(s), 1 Administered
```

Mimo że podana nazwa obiektu (`testQueue`) nie ma przedrostka, narzędzie automatycznie dodaje jeden, aby upewnić się, że jest zgodna z konwencją nazewnictwa LDAP. Podobnie, wprowadzenie komendy `DISPLAY Q(testQueue)` powoduje dodanie tego przedrostka.

Może być konieczne skonfigurowanie serwera LDAP do przechowywania obiektów Java. Informacje pomocne w tej konfiguracji można znaleźć w dokumentacji serwera LDAP.

Przykładowe warunki błędu podczas tworzenia obiektu JMS

Podczas tworzenia obiektu może wystąpić pewna liczba typowych warunków błędu.

Poniżej przedstawiono przykłady tych warunków błędu:

Właściwość CipherSpec odwzorowana na wartość CipherSuite

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) SSLCIPHERSUITE(RC4_MD5_US)
  WARNING: Converting CipherSpec RC4_MD5_US to
  CipherSuite SSL_RSA_WITH_RC4_128_MD5
```

Niepoprawna właściwość dla obiektu

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) PRIORITY(4)
  Unable to create a valid object, please check the parameters supplied
  Invalid property for a QCF: PRI
```

Niepoprawny typ wartości właściwości

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) CCSID(english)
  Unable to create a valid object, please check the parameters supplied
  Invalid value for CCS property: English
```

Właściwość clash-client/bindings

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) HOSTNAME(polaris.hursley.ibm.com)
  Unable to create a valid object, please check the parameters supplied
  Invalid property in this context: Client-bindings attribute clash
```

Clash właściwości-inicjowanie wyjścia

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) SECEXITINIT(initStr)
  Unable to create a valid object, please check the parameters supplied
  Invalid property in this context: ExitInit string supplied
  without Exit string
```

Wartość właściwości poza poprawnym zakresem

```
InitCtx/cn=Trash> DEFINE Q(testQ) PRIORITY(12)
  Unable to create a valid object, please check the parameters supplied
  Invalid value for PRI property: 12
```

nieznana właściwość

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) PIZZA(ham and mushroom)
  Unable to create a valid object, please check the parameters supplied
  Unknown property: PIZZA
```

Poniżej znajdują się przykłady warunków błędów, które mogą wystąpić w systemie Windows podczas wyszukiwania obiektów administrowanych JNDI z aplikacji JMS.

1. Jeśli używany jest dostawca JNDI WebSphere (com.ibm.websphere.naming.WsnInitialContextFactory), należy użyć ukośnika (/) w celu uzyskania dostępu do administrowanych obiektów zdefiniowanych w podkontekstach, na przykład jms/MyQueueNazwa. Jeśli używany jest ukośnik odwrotny (\), zgłaszany jest wyjątek InvalidName.
2. Jeśli używany jest dostawca JNDI Sun, com.sun.jndi.fscontext.RefFSContextFactory, należy użyć ukośnika odwrotnego (\), aby uzyskać dostęp do administrowanych obiektów zdefiniowanych w kontekstach podrzędnych. Na przykład ctx1\\fred. Jeśli używany jest ukośnik (/), zostanie zgłoszony wyjątek NameNotFoundException .

Korzystanie z programu WebSphere MQ Explorer dla konfiguracji JMS

Graficzny interfejs użytkownika programu IBM WebSphere MQ Explorer umożliwia tworzenie obiektów JMS z obiektów WebSphere MQ oraz obiektów WebSphere MQ z obiektów JMS, a także administrowanie i monitorowanie innych obiektów produktu WebSphere MQ .

Zanim rozpoczniesz

Przed utworzeniem i skonfigurowaniem obiektów administrowanych JMS za pomocą programu WebSphere MQ Explorer należy dodać kontekst początkowy w celu zdefiniowania elementu głównego przestrzeni nazw JNDI, w której obiekty JMS są przechowywane w usłudze katalogowej i nazewnictwa. Więcej informacji na ten temat można znaleźć w pomocy dla użytkownika programu IBM WebSphere MQ Explorer dla obiektów administrowanych JMS.

O tym zadaniu

Za pomocą programu IBM WebSphere MQ Explorer można wykonywać następujące czynności: kontekstowo z istniejącego obiektu w Eksploratorze IBM WebSphere MQ lub z poziomu kreatora tworzenia nowego obiektu. Przykłady typowych zadań dla programu WebSphere MQ Explorer można znaleźć w pomocy do programu WebSphere MQ Explorer.

Procedura

- Utwórz fabrykę połączeń JMS z dowolnego z następujących obiektów produktu WebSphere MQ :
  a) Menedżer kolejek produktu WebSphere MQ , niezależnie od tego, czy na komputerze lokalnym, czy w systemie zdalnym.
  b) Kanał produktu WebSphere MQ
  c) Program nasłuchujący produktu WebSphere MQ
- Dodawanie menedżera kolejek produktu WebSphere MQ do programu WebSphere MQ Explorer przy użyciu fabryki połączeń JMS
- Tworzenie kolejki JMS z kolejki produktu WebSphere MQ
- Tworzenie kolejki produktu WebSphere MQ z kolejki JMS
- Tworzenie tematu JMS z poziomu tematu produktu WebSphere MQ , który może być obiektem produktu WebSphere MQ lub tematem dynamicznym.
- Tworzenie tematu produktu WebSphere MQ z tematu JMS

Korzystanie z pakietu WebSphere MQ Headers

Pakiet WebSphere MQ Headers udostępnia zestaw pomocniczych interfejsów i klas, których można użyć do manipulowania nagłówkami WebSphere MQ komunikatu. Zwykle jest używany pakiet WebSphere MQ Headers, ponieważ usługi administracyjne mają być wykonywane przy użyciu serwera komend (przy użyciu komunikatów PCF).

O tym zadaniu

Pakiet WebSphere MQ Headers znajduje się w pakietach `com.ibm.mq.headers` i `com.ibm.mq.pcf` . Tego obiektu można użyć dla obu alternatywnych interfejsów API, które produkt WebSphere MQ udostępnia do użycia w aplikacjach Java:

- Klasy produktu WebSphere MQ dla języka Java (nazywane również WebSphere MQ Headers Base Java).
- Klasy produktu WebSphere MQ dla usługi Java Message Service (klasy produktu WebSphere MQ dla usługi JMS, zwane również WebSphere MQ JMS).

WebSphere MQ Podstawowe aplikacje Java zwykle manipulują obiektami MQMessage, a klasy obsługi nagłówków mogą bezpośrednio wchodzić w interakcje z tymi obiektami, ponieważ rozpoznają one natywnie interfejsy Java produktu WebSphere MQ Base.

W produkcie WebSphere MQ JMS ładunkiem dla komunikatu jest zwykle obiekt typu String lub bajtowy, który może być manipulowany strumieniami DataInput i DataOutput . Pakiet WebSphere MQ Headers może być używany do interakcji z tymi strumieniami danych i może być używany do manipulowania komunikatami produktu MQ , które są wysyłane i odbierane przez aplikacje JMS produktu WebSphere MQ .

Oznacza to, że chociaż pakiet WebSphere MQ Headers zawiera odwołania do pakietu WebSphere MQ Base Java, jest on również przeznaczony do użycia w aplikacjach WebSphere MQ JMS i jest odpowiedni do użytku w środowiskach Java Platform, Enterprise Edition (Java EE).

Typowym sposobem, w jaki można użyć pakietu WebSphere MQ Headers, jest manipulowanie komunikatami administracyjnymi w formacie PCF (Programmable Command Format), na przykład z następujących powodów:

- Aby uzyskać dostęp do szczegółowych informacji na temat zasobu WebSphere MQ .
- Służy do monitorowania głębokości kolejki.
- Zablokowanie dostępu do kolejki.

Korzystając z komunikatów PCF z interfejsem API JMS produktu WebSphere MQ , tego rodzaju administrowanie zasobami centrycznymi aplikacji można wykonywać w aplikacjach Java EE bez konieczności używania interfejsu API języka Java produktu WebSphere MQ .

Procedura

- Aby użyć pakietu WebSphere MQ Headers do manipulowania nagłówkami komunikatów dla klas produktu WebSphere MQ dla języka Java, należy zapoznać się z treścią “Używanie z klasami produktu WebSphere MQ dla języka Java” na stronie 970.
- Informacje na temat używania pakietu WebSphere MQ Headers do manipulowania nagłówkami komunikatów dla usługi JMS zawiera sekcja “Używanie z klasami produktu WebSphere MQ dla usługi JMS” na stronie 971.

Używanie z klasami produktu WebSphere MQ dla języka Java

Klasy produktu WebSphere MQ dla aplikacji Java zwykle manipulują obiektami MQMessage, a klasy obsługi nagłówków mogą bezpośrednio wchodzić w interakcje z tymi obiektami, ponieważ rozpoznają one natywnie klasy WebSphere MQ dla interfejsów Java.

O tym zadaniu

Produkt WebSphere MQ udostępnia przykładowe aplikacje, które demonstrują sposób korzystania z pakietu WebSphere MQ Headers za pomocą interfejsu API języka Java produktu WebSphere MQ (klasy WebSphere MQ dla języka Java).

Przykłady pokazują dwie rzeczy:

- Jak utworzyć komunikat PCF w celu wykonania czynności administracyjnej i przeanalizowania komunikatu odpowiedzi.

- Sposób wysyłania tego komunikatu PCF przy użyciu klas produktu WebSphere MQ dla języka Java.

W zależności od używanej platformy, te przykłady są instalowane w katalogu `pcf` w katalogu `samples` lub `tools` instalacji produktu WebSphere MQ (patrz "Katalogi instalacyjne dla klas produktu WebSphere MQ dla języka Java" na stronie 673).

Procedura

1. Utwórz komunikat PCF, aby wykonać działanie administracyjne i przeanalizuj komunikat odpowiedzi.
2. Wyślij ten komunikat PCF za pomocą klas WebSphere MQ dla języka Java.

Pojęcia pokrewne

"Obsługa nagłówków komunikatów produktu WebSphere MQ z klasami produktu WebSphere MQ dla języka Java" na stronie 695
Dostępne są klasy Java reprezentujące różne typy nagłówków komunikatów. Dostępne są również dwie klasy pomocnicze.

"Obsługa komunikatów PCF z klasami produktu WebSphere MQ dla języka Java" na stronie 700
Klasy Java są udostępniane do tworzenia i analizowania komunikatów ustrukturyzowanych PCF, a także do ułatwiania wysyłania żądań PCF i zbierania odpowiedzi PCF.

Używanie z klasami produktu WebSphere MQ dla usługi JMS

Aby używać nagłówków WebSphere MQ z klasami WebSphere MQ dla usługi JMS, należy wykonać te same podstawowe kroki, co w przypadku klas produktu WebSphere MQ dla języka Java. Komunikat PCF może zostać utworzony, a odpowiedź przeanalizowana dokładnie w ten sam sposób, korzystając z pakietu WebSphere MQ Headers i tego samego przykładowego kodu, co dla klas WebSphere MQ dla języka Java.

O tym zadaniu

Aby wysłać komunikat PCF za pomocą interfejsu API WebSphere MQ , ładunek komunikatu musi zostać zapisany w bajcie JMS (JMS Bytes) i wysyłany przy użyciu standardowych interfejsów API JMS. Jedyną kwestią jest to, że komunikat nie może zawierać zapytania JMS RFH2 ani żadnych innych nagłówków o konkretnych wartościach w strukturze MQMD.

Aby wysłać komunikat PCF, wykonaj następujące kroki. Sposób tworzenia komunikatu PCF i informacje są wyodrębniane z komunikatu odpowiedzi w taki sam sposób, jak w przypadku klas WebSphere MQ dla języka Java (patrz "Używanie z klasami produktu WebSphere MQ dla języka Java" na stronie 970).

Procedura

1. Utwórz miejsce docelowe kolejki JMS, które reprezentuje SYSTEM.ADMIN.COMMAND.QUEUE.

   Aplikacje JMS produktu WebSphere MQ wysyłają komunikaty PCF do systemu SYSTEM.ADMIN.COMMAND.QUEUEi musi mieć dostęp do obiektu docelowego JMS, który reprezentuje tę kolejkę. Miejsce docelowe musi mieć ustawione następujące właściwości:

   ```
   WMQ_MQMD_WRITE_ENABLED = YES
   WMQ_MESSAGE_BODY = MQ
   ```

   Jeśli używany jest serwer aplikacji WebSphere Application Server, należy zdefiniować te właściwości jako właściwości niestandardowe w miejscu docelowym.

   Aby utworzyć miejsce docelowe w sposób programowy z poziomu aplikacji, należy użyć następującego kodu:

   ```
   Queue q1 = session.createQueue("SYSTEM.ADMIN.COMMAND.QUEUE");
   ((MQQueue) q1).setIntProperty(WMQConstants.WMQ_MESSAGE_BODY,
       WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ((MQQueue) q1).setMQMDWriteEnabled(true);
   ```

2. Przekształć komunikat PCF w komunikat z bajtłem JMS zawierający poprawne wartości MQMD.

Konieczne jest utworzenie komunikatu JMS (w bajtach) i napisanego do niego komunikatu PCF. Należy utworzyć kolejkę odpowiedzi, ale nie musi ona mieć żadnych konkretnych ustawień.

W poniższym przykładowym fragmencie kodu przedstawiono, w jaki sposób można utworzyć komunikat JMS B i zapisać w nim obiekt com.ibm.mq.headers,pcf.PCFMessage . Obiekt PCFMessage (pcfCmd) został wcześniej zbudowany przy użyciu pakietu WebSphere MQ Headers. (Należy zwrócić uwagę na pakiet, który ma załadować komunikat PCFMessage: com.ibm.mq.headers.pcf.PCFMessage).

```
 // create the JMS Bytes Message
final BytesMessage msg = session.createBytesMessage();

// Create the wrapping streams to put the bytes into the message payload
ByteArrayOutputStream baos = new ByteArrayOutputStream();
DataOutput dataOutput = new DataOutputStream(baos);

// Set the JMSReplyTo so the answer comes back
msg.setJMSReplyTo(new MQQueue("adminResp"));

// write the pcf into the stream
pcfCmd.write(dataOutput);
baos.flush();
msg.writeBytes(baos.toByteArray());

// we have taken control of the MD, so need to set all
// flags in the MD that we require - main one is the format
msg.setJMSPriority(4);
msg.setIntProperty(WMQConstants.JMS_IBM_MQMD_PERSISTENCE,
  CMQC.MQPER_NOT_PERSISTENT);
msg.setIntProperty(WMQConstants.JMS_IBM_MQMD_EXPIRY, 300);
msg.setIntProperty(WMQConstants.JMS_IBM_MQMD_REPORT,
    CMQC.MQRO_PASS_CORREL_ID);
msg.setStringProperty(WMQConstants.JMS_IBM_MQMD_FORMAT, "MQADMIN");

// and send the message
sender.send(msg);
```

3. Wyślij komunikat i odeślij odpowiedź przy użyciu standardowych interfejsów API JMS.
4. Przekształć komunikat odpowiedzi w komunikat PCF w celu przetworzenia.

   Aby pobrać komunikat odpowiedzi i przetworzyć go jako komunikat PCF, należy użyć następującego kodu:

```
 // Get the message back
BytesMessage msg = (BytesMessage) consumer.receive();

// get the size of the bytes message & read into an array
int bodySize = (int) msg.getBodyLength();
byte[] data = new byte[bodySize];
msg.readBytes(data);

// Read into Stream and DataInput Stream
ByteArrayInputStream bais = new ByteArrayInputStream(data);
DataInput dataInput = new DataInputStream(bais);

// Pass to PCF Message to process
PCFMessage response = new PCFMessage(dataInput);
```

Pojęcia pokrewne

“Komunikaty JMS” na stronie 829
Komunikaty JMS składają się z nagłówka, właściwości i treści. Usługa JMS definiuje pięć typów treści komunikatu.

Korzystanie z usług Web Service w produkcie WebSphere MQ

Aplikacje produktu IBM WebSphere MQ dla usług Web Services można tworzyć przy użyciu transportu IBM WebSphere MQ dla protokołu SOAP lub mostu IBM WebSphere MQ dla protokołu HTTP.

Transport produktu IBM WebSphere MQ dla protokołu SOAP udostępnia transport JMS dla protokołu SOAP. Transport produktu IBM WebSphere MQ dla protokołu SOAP jest również zintegrowany z innymi środowiskami, takimi jak: Microsoft Windows Communication Foundation, WebSphere Application Server i CICS Transaction Server.

Więcej informacji na temat transportu IBM WebSphere MQ dla protokołu SOAP zawiera sekcja “Transport produktu WebSphere MQ dla protokołu SOAP” na stronie 973.

Za pomocą mostu IBM WebSphere MQ dla protokołu HTTP aplikacje klienckie mogą wymieniać komunikaty z produktem IBM WebSphere MQ bez konieczności instalowania klienta MQI produktu WebSphere MQ . Produkt WebSphere MQ można wywołać z dowolnej platformy lub języka z możliwościami HTTP.

Więcej informacji na temat mostu IBM WebSphere MQ dla protokołu HTTP zawiera sekcja “Most produktu WebSphere MQ dla protokołu HTTP” na stronie 1052.

Pojęcia pokrewne

“Pojęcia związane z projektowaniem aplikacji” na stronie 8
Do pisania aplikacji IBM WebSphere MQ można użyć wyboru języków proceduralnych lub obiektowych. Odsyłacze w tym temacie można znaleźć w informacjach dotyczących pojęć związanych z produktem IBM WebSphere MQ , które są przydatne dla programistów aplikacji.

“Wybór języka programowania do użycia” na stronie 80
W tej sekcji znajdują się informacje na temat języków programowania i środowisk obsługiwanych przez produkt IBM WebSphere MQ, a także niektóre zagadnienia dotyczące ich używania.

“Projektowanie aplikacji produktu IBM WebSphere MQ” na stronie 91
Po zdecydowaniu się, w jaki sposób aplikacje mogą korzystać z platform i środowisk, które są dostępne dla użytkownika, należy zdecydować, w jaki sposób korzystać z funkcji oferowanych przez produkt WebSphere MQ.

“Przykładowe programy produktu WebSphere MQ” na stronie 98
Ta kolekcja tematów zawiera informacje na temat przykładowych programów WebSphere MQ na różnych platformach.

“Pisanie aplikacji kolejkowania” na stronie 199
Ta sekcja zawiera informacje na temat pisania aplikacji kolejkowania, łączenia się i rozłączania z menedżerem kolejek, publikowania/subskrypcji oraz obiektów otwierających i zamykających.

“Pisanie aplikacji klienckich” na stronie 360
Co należy wiedzieć, aby pisać aplikacje klienckie w produkcie WebSphere MQ.

“Zapisywanie aplikacji publikowania/subskrypcji” na stronie 285
Rozpocznij pisanie aplikacji WebSphere MQ publikowania/subskrypcji.

“Budowanie aplikacji IBM WebSphere MQ” na stronie 439
Ta sekcja zawiera informacje na temat budowania aplikacji produktu IBM WebSphere MQ na różnych platformach.

“Obsługa błędów programu” na stronie 561
Te informacje wyjaśniają błędy związane z wywołaniami MQI aplikacji, gdy wywołuje połączenie, lub gdy jego komunikat jest dostarczany do miejsca docelowego.

Transport produktu WebSphere MQ dla protokołu SOAP

Transport produktu WebSphere MQ dla protokołu SOAP udostępnia transport JMS dla protokołu SOAP. Transport produktu WebSphere MQ dla protokołu SOAP jest również zintegrowany z innymi środowiskami, takimi jak Microsoft Windows Communication Foundation, WebSphere Application Server i CICS Transaction Server.

Wprowadzenie do transportu IBM WebSphere MQ dla protokołu SOAP

Transport produktu IBM WebSphere MQ dla protokołu SOAP udostępnia transport JMS dla protokołu SOAP. Nadawca i program nasłuchujący SOAP produktu WebSphere MQ umożliwiają wywoływanie usług Web Service.

Program nasłuchujący SOAP produktu WebSphere MQ obsługuje usługi udostępniane przez środowisko .NET Framework 1, .NET Framework 2 i Axis 1.4. Nadawca SOAP produktu WebSphere MQ obsługuje klienty usług Web Service działające w środowisku .NET Framework 1, .NET Framework

2, Axis 1.4 i Axis2. Klientami mogą być zarówno serwer WebSphere MQ , jak i aplikacja kliencka. Transport produktu IBM WebSphere MQ dla protokołu SOAP jest również zintegrowany z innymi środowiskami, takimi jak Microsoft Windows Communication Foundation, WebSphere Application Server i CICS Transaction Server.

Integracja z produktem Microsoft Windows Communication Foundation jest częścią obsługi produktu IBM WebSphere MQ dla środowiska .NET Framework 3.

Transport produktu IBM WebSphere MQ dla protokołu SOAP to zestaw protokołów i narzędzi do transportu komunikatów SOAP przy użyciu usługi JMS w produkcie IBM WebSphere MQ. Jest on spakowany na różne sposoby w różnych środowiskach aplikacji, jak pokazano na Tabela 137 na stronie 974.

Tabela 137. Transport produktu IBM WebSphere MQ dla środowisk aplikacji SOAP

	Zintegrowane z dodatkowymi komponentami produktu WebSphere MQ	**Zintegrowane z ramami**
Udostępniane jako część instalacji produktu WebSphere MQ	Środowisko .NET Framework 1 .Środowisko .NET Framework 2 Axis 1.4	Windows Communication Foundation (.NET Framework 3) Axis2 (tylko klient)
Udostępnione w innym pakiecie oprogramowania		WebSphere Application Server CICS Transaction Server 4.1 WebSphere ESB WebSphere Process Server for Multiplatforms

Integracja transportu produktu IBM WebSphere MQ na potrzeby protokołu SOAP w środowisku aplikacji upraszcza projektowanie i wdrażanie usług Web Service w produkcie IBM WebSphere MQ.

Za pomocą dodatkowych komponentów SOAP produktu IBM WebSphere MQ można bezpośrednio współdziałać z komponentami SOAP produktu WebSphere MQ w celu tworzenia i wdrażania usług. Za pomocą narzędzi SOAP produktu IBM WebSphere MQ można konfigurować i wdrażać usługi Web Service oraz klienty usługi Web Service w produkcie IBM WebSphere MQ.

W zintegrowanych środowiskach programowanie i wdrażanie jest prostsze. Te same narzędzia do tworzenia i wdrażania są używane w taki sam sposób, jak w przypadku tworzenia i wdrażania usługi Web Service protokołu SOAP HTTP. Nadal należy skonfigurować kolejki, kanały i menedżery kolejek produktu IBM WebSphere MQ , które są wymagane przy użyciu narzędzi produktu WebSphere MQ .

Z dowolnego z tych środowisk można mieszać i dopasowować klienty SOAP i serwery produktu IBM WebSphere MQ .

Korzyści

Transport produktu WebSphere MQ dla protokołu SOAP korzysta z istniejących użytkowników produktu IBM WebSphere MQ , które są następujące:

Korzystanie z sieci produktu IBM WebSphere MQ w celu połączenia istniejących usług Web Service.

Usługi te mogą być świadczone przez użytkownika lub usługi udostępniane jako interfejsy do innych spakowanych aplikacji oprogramowania, które zostały wdrożone.

Korzyści pochodzą z korzystania z istniejącej sieci WebSphere MQ w celu połączenia usług Web Service. Transport produktu IBM WebSphere MQ ma tę zaletę, że jest to zarządzana i niezawodna usługa przesyłania komunikatów w kolejce.

Pisanie nowych aplikacji lub przekształcanie istniejących aplikacji w celu użycia interfejsów SOAP, a nie IBM WebSphere MQ .

Zwykle aplikacje wymagają, aby konkretny adapter WebSphere MQ był rozwijany w celu integracji z inną aplikacją. Adaptery mają dwie części: element konektora, który umieszcza i pobiera komunikaty

do i z transportu, a także element adaptera, który przekształca dane w formaty specyficzne dla aplikacji i z formatów specyficznych dla aplikacji. Integracja każdej pary aplikacji jest nowym wyzwaniem.

Korzyści wynikające z protokołu SOAP pochodzą ze standaryzacji w protokole SOAP do definiowania interfejsów aplikacji, a następnie wyboru transportów. Nie ma potrzeby pisania adapterów specyficznych dla aplikacji, a użytkownik może wybrać, czy jako konektor ma być używany produkt IBM WebSphere MQ , czy HTTP. Wybór transportu zależy od jakości usług i połączeń, które są wymagane.

W przypadku istniejących użytkowników protokołu SOAP korzystający z protokołu HTTP korzyść dla transportu WebSphere MQ dla protokołu SOAP jest dostarczana z użyciem zarządzanego i niezawodnego transportu asynchronicznego. Korzyści są dwojakie:

Prawdziwie asynchroniczny model programowania dostępności i wydajności.
Użycie asynchronicznego interfejsu klienta powoduje, że aplikacje klienckie i aplikacje nie muszą być dostępne w tym samym czasie. Żądania wysłane przez klienta będą przechowywane do momentu udostępnienia usługi w celu ich przetworzenia.

Gotowa zbudowana sieć zarządzana, która została zaprojektowana w taki sposób, aby była niezawodna i dostępna.

Wybierając produkt IBM WebSphere MQ jako transport, otrzymujesz korzyści z korzystania z sieci zarządzanej, która zapewnia niezawodne przesyłanie komunikatów.

Natomiast transporty, takie jak HTTP i FTP przez TCP/IP, są niezarządzane. Niezarządzana sieć jest idealna do nieprzewidywalnych połączeń: jest mniej zadań zarządzania.

Podsumowanie

Transport produktu IBM WebSphere MQ dla protokołu SOAP udostępnia następujące komponenty:

- Powiązanie transportu SOAP/JMS jest używane w dokumentach WSDL w celu powiązania usługi SOAP z transportem JMS. Implementacja powiązania SOAP/JMS produktu WebSphere MQ korzysta z identyfikatora URI, który ma jedną z dwóch form:

  Transport produktu WebSphere MQ dla protokołu SOAP

  ```
  jms:/queue?&Name=Value&Name=Value...
  ```

  Format łącznika produktu WebSphere MQ dla rekomendacji kandydata W3C

  ```
  jms:queue:qName?connectionFactory=connectQueueManager(qMgrName)&Name=Value&Name=Value...
  ```

- Odwzorowanie komunikatu SOAP na komunikat produktu WebSphere MQ .
- Dwa programy nasłuchujące IBM WebSphere MQ SOAP do odbierania żądań SOAP, jeden dla języka Java i jeden dla środowiska .NET Framework 1 lub .NET Framework 2. Programy nasłuchujące używają środowiska .NET lub Axis 1.4 do przetwarzania żądania SOAP.
- Dwa nadawcy SOAP IBM WebSphere MQ do tworzenia żądań SOAP IBM WebSphere MQ . Klienty usług Web Service rejestrują się z nadawcą w celu przetwarzania żądań SOAP `jms:` .
- Integracja z produktem Windows Communication Foundation (WCF), czasami nazywana .NET 3, w celu wysyłania i odbierania komunikatów WebSphere MQ Transport for SOAP.
- Integracja klienta z serwerem Axis2, czasem znanym jako JAX-WS, w celu wysyłania komunikatów WebSphere MQ Transport for SOAP lub W3C SOAP JMS.
- Komenda **amqwdeployWMQService**, która tworzy komponenty programistyczne i wykonawcze oraz skrypty w celu wdrożenia usługi Web Service przy użyciu transportu IBM WebSphere MQ dla protokołu SOAP.
- Przykładowy kod klienta i usługi Java oraz .NET.
- Skrypt do ustawienia ścieżki klasy i innych skryptów narzędziowych.

W środowiskach zintegrowanych program nasłuchujący i program nasłuchujący są integrowane z każdym środowiskiem, podobnie jak rozszerzenia narzędzi programistycznych i wdrażania.

Integracja protokołu SOAP i produktu WebSphere MQ

Transport produktu WebSphere MQ dla protokołu SOAP rozszerza protokół SOAP i narzędzia usług Web Service oraz środowisko wykonawcze za pomocą produktu WebSphere MQ jako alternatywnego transportu z protokołem HTTP dla protokołu SOAP. Nie ma potrzeby modyfikowania istniejących usług Web Service w celu używania transportu produktu WebSphere MQ dla protokołu SOAP jako transportu. W transporcie używany jest niestandardowy format identyfikatora URI dla SOAP/JMS. Format identyfikatora URI W3C dla SOAP/JMS jest obsługiwany w ograniczonym zakresie przez klienty Axis2 .

Dodatkowy wiersz kodu musi zostać dodany do klientów w środowiskach .NET Framework 1, .NET Framework 2 i Axis 1.4 . W klientach Axis 2 i Windows Communication Foundation (WCF) nie jest wymagany żaden dodatkowy kod. Program nasłuchujący SOAP WebSphere MQ uruchamia usługi w środowiskach .NET Framework 1, .NET Framework 2 i Axis 1.4 . Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z niektórymi innymi środowiskami serwera aplikacji, w tym WCF, CICS i WebSphere Application Server.

Co to jest SOAP?

SOAP[9]Opisuje standaryzowany format komunikatów i protokołów interakcji używanych przez aplikacje do wymiany żądań, odpowiedzi i datagramów. Protokół SOAP jest niezależny od transportu używanego do przesyłania komunikatów, a także do środowiska aplikacji, które wysyła i odbiera komunikaty. Usługa W3C definiuje wersję protokołu SOAP 1.2 w sposób zwięźle:

Wersja SOAP 1.2 udostępnia definicję informacji opartych na języku XML, które mogą być używane do wymiany informacji o strukturach i typie między rówieśnikami w zdecentralizowanym, rozproszonym środowisku.[10].

Aby można było używać protokołu SOAP, musi on być powiązany z transportem, takim jak HTTP, e-mail lub WebSphere MQ.

Struktura powiązania protokołu SOAP to zestaw reguł do przenoszenia komunikatu SOAP na początku innego protokołu, takiego jak HTTP. SOAP Version 1.2 Part 2: Adjuncts (Second Edition) opisuje powiązanie HTTP SOAP.

Rekomendacja kandydata W3C , 4 czerwca 2009 r., SOAP over Java Message Service 1.0, opisuje rekomendację dla powiązania SOAP JMS. Jako że usługa JMS jest specyfikacją interfejsu API, a nie protokołem transportowym, rekomendacja SOAP JMS nie opisuje formatu łącznika komunikatów SOAP JMS. Opisano w nim protokoły interakcji SOAP i powiązanie interfejsu API JMS. W związku z tym, jeśli jest używana rekomendacja JMS SOAP, należy nadal używać tej samej implementacji JMS dla klienta SOAP i serwera SOAP. Umożliwia ona uruchamianie aplikacji JMS JMS na dowolnej implementacji usługi JMS. Implementację JMS można podłączać do serwera aplikacji J2EE , jeśli zarówno serwer, jak i implementacja JMS są zgodne ze specyfikacją JCA. Produkt WebSphere MQ JMS jest zgodny ze specyfikacją JCA i może być podłączony do zgodnego serwera aplikacji.

Transport produktu WebSphere MQ dla powiązania SOAP jest podobny do proponowanego standardu W3C , ale nie jest on taki sam. Jego użycie jest opisane w temacie MQRFH2 SOAP settings(Ustawienia SOAP MQRFH2). W przeciwieństwie do rekomendacji kandydata W3C , powiązanie SOAP nie jest formalnie określone. W rzeczywistości jest to powiązanie HTTP, a adres usługi ma postać `jms:/queue?name=value&name=value...`, a nie `http://authority/path?query#fragment`. `jms:` nie jest oficjalnie zarejestrowanym schematem identyfikatora URI IANA.

Czym jest usługa WWW?

Protokół SOAP umożliwia programom napisanych w różnych językach, działającym na różnych platformach, komunikowanie się za pomocą różnych protokołów transportowych. SOAP jest specyfikacją

[9] Historycznie, akronim stał się dla Simple Object Access Protocol.
[10] W3C: SOAP, wersja 1.2 , część 0

protokołu. Usługa Web Service to aplikacja, która udostępnia usługę za pośrednictwem interfejsu SOAP, do którego dostęp można uzyskać za pomocą protokołów internetowych.

Ważnym celem SOAP jest zapewnienie usług, które klienci mogą łatwo wykorzystać. Po zaprojektowaniu klienta do korzystania z usługi można zaprogramować wywołanie w celu wywołania usługi bez odwołania się do zewnętrznej dokumentacji. Interfejsy usług są opisane w języku XML, w dokumencie WSDL. Zapytanie `http://authority/path?wsdl`zwraca opis WSDL usługi SOAP.

Wskazówka: Podczas wdrażania usługi Web Service w celu użycia produktu WebSphere MQnależy również wdrożyć usługę na potrzeby protokołu HTTP, tak aby standardowe zapytanie WSDL było działać.

Projektowanie usług Web Services

Usługi Web Service mają klienta i część usługi. Usługa jest zapisywana jako pierwsza, począwszy od opisu interfejsu w pliku WSDL, albo zgodnie z regułami pisania klasy usługi. Biblioteki narzędziowe usługi Web Service mają narzędzia do generowania pliku WSDL z definicji interfejsu klasy, na przykład **`java2wsdl`** lub **`disco`**. Mają również narzędzia do generowania szkieletów lub klas z opisów interfejsów WSDL, na przykład **`wsdl2java`**, **`wsimport`**lub **`wsdl`**. Ten pierwszy jest znany jako oddolny rozwój, a ten ostatni w dół.

Komenda **`amqwdeployWMQService`** w transporcie WebSphere MQ dla protokołu SOAP korzysta z tych narzędzi w celu wygenerowania kodu WSDL, kodów pośredniczących klienta i proxy klienta.

Usługi Web Services są zwykle zapisywane przy użyciu zintegrowanego środowiska programistycznego, które jest przeznaczone dla konkretnego środowiska serwera aplikacji:

Eclipse IDE for Java EE Developers
Tworzy usługi Web Services dla osi 2. Obsługuje JAX-RPC i JAX-WS

Produkt Rational Application Developer V7.5
Tworzy usługi Web Services dla serwera WebSphere Application Server V7 i poprzednich wersji, a także dla osi Axis. Obsługuje JAX-RPC i JAX-WS.

Produkt WebSphere Integration Developer V6.2
Tworzy usługi Web Service dla produktów WebSphere Process Server i WebSphere ESB. Obsługuje JAX-RPC i JAX-WS.

Visual Studio 2008 (wersja 9)
Tworzy usługi Web Services dla środowiska .NET Framework 3.5 i wcześniejszych wersji (Windows Communication Foundation)

Visual Studio 2005 (wersja 8)
Tworzy usługi Web Service dla środowiska .NET Framework 2 i wcześniejszych

Można użyć dowolnego z tych narzędzi w połączeniu z transportem WebSphere MQ dla protokołu SOAP. Po opracowaniu usługi, która ma być używana z protokołem HTTP, należy użyć narzędzia **`amqwdeployWMQService`** w celu wdrożenia usług w celu użycia produktu WebSphere MQ jako transportu. Można napisać nowy klient, korzystając z danych wyjściowych narzędzia, lub zmodyfikować istniejące klienty w taki sposób, aby korzystały z transportu WebSphere MQ dla protokołu SOAP.

Jeśli transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany ze środowiskiem aplikacji, wówczas nie trzeba używać narzędzia **`amqwdeployWMQService`** ani modyfikować kodu klienta. Warstwa SOAP klienta kieruje żądania klienta, które mają identyfikator URI z przedrostkiem `jms:` , do transportu WebSphere MQ dla protokołu SOAP. Warstwa SOAP serwera wywołuje transport produktu WebSphere MQ dla protokołu SOAP w celu oczekiwania na żądania SOAP `jms:` i zwraca odpowiedzi do produktu WebSphere MQ transport dla protokołu SOAP.

Zwykle usługi .NET zostały opracowane w dół przy użyciu adnotacji usługi Web Service w kodzie, a usługi Java z góry w dół, przy użyciu definicji interfejsu WSDL. Różnica w podejściach jest zawężana, ponieważ Java Standard Edition w wersji 6 obsługuje interfejs JAX-WS 2.0i korzysta z adnotacji w celu zakwalifikowania definicji interfejsów usług. Teraz jest tak łatwo rozwijać usługi Java u dołu, aż do góry nogami. To, które podejście wybrać, jest kwestią metody rozwoju.

Klient usług Web Services jest zapisywany po usłudze przy użyciu definicji usługi WSDL oraz wygenerowanych kodów pośredniczących klienta i proxy. W niektórych aplikacjach definicja usługi nie jest

znana, gdy klient jest zapisywany. Klient pobiera kod WSDL usługi i tworzy dynamicznie zlecenia usług. Bardziej powszechnie znana jest definicja usługi, ale adres, do którego usługa jest wdrażana, nie jest. Pakiet usług Web Service generuje interfejsy dla klienta, które mają być używane do wykonywania żądań usług. Klient udostępnia adres usługi, gdy jest on wymagany. W trzecim przypadku plik WSDL zawiera wszystkie informacje, których potrzebuje klient. Plik WSDL zawiera zarówno interfejs, jak i adres usługi. Kod wygenerowany przez pakiet usług Web Service zawiera wszystkie informacje wymagane przez klienta w celu wykonania żądań usługi.

Można użyć dowolnego z tych trzech stylów z transportem WebSphere MQ dla protokołu SOAP.

Środowiska aplikacji usług Web Service

Pakiety usług Web Service wymagają odwzorowania z definicji WSDL usługi na strumienie bajtów, które są przesyłane w żądaniach i odpowiedziach SOAP. Strumień bajtów jest zdefiniowany przez specyfikację SOAP i jest zawarty w kopercie SOAP. Koperta SOAP jest wyświetlana w produkcie Rysunek 166 na stronie 978.

```
<?xml version='1.0'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header> <!-- optional -->
<!-- headers... -->
</soap:Header>
<soap:Body>
<!-- payload or fault message -->
</soap:Body>
</soap:Envelope>
```

Rysunek 166. koperta SOAP

Odwzorowanie z koperty SOAP na powiązanie języka i z powrotem jest częścią standaryzowaną, a część jest prawnie zastrzeżona. Odwzorowanie ma podstawowe znaczenie dla architektury .NET i jest udostępniane jako część środowiska wykonawczego Common Language Runtime (CLR). Odwzorowanie jest standaryzowane w specyfikacji Java według specyfikacji JAX. Ponieważ odwzorowania Java są standaryzowane, klienty usługi Web Service Java i usługi są przenośne między różnymi środowiskami aplikacji Java. JAX-RPC (nazywany czasem JAX-WS 1.0) jest to odwzorowanie, które jest obecnie najbardziej używane w użyciu. Jest ona obsługiwana przez produkt Axis 1.4. JAX-WS (nazywany czasem JAX-WS 2.0) jest znacznie ulepszonym standardem i prawdopodobnie szybko zastępuje interfejs JAX-RPC. Interfejs JAX-WS jest obsługiwany przez produkt Axis 2.0. Produkt WebSphere MQ 7.0.1 nie obsługuje usług JAX-WS i Axis 2.

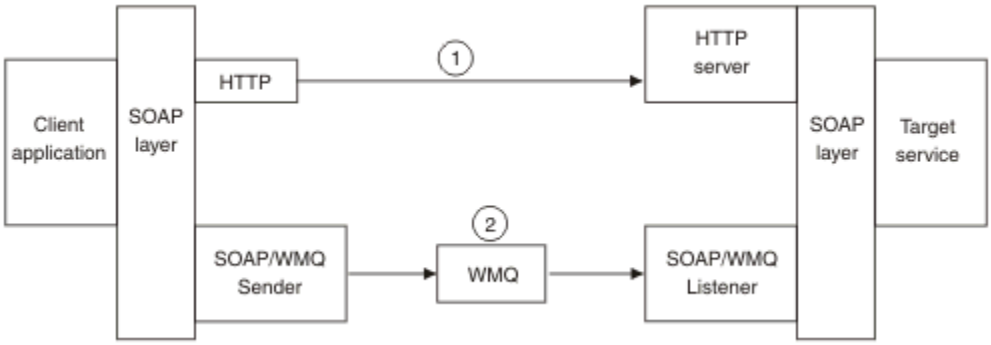
Transport produktu WebSphere MQ dla protokołu SOAP nie zmienia treści koperty SOAP, a treść nie ma wpływu na transport. Powiązania języka mają wpływ na transport produktu WebSphere MQ dla protokołu SOAP. Produkt WebSphere MQ 7.0.1 obsługuje środowisko .NET Framework 1, .NET Framework 2 i Axis 1.4 , korzystając z kodu i programów narzędziowych dostarczanych z produktem WebSphere MQ do transportu SOAP. Obsługa transportu WebSphere dla protokołu SOAP w środowisku .NET Framework 3 i 3.5 jest implementowana przy użyciu niestandardowego kanału produktu WebSphere MQ dla programu Windows Communication Foundation.

Inne środowiska programowania SOAP i środowiska wykonawczego mogą obsługiwać transport produktu WebSphere MQ dla protokołu SOAP i obsługiwać różne języki. Na przykład usługi Web Service działające w systemie CICS obsługują języki, takie jak COBOL i PL/1.

Uwaga: Używane odwzorowanie nie różni się od współdziałania usług Web Service. Istnieje możliwość mieszania i dopasowania klientów i usług napisanych przy użyciu odwzorowań .NET, JAX-RPC i JAX-WS.

Czym jest transport produktu WebSphere MQ dla protokołu SOAP?

Transport produktu WebSphere MQ dla protokołu SOAP jest powiązaniem SOAP i przybornikiem usług Web Service. Dzięki temu aplikacje umożliwiają wymianę komunikatów SOAP za pomocą produktu WebSphere MQ , a nie HTTP. Rysunek 167 na stronie 979 Przedstawia WebSphere MQ jako alternatywę dla protokołu HTTP jako transportu SOAP.

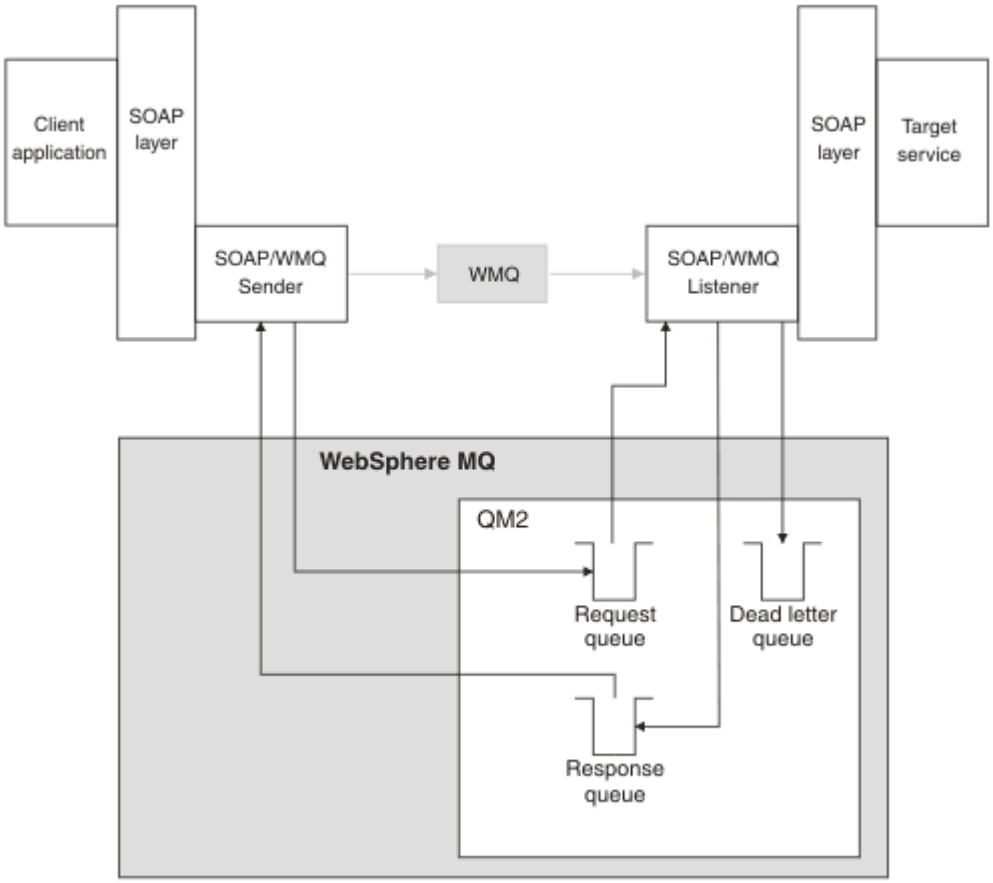


Rysunek 167. Przegląd transportu produktu WebSphere MQ dla protokołu SOAP

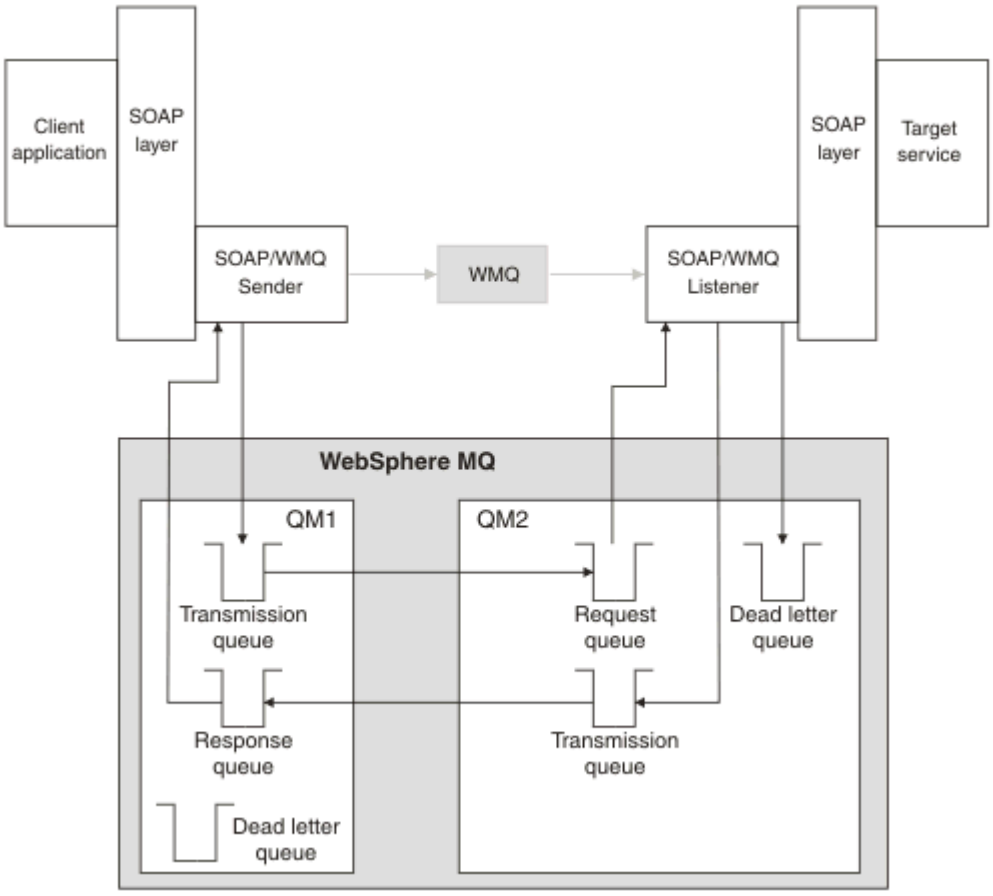
Protokół SOAP za pośrednictwem protokołu HTTP jest wyświetlany jako (1) na diagramie. Warstwa SOAP klienta przekształca żądanie w komunikat SOAP, a komponent HTTP wysyła za pośrednictwem protokołu TCP/IP. Komponent serwera HTTP nasłuchuje żądań HTTP, zwykle na porcie TCP/IP 80. Jeśli żądanie dotyczy usługi SOAP, komponent serwera HTTP wywołuje warstwę SOAP, aby przekształcić żądanie SOAP w wywołanie metody. Następnie zwracana jest odpowiedź.

Protokół SOAP w produkcie WebSphere MQ jest wyświetlany jako (2). Aplikacja kliencka rejestruje komponent nadawcy SOAP produktu WebSphere MQ jako procedurę obsługi dla protokołu `jms:` z warstwą SOAP. Warstwa SOAP przekazuje komunikaty SOAP adresowane do produktu `jms:` do nadawcy SOAP produktu WebSphere MQ . Nadawca używa identyfikatora URI w komunikacie, aby umieścić komunikat w kolejce żądań z wymaganymi cechami jakości usługi. Odpowiedni program nasłuchujący SOAP produktu WebSphere MQ oczekuje na komunikaty w swojej kolejce żądań i wywołuje warstwę SOAP w celu przetwarzania żądań i zwracania odpowiedzi.

Nadawcą i programem nasłuchującym SOAP są zwykłe programy produktu WebSphere MQ . Mogą one być połączone z tym samym menedżerem kolejek, co w programie Rysunek 168 na stronie 980, lub połączone z różnymi menedżerami kolejek; patrz Rysunek 169 na stronie 981. Klient może być połączony za pomocą połączenia klienckiego.



Rysunek 168. Kolejki używane przez protokół SOAP/WebSphere MQ (pojedynczy menedżer kolejek)



Rysunek 169. Kolejki używane przez protokół SOAP/WebSphere MQ (oddzielne menedżery kolejek)

Rekomendacja kandydata W3C dla powiązania SOAP z JMS.

Rekomendacja kandydatów W3C definiuje powiązanie protokołu SOAP korzystającego z usługi JMS (patrz: Protokół SOAP korzystający z usługi Java Message Service 1.0). Przydatny dla potrzeb przykładu jest również Schemat identyfikatora URI dla usługi Java (tm) Message Service 1.0[11].

Niektóre środowiska aplikacji, takie jak WebSphere Application Server v7, obsługują rekomendację kandydata W3C . Żądania SOAP są formatowane przy użyciu identyfikatora URI zgodnego z rekomendacją kandydata W3C przy użyciu klienta Axis2 . Patrz sekcja W3C Identyfikator URI protokołu SOAP over JMS dla klienta WebSphere MQ Axis 2 . Klient Axis2 wysyła żądanie SOAP sformatowane z użyciem transportu W3C lub WebSphere MQ dla protokołu SOAP w oparciu o identyfikator URI w żądaniu SOAP.

Obsługa klienta Axis2 dla rekomendacji W3C jest wprowadzana w pakiecie poprawek 7.0.1.3 . Obsługa innych klientów i programów nasłuchujących SOAP udostępnianych przez produkt WebSphere MQ nie jest udostępniana.

Pojęcia pokrewne

Implementacja transportu WebSphere dla protokołu SOAP w środowisku .NET Framework 1, .NET 2 i Axis 1.4
Użytkownik może chcieć napisać własny nadajnik i program nasłuchujący SOAP WebSphere MQ . Należy użyć implementacji transportu produktu WebSphere MQ dla protokołu SOAP w środowisku .NET Framework 1, .NET Framework 2 i Axis 1.4 jako przewodnika.

Transport produktu WebSphere MQ do niezawodnego przesyłania komunikatów dla usług SOAP i usług Web Service

[11] W specyfikacji produktu W3C należy wyszukać *Schemat identyfikatora URI dla usługi JMS*, aby uzyskać najnowszą wersję roboczą.

Niezawodne przesyłanie komunikatów usług Web Service jest protokołem do niezawodnego wymiany żądań i odpowiedzi usług Web Service przez niewiarygodne połączenie. Najlepiej nadaje się do rozwiązywania problemów związanych z krótkotrwowym zakłóceniem połączenia.

Implementacja transportu WebSphere dla protokołu SOAP w środowisku .NET Framework 1, .NET 2 i Axis 1.4

Użytkownik może chcieć napisać własny nadajnik i program nasłuchujący SOAP WebSphere MQ . Należy użyć implementacji transportu produktu WebSphere MQ dla protokołu SOAP w środowisku .NET Framework 1, .NET Framework 2 i Axis 1.4 jako przewodnika.

1. Program kliencki korzysta z odpowiednich struktur usług Web Services w taki sam sposób, jak w przypadku transportu HTTP. Musi także zarejestrować przedrostek `jms:` . Przedrostek jest rejestrowany przy użyciu metody języka Java produktu `com.ibm.mq.soap.Register.extension()` lub metody CLR `IBM.WMQSOAP.Register.Extension()` .
2. Środowisko Axis 1.4 lub środowisko .NET Framework 1 lub 2 zestawiło wywołanie do komunikatu żądania SOAP dokładnie tak, jak w przypadku protokołu SOAP/HTTP.
3. Usługa produktu WebSphere MQ jest identyfikowana za pomocą identyfikatora URI poprzedzonego przedrostkiem `jms:`. Gdy środowisko identyfikuje identyfikator URI produktu `jms:` , wywołuje on kod nadawcy transportu produktu WebSphere MQ ; `com.ibm.mq.soap.transport.jms.WMQSender` (dla osi 1.4) lub `IBM.WMQSOAP.MQWebRequest` (dla .NET1 i 2). Jeśli środowisko napotka identyfikator URI z przedrostkiem `http:` , wywołuje on standardowy nadawca protokołu SOAP przez HTTP.
4. Komunikat SOAP jest transportowany przez nadawcę SOAP WebSphere MQ przy użyciu kolejki żądań. **`SimpleJavaListener`** (dla środowiska Java) lub **`amqwSOAPNETListener`** (dla środowiska .NET) odbiera komunikat żądania.

   Programy nasłuchujące SOAP produktu WebSphere MQ są procesami autonomicznymi i są wielowątkowe z dostosowywalną liczbą wątków.
5. Program nasłuchujący SOAP produktu WebSphere MQ odczytuje przychodzące żądanie SOAP i przekazuje je do odpowiedniej infrastruktury usługi Web Service.
6. Infrastruktura usługi Web Service analizuje komunikat żądania SOAP i wywołuje usługę. Procedura jest taka sama, jak w przypadku komunikatu, który przybył do transportu HTTP.
7. Infrastruktura formatuje odpowiedź w komunikacie odpowiedzi SOAP i zwraca ją do programu nasłuchującego SOAP produktu WebSphere MQ .
8. Program nasłuchujący umieszcza komunikat w kolejce odpowiedzi, a komunikat jest przesyłany do nadawcy SOAP produktu WebSphere MQ . Nadawca przekazuje go do infrastruktury usługi Web Service klienta.
9. Infrastruktura klienta analizuje komunikat odpowiedzi SOAP i przekazuje wynik z powrotem do aplikacji klienckiej.

Każdy kontekst aplikacji jest obsługiwany przez osobną kolejkę żądań WebSphere MQ .

Kontekst aplikacji jest sterowany w osi 1.4 , upewniając się, że program nasłuchujący SOAP produktu WebSphere MQ i usługa są wykonywane w odpowiednim katalogu. Oś 1.4 ustawia poprawną zmienną CLASSPATH dla katalogu.

Kontekst aplikacji jest sterowany w środowisku .NET przez program nasłuchujący SOAP produktu WebSphere MQ wykonujący usługę w kontekście utworzonym przez wywołanie funkcji `ApplicationHost.CreateApplicationHost`. Wywołanie określa docelowy katalog wykonywania. Każda usługa następnie działa w katalogu, w którym został wdrożony.

Produkt **`amqwdeployWMQService`** generuje kolejki żądań i odpowiedzi. Generuje on również infrastrukturę niezbędną do obsługi kolejek i wdrażania usług w osi 1.4.

Pojęcia pokrewne

Integracja protokołu SOAP i produktu WebSphere MQ

Transport produktu WebSphere MQ do niezawodnego przesyłania komunikatów dla usług SOAP i usług Web Service
Niezawodne przesyłanie komunikatów usług Web Service jest protokołem do niezawodnego wymiany żądań i odpowiedzi usług Web Service przez niewiarygodne połączenie. Najlepiej nadaje się do rozwiązywania problemów związanych z krótkotrwowym zakłóceniem połączenia.

Transport produktu WebSphere MQ do niezawodnego przesyłania komunikatów dla usług SOAP i usług Web Service

Niezawodne przesyłanie komunikatów usług Web Service jest protokołem do niezawodnego wymiany żądań i odpowiedzi usług Web Service przez niewiarygodne połączenie. Najlepiej nadaje się do rozwiązywania problemów związanych z krótkotrwowym zakłóceniem połączenia.

Produkt WebSphere MQ for SOAP korzysta z zarządzanej i niezawodnej sieci produktu WebSphere MQ w celu przekazywania komunikatów SOAP. Transporty, takie jak HTTP i FTP, są niezarządzane. Sieci niezarządzane idealnie nadają się do nieprzewidywalnych połączeń, w których trudności i koszty zarządzania połączeniami przeważają nad korzyściami płynące z niepoluzowania żądań i odpowiedzi.

Aby rozwiązać problem luźnych plików, gdy połączenia przerwa się w niezarządzanych sieciach, usługi takie jak zarządzane FTP budują warstwę zarządzania na serwerze FTP. Warstwa zarządzania przejmuje ciężar sprawdzenia, czy pliki zostały pomyślnie przeniesione z użytkowników, a w razie potrzeby ponownie przesyła brakujące pliki. Aby korzystać z zarządzanego protokołu FTP, na obu końcach połączenia musi być zainstalowane oprogramowanie do zarządzania.

Niezawodne przesyłanie komunikatów usług Web Service (WSRM) wymaga innego podejścia do rozwiązania problemu niezawodnych połączeń. Jego celem jest niezawodne przesyłanie żądań i odpowiedzi usług Web Service bez konieczności użycia obu końców połączenia z tym samym oprogramowaniem. Każde oprogramowanie, implementując protokół niezawodnego przesyłania komunikatów usług Web Service, może niezawodnie wymieniać komunikaty przy użyciu innego oprogramowania.

Jeśli nawiązanie połączenia nie powiedzie się, nadawca i odbiorca muszą zachować kontekst przesyłania komunikatów WSRM, używając wygenerowanego identyfikatora URI jako klucza. Nadawca i odbiorca nadal próbują nawiązać nowe połączenie. Jeśli nowe połączenie zostanie pomyślnie nawiązane, przesyłanie zostanie zakończone. Specyfikacja WSRM nie określa, w jaki sposób kontekst jest zachowywane, czy też podczas próby nawiązania nowego połączenia.

Możesz zdecydować, że tylko krótkotrne przestoje są interesujące. W przypadku dłuższych wyłączeń można być przygotowanym do odrzucania transferów, których nie można było zrestartować po określonym czasie. Podobnie, można być przygotowanym do odrzucenia transferów, jeśli klient lub usługa nie powiedzie się. Pozostawiając użytkownika odpowiedzialnego za zapewnienie transferów, stawia coraz mniejsze wymagania w zakresie zarządzania koordynacją klienta i usługi.

Jeśli wyłączenia sieci są długotrwałe, w ciągu 30 minut lub w przypadku awarii klienta lub serwera, istnieje zwiększone prawdopodobieństwo, że niektóre połączenia nie zostaną ponownie nawiązane. Nie można już polegać na automatycznym odtwarzaniu przesyłania komunikatów w systemie WSRM w sposób niezarządzany. Konieczne jest rozważenie zarządzania nieudanym połączeniami WSRM, co oznacza rozwój oprogramowania do zarządzania siecią klientów i usług.

Korzystanie z aplikacji WSRM w celu pokonania krótkich wyłączeń może znacznie zmniejszyć liczbę transakcji dotyczących utraconych komunikatów w sieci komórkowej. Jeśli dostarczenie komunikatu nie jest konieczne, korzyści wynikające ze zmniejszenia utraty wiadomości mogą uzasadniać dodatkowe koszty związane z opracowaniu implementacji WSRM.

Protokół SOAP over JMS udostępnia gwarantowane dostarczanie komunikatów i zajmuje się dłuższą przerwami w czasie trwania klienta, serwera i sieci. Jeśli szukasz bardziej niezawodnej jakości usługi dla protokołu SOAP niż HTTP, które rozwiązanie jest dostępne: WebSphere MQ transport dla protokołu SOAP lub WSRM? Odpowiedź zależy od wielu czynników. Poniżej przedstawiono niektóre z czynników, które należy wziąć pod uwagę:

1. Określa, czy niezawodność jest spowodowana awarią połączenia.
2. Jak długo-bliskimi awariami połączeń są.

3. Jeśli możliwe jest zarządzanie zarówno stroną klienta, jak i po stronie serwera połączenia.
4. Jeśli użytkownik lub administrator ostatecznie jest odpowiedzialny za dostarczanie komunikatów.

Pojęcia pokrewne

Integracja protokołu SOAP i produktu WebSphere MQ

Implementacja transportu WebSphere dla protokołu SOAP w środowisku .NET Framework 1, .NET 2 i Axis 1.4

Użytkownik może chcieć napisać własny nadajnik i program nasłuchujący SOAP WebSphere MQ . Należy użyć implementacji transportu produktu WebSphere MQ dla protokołu SOAP w środowisku .NET Framework 1, .NET Framework 2 i Axis 1.4 jako przewodnika.

Instalowanie i weryfikowanie usług Web Service produktu WebSphere MQ

Aby zainstalować i zweryfikować transport produktu WebSphere MQ dla protokołu SOAP, należy skorzystać z instrukcji znajdujących się w tych tematach.

Instalowanie produktu WebSphere MQ Web transport dla protokołu SOAP

Poniższe instrukcje umożliwiają zainstalowanie transportu WWW produktu WebSphere MQ dla protokołu SOAP. Podczas instalacji tworzone są narzędzia służące do uruchamiania klientów usług Web Service lub usług za pomocą produktu WebSphere MQ jako transportu SOAP. Narzędzia te są używane w środowiskach .NET Framework 1, .NET 2, Axis 1.4lub Axis2 SOAP.

Zanim rozpoczniesz

Sprawdź wstępnie wymagane produkty pod adresem Wymagania systemowe produktu IBM WebSphere MQ. Proces instalacji nie sprawdza obecności i dostępności wstępnie wymaganego oprogramowania. Należy sprawdzić, czy zostały zainstalowane wymagania wstępne.

Produkt WebSphere MQ udostępnia kopię środowiska wykonawczego Axis 1.4 . Tej wersji należy używać razem z produktem WebSphere MQ , a nie z innymi, które mogły zostać zainstalowane. Firma IBM nie zapewnia wsparcia technicznego dla osi Apache . Skontaktuj się z Fundacją Apache Software Foundation, jeśli masz problemy techniczne z tym oprogramowaniem.

Aby można było uruchamiać usługi Web Service w środowisku SOAP środowiska .NET Framework 3, produkt WebSphere MQ korzysta z programu Windows Communication Foundation. Niestandardowy kanał produktu WebSphere MQ dla programu Windows Communication Foundation uruchamia klienty usługi Web Service i usługi przy użyciu produktu WebSphere MQ jako transportu dla komunikatów SOAP.

O tym zadaniu

Transport WWW produktu WebSphere MQ dla protokołu SOAP można zainstalować jako aplikację klienta MQI produktu WebSphere MQ lub aplikację serwera. Jeśli produkt WebSphere MQ jest już zainstalowany jako klient lub serwer na komputerze, należy sprawdzić, czy zostały zainstalowane komponenty wymienione na liście.

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

Wykonaj następujące kroki instalacji.

Procedura

1. Wybierz komponent "Java and. Net Messaging and Web services" do instalacji.
2. W systemach Solaris i HP-UXwybierz komponent "Środowisko Java Runtime Environment" do zainstalowania.
3. Wybierz pakiet Development Toolkit do zainstalowania.
4. Zainstaluj i zweryfikuj produkt WebSphere MQ zgodnie z opisem na stronie Szybki początek dla używanej platformy.

5. Skopiuj środowisko wykonawcze produktu Apache Axis 1.4 `axis.jar` z katalogu `prereqs/axis` na nośniku instalacyjnym produktu WebSphere MQ . Skopiuj go do katalogu instalacyjnego opisanego w sekcji Tabela 138 na stronie 985, Tabela 139 na stronie 985 lub Tabela 140 na stronie 985.

   Windows

   ```
   Copy D:\PreReqs\axis\axis.jar MQ_INSTALLATION_PATH\java\lib\soap
   ```

   AIX

   ```
   cp -f PreReqs/axis/axis.jar MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chown mqm:mqm MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chmod 444 MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   ```

   Katalogi instalacyjne HP-UX, Solaris i Linux (wszystkie platformy)

   ```
   cp -f PreReqs/axis/axis.jar MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chown mqm:mqm MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chmod 444 MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   ```

6. W systemie Windows 2003 uruchom program **Aspnet_regiis.exe** , aby zaktualizować odwzorowania skryptów tak, aby wskazywały na wersję używanego środowiska wykonawczego języka Common Language Run.

   Poszukaj programu narzędziowego **Aspnet_regiis.exe** w produkcie `%SystemRoot%Microsoft.NET/Framework/`*`version-number`* .

7. Ustaw zmienną środowiskową `WMQSOAP_HOME` , tak aby wskazywała na katalog instalacyjny produktu WebSphere MQ .

Wyniki

Tabela 138. Katalogi instalacyjne systemu Windows

Położenie	**Spis treści**
MQ_INSTALLATION_PATH`\programs\bin`	Pliki binarne, komendy, biblioteki DLL i pliki wykonywalne
MQ_INSTALLATION_PATH`\programs\java\lib`	`.jar files`
MQ_INSTALLATION_PATH`\programs\java\lib\soap`	SOAP `.jar files`
MQ_INSTALLATION_PATH`\programs\soap\samples`	Próbki i IVT

Tabela 139. Katalogi instalacyjne AIX

Położenie	**Spis treści**
MQ_INSTALLATION_PATH`/bin`	Skrypty powłoki
MQ_INSTALLATION_PATH`/java/lib`	`.jar files`
MQ_INSTALLATION_PATH`/java/lib/soap`	`axis.jar` i inne pliki JAX-RPC `.jar`
MQ_INSTALLATION_PATH`/samp/soap`	Próbki i IVT

Tabela 140. Katalogi instalacyjne HP-UX, Solaris i Linux (wszystkie platformy)

Położenie	**Spis treści**
MQ_INSTALLATION_PATH`/bin`	Skrypty powłoki
MQ_INSTALLATION_PATH`/java/lib`	`.jar files`

Tabela 140. Katalogi instalacyjne HP-UX, Solaris i Linux (wszystkie platformy) (kontynuacja)	
Położenie	**Spis treści**
MQ_INSTALLATION_PATH/java/lib/soap	axis.jar i inne pliki JAX-RPC .jar
MQ_INSTALLATION_PATH/samp/soap	Próbki i IVT

Co dalej

1. Tylko w przypadku środowiska .NET, należy zarejestrować transport produktu WebSphere MQ dla plików SOAP z globalną pamięcią podręczną zespołu. Jeśli środowisko .NET jest już zainstalowane podczas instalowania produktu WebSphere MQ, rejestracja jest przeprowadzana automatycznie podczas instalacji. Jeśli produkt .NET jest instalowany po WebSphere MQ, rejestracja jest przeprowadzana automatycznie po pierwszym uruchomieniu programu IVT.

   Produkt **amqiregisterdotnet.cmd** można uruchomić w celu przeprowadzenia rejestracji zespołów .NET. Produkt **amqiregisterdotnet.cmd** można również uruchomić w celu wymuszenia ponownej rejestracji na dowolnym etapie. Po dokonaniu tej rejestracji system zostanie zrestartowany, a następna ponowna rejestracja nie jest zwykle konieczna.
2. Uruchom test weryfikujący instalację zgodnie z opisem w sekcji "Weryfikowanie transportu produktu IBM WebSphere MQ dla protokołu SOAP" na stronie 986.
3. Jeśli planowane jest opracowanie klienta Axis2 , należy pobrać plik Axis2 1.4.1 z serwera Apache; patrz "Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse" na stronie 1010.

Weryfikowanie transportu produktu IBM WebSphere MQ dla protokołu SOAP

Sprawdź, czy transport IBM WebSphere MQ dla protokołu SOAP jest używany przy użyciu komendy **runivt** . Ta komenda uruchamia wiele aplikacji demonstracyjnych i zapewnia, że środowisko jest poprawnie skonfigurowane po instalacji.

Zanim rozpoczniesz

Przed uruchomieniem komendy **runivt** należy upewnić się, że dostępne są następujące środowiska wykonawcze:

- Aby uruchomić tylko w środowisku Axis: w systemie musi być dostępny pakiet Java SDK (w ramach SOE). Należy również uwzględnić położenie komend java.exe i javac.exe w zmiennej środowiskowej **PATH** systemu.
- Aby uruchomić test tylko w środowisku .NET (obsługiwany tylko w systemie Windows): w systemie musi być kompilator Java SDK oraz kompilatory i narzędzia .NET. W tym celu należy uzyskać dostęp do wiersza komend programu Visual Studio lub wiersza komend pakietu Microsoft Windows SDK, a następnie dodać położenie plików java.exe i javac.exe do zmiennej środowiskowej **PATH** .
- Aby uruchomić wszystkie dostępne testy: w przypadku platform Windows , środowisko musi zostać skonfigurowane zgodnie z opisem w wykonaniu testu .NET. W przypadku platform UNIX and Linux środowisko musi być skonfigurowane zgodnie z opisem w uruchomionym tylko teście Axis.

O tym zadaniu

Zamiast uruchamiać test weryfikacyjny zarówno w środowisku .NET, jak i w środowisku Axis, można uruchomić test tylko w środowisku Axis lub tylko w środowisku .NET.

Jeśli wystąpią problemy z testami i chcesz zacząć od nowa:

1. Zatrzymaj menedżer kolejek WMQSOAP.DEMO.QM za pomocą opcji immediate (immediate).
2. Zatrzymaj nasłuchiwanie, które zostało uruchomione w innym oknie.
3. Usuń menedżer kolejek.
4. Usuń katalog przykładów tymczasowych, który został utworzony i ponownie uruchom.

Na platformach UNIX and Linux należy uruchomić komendę przy użyciu sesji systemu X Windows .

Komenda **runivt** zmienia zawartość katalogu `soap/samples` . Aby zachować obraz instalacyjny bez zmian, skopiuj katalog przykładów do położenia tymczasowego i uruchom test weryfikacyjny z położenia tymczasowego.

Weryfikację instalacji można uruchomić tyle razy, ile chcesz.

Wykonaj następujące kroki, aby zweryfikować instalację transportu produktu IBM WebSphere MQ dla protokołu SOAP w środowisku .NET Framework 1, .NET Framework 2 i Axis 1.4:

Procedura

1. Skopiuj drzewo katalogów `./tools/soap/samples` do położenia tymczasowego.
2. Uruchom okno komend z katalogiem tymczasowym jako bieżącym.
3. Użyj komendy **runivt** , aby uruchomić test instalacji. Skrypt `runivt` kompiluje klasę testową, przykładową klienta i usługi przed ich wdrożeniem i uruchomieniem. W przypadku klasy testowej, przykładowego klienta i usług, które mają zostać uruchomione, wykonaj kroki instalacji opisane w sekcji Instalowanie produktu WebSphere(r) MQ Web transport for SOAP i upewnij się, że wiersz komend używany do uruchamiania komendy `runivt` ma ustawiony wymagany środowisko wykonawcze. Aby uruchomić komendę **runivt** , należy użyć dowolnej z następujących metod:
   - Uruchom test tylko w środowisku Axis: `runivt Axis`.
   - Uruchom test tylko w środowisku .NET (obsługiwany tylko w systemie Windows): `runivt DotNet`.
   - Uruchom wszystkie dostępne testy:`runivt`.

   Więcej informacji na temat składni komend i parametrów komendy `runivt` zawiera sekcja **runivt**: transport produktu IBM WebSphere MQ do sprawdzania poprawności instalacji SOAP. Testy, które można uruchomić, są wymienione w pliku `ivttests.txt` w systemach Windows i `ivttests_unix.txt` na platformach UNIX and Linux .

Odsyłacze pokrewne

runivt: transport produktu WebSphere MQ do sprawdzania poprawności instalacji SOAP

Projektowanie usług Web Service dla transportu WebSphere MQ dla protokołu SOAP

Użyj normalnego środowiska programistycznego usług WWW do tworzenia usług, które będą używane z transportem WebSphere MQ dla protokołu SOAP.

Zanim rozpoczniesz

1. Jeśli planowane jest korzystanie z narzędzi wiersza komend dostarczanych z produktem WebSphere MQ transport for SOAP:
   a. Utwórz katalog wdrażania dla usługi.
   b. Uruchom okno komend w katalogu.
   c. W przypadku środowiska .NET, pliki csc.exe i wsdl.exe muszą znajdować się w ścieżce i muszą być z tej samej wersji środowiska .NET Framework.
   d. Dla języka Java,
      i) Uruchom komendę **amqwsetcp** , aby skonfigurować ścieżkę klasy.
      ii) Środowisko IBM JRE i pakiet JDK na tym samym poziomie wersji muszą znajdować się w bieżącej ścieżce. Poziom wersji musi być co najmniej 5.0.
      iii) Dostosuj ścieżkę klasy tak, aby uwzględniała położenia wszystkich dodatkowych bibliotek produktu `.jar` oraz katalogów zawierających pakiety .java, w tym również dla rozwijającej się usługi. Umieść bieżący katalog "`.`" w ścieżce klasy.

iv) Utwórz katalog w odniesieniu do bieżącego katalogu okna komend, odpowiadającego nazwie pakietu, który jest tworzony przez użytkownika.

2. Alternatywnie można użyć narzędzi środowiska roboczego, które obsługują projektowanie usług Web Services. Przykładowe zadania programistyczne to: Microsoft Visual Studio 2008, Eclipse IDE for Java EE Developers oraz WebSphere Application Server Community Edition.

O tym zadaniu

Istniejące usługi Web Service nie wymagają modyfikacji w celu pracy z transportem WebSphere dla protokołu SOAP. Narzędzia dostarczane z produktem WebSphere MQ transport for SOAP wdrażają usługę Web Service i uruchamiają ją przy użyciu nasłuchiwania SOAP produktu WebSphere MQ . Narzędzia generują również pliki WSDL, kody pośredniczące klienta .NET i klasy proxy produktu `.java` w celu utworzenia transportu produktu WebSphere MQ dla klientów SOAP.

Wykonaj poniższe kroki, aby utworzyć usługę i przygotować ją do wdrożenia i generowania klientów. Aby utworzyć usługę przy użyciu produktu Eclipse lub Microsoft Visual Studio 2008, należy wykonać kroki opisane w sekcji czynności pokrewnych.

Procedura

1. Należy utworzyć usługę przy użyciu normalnego środowiska programistycznego.
2. Przetestuj usługę przy użyciu klienta usług Web Service HTTP
3. Aby przygotować katalog wdrażania, wykonaj następujące kroki:
   - Dla języka Java
     a. Skopiuj plik `.java` definiujący interfejs usługi do katalogu wdrożenia.
     b. Skopiuj wszystkie pliki `.class` dla usługi do katalogu odpowiadającego nazwie pakietu.
     c. Sprawdź, czy ścieżka klasy może znaleźć wszystkie wymagane klasy: skompilować plik `.java` usługi przy użyciu produktu **javac**.
   - Dla środowiska .NET
     a. Skopiuj plik `.asmx` definiujący usługę do katalogu wdrożenia.
     b. Jeśli używany jest model za pomocą kodu, skopiuj wszystkie pliki `.dll` do katalogu *deployment directory*`\bin` .

Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse

Zaprojektuj usługę Web Service Axis 1.4 , która będzie uruchamiana przy użyciu produktu WebSphere MQ jako dostawcy usług. Użyj normalnego środowiska programistycznego usług Web Service w celu utworzenia usługi dla wdrożenia w osi 1.4.

Zanim rozpoczniesz

Należy wziąć pod uwagę wymagania dotyczące wdrażania serwera WWW w produkcie WebSphere MQ SOAP nasłuchiwania dla osi 1.4.

- Program nasłuchujący SOAP produktu WebSphere MQ dla środowiska Axis 1.4 wymaga środowiska JRE firmy IBM w wersji 5.0 lub nowszej. Środowisko JRE i pakiet JDK używane na potrzeby programowania muszą być na tym samym poziomie wersji.
- Program nasłuchujący SOAP produktu WebSphere MQ dla środowiska Axis 1.4 wymaga zainstalowania produktu `axis.jar` z produktem WebSphere MQ. Zmień ścieżkę budowania w środowisku programistycznym, aby odwoła się do pliku `axis.jar` zainstalowanego razem z produktem WebSphere MQ, a nie z plikami produktu `axis.jar` zainstalowanymi w środowisku programistycznym.
- Up to, and including, WebSphere MQ V7.0.1, the WSDL generated for the deployed service is RPC/encoded. W wersji V7.1można również zażądać pliku WSDL w stylu RPC/literał. Wygenerowany plik

WSDL jest używany tylko do wdrażania. Usługę można zdefiniować za pomocą pliku WSDL zgodnego ze standardem WS-I.

O tym zadaniu

Utwórz usługę przy użyciu normalnego środowiska programistycznego usług Web Service.

W tym zadaniu korzystamy z swobodnie otwieranego źródła Eclipse Java EE IDE for Web Developers, znanego pod nazwą Galileo. W przypadku serwera aplikacji używamy produktu WebSphere Application Server Community Edition v2.1 (Community Edition) w oparciu o Geronimo. Informacje na temat uzyskiwania, instalowania i konfigurowania środowiska IDE i serwera można znaleźć w pokrewnych zadaniach.

Przetestuj usługę, używając protokołu HTTP jako transportu, korzystając z Eksploratora usług Web Services udostępnionego w środowisku IDE. Alternatywnie wygeneruj serwer proxy klienta HTTP i przetestuj usługę przy użyciu własnego kodu klienta.

Aby zaprojektować oddolną usługę Web Service, należy wykonać następujące kroki. Użyj przykładowego programu `StockQuoteAxis.java` jako przykładu.

Procedura

1. Uruchom środowisko Eclipse IDE for Java EE Developers z nowym obszarem roboczym.
2. Skonfiguruj obszar roboczy tak, aby używany był Java50,
   Produkt WebSphere Application Server Community Edition 2.1.4 nie działa z produktem Java60.
   a) **Okno** > **Preferencje** > **Java** > **Zainstalowane środowiska JRE** > **Dodaj ...** > **Standardowa maszyna VM** > **Dalej** > **Katalog ...**
   b) Przejdź do katalogu instalacyjnego produktu **Java50** > **OK** > **Zakończ**
   c) Sprawdź środowisko **Java50** JRE > OK
3. Dodaj środowisko wykonawcze Community Edition i uruchom Community Edition.
   a) **Okno** > **Preferencje** > **Serwer** > **Środowiska wykonawcze** > **Dodaj ...**
   b) Wybierz opcję **IBM WASCE v2.1** z listy **New Server Runtime Environment** (Nowe środowisko wykonawcze serwera) > Check **Create a new local server** > **Dalej**
      Jeśli produkt **IBM WASCE 2.1** nie znajduje się na liście, należy wykonać dwa inne zadania:
      i) Zainstaluj produkt WebSphere Application Server Community Edition.
      ii) Zainstaluj aktualizację Eclipse dla produktu Community Edition.
      Szczegółowe informacje można znaleźć w sekcji WebSphere Serwer aplikacji Community Edition
   c) Przejdź do katalogu instalacyjnego serwera aplikacji > **OK** > **Zakończ** > **OK**.
   d) Kliknij prawym przyciskiem myszy serwer **IBM WASCE v2.1** w widoku serwerów > **Uruchom** .
      Wskazówka: WASCE można zarządzać w środowisku Eclipse: kliknij prawym przyciskiem myszy opcję **IBM WASCE v2.1 server** > **Uruchom konsolę WASCE** . the default **Username** and **Password** are `system` and `manager` .
4. Skonfiguruj serwer i środowisko wykonawcze dla usług Web Service.
   a) **Okno** > **Preferencje** > **Usługi Web Service** > **Serwer i środowisko wykonawcze**
   b) Wybierz serwer **IBM WASCE v2.1 Server** jako serwer.
   c) Pozostaw środowisko **Apache Axis** jako środowisko wykonawcze usługi Web Service.
5. Utwórz dynamiczny projekt WWW.
   a) **Plik** > **Nowy** > **Dynamiczny projekt WWW**.
      Nadaj nazwę projektowi `StockQuoteAxis`.
   b) Zaznacz pole wyboru **Dodaj projekt do pliku EAR** > **Nowy ...**
   c) Na stronie **Projekt aplikacji EAR** wpisz **Project name** `StockQuoteAxisEAR` > **Zakończ** .

Odpowiedz **OK** w odpowiedzi na to okno dialogowe sugerujące przełączenie się do perspektywy Java EE lub pozostań w perspektywie Java, aby dokładnie wykonać te instrukcje.

d) Opcja **IBM WASCE 2.1 server** jest wybrana jako docelowe środowisko wykonawcze. Zaakceptuj ją, a pozostałe wartości domyślne > **Finish**(Zakończ).

Odpowiedz **OK** w odpowiedzi na to okno dialogowe sugerujące przełączenie się do perspektywy Java EE lub pozostań w perspektywie Java, aby dokładnie wykonać te instrukcje.

6. Zaimportuj przykładowy program `StockQuoteAxis.java`

a) Otwórz projekt WWW **StockQuoteAxis** > Kliknij prawym przyciskiem myszy folder **`src`** > **Importuj ...**

b) Wybierz opcję **Ogólne** > **System plików** > **Dalej** .

c) Przejdź do opcji *`MQ_INSTALLATION_PATH`*`\tools\soap\samples\java\server` > zaznacz opcję **`StockQuoteAxis.java`** > **Zakończ**

`MQ_INSTALLATION_PATH` reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ . Aby wyświetlić zawarte w nim pliki, należy podświetlić katalog serwera.

7. Popraw błąd kompilacji, przenosując produkt `StockQuoteAxis.java` do jego poprawnego pakietu.

a) Otwórz `StockQuoteAxis.java` i kliknij prawym przyciskiem myszy problem > **Szybka poprawka** .

b) Kliknij dwukrotnie opcję **Move 'StockQuoteAxis.java' do pakietu 'soap.server'** > **Save**.

8. Utwórz usługę Web Service z produktu `StockQuoteAxis.java` .

a) Kliknij prawym przyciskiem myszy opcję `StockQuoteAxis.java` > **Usługi Web Service** > **Utwórz usługę Web Service** > **Dalej**.

Zaakceptuj domyślną konfigurację dla usługi:

Typ usługi Web Service
Ustępuj usługę Java beanWeb

Implementacja usługi
soap.server.StockQuoteAxis

Serwer
`IBM WASCE v2.1 server`

Środowisko wykonawcze usługi Web Service
Oś Apache

projekt usługi
Oś StockQuote

Projekt EAR usługi
StockQuoteAxisEAR

Konfiguracja
Brak generowania klienta

9. Wybierz metody dostępu i styl usługi Web Service > **Dalej**.

Jeśli zostanie wyświetlone zapytanie, uruchom serwer.

a) Pozostaw wszystkie wybrane metody.

b) Wybierz styl document/literal (wrapped).

10. **Zakończ**

Po wdrożeniu usługi zajrzyj do folderu `WebContent\wsdl` w projekcie WWW `StockQuoteAxis` i znajdź wygenerowany plik `StockQuoteAxis.wsdl` .

11. Przetestuj usługę przy użyciu protokołu HTTP przy użyciu Eksploratora usług Web Services.

a) Kliknij prawym przyciskiem myszy opcję `StockQuoteAxis.wsdl` > **Testuj za pomocą eksploratora usług Web Services**.

b) Kliknij operację **getQuote** w czynności **StockQuoteAxisSoapBinding** w oknie **Eksplorator usług Web Services** .

c) Wpisz `ibm` w polu wejściowym **symbol** > **Wykonaj** .

12. Przetestuj usługę przy użyciu transportu WebSphere MQ dla protokołu SOAP.

    Aby wdrożyć usługę, należy użyć produktu **`SimpleJavaListener`**, który znajduje się w sekcji `com.ibm.mq.soap.jar`. Do ścieżki budowania należy dodać biblioteki języka Java i SOAP produktu WebSphere MQ .

    a) Kliknij prawym przyciskiem myszy projekt WWW **StockQuoteAxis** > **Ścieżka budowania** > **Konfiguruj ścieżkę budowania ...**

    b) Kliknij kartę **Libraries** (Biblioteki) > **Add External Jars ...**(Dodaj zewnętrzne pliki JAR ...) Przejdź do sekcji *`MQ_INSTALLATION_PATH`*`\java\lib` . i wybierz wszystkie pliki `.jar` > **Otwórz** > **Dodaj zewnętrzne pliki JAR ...**. Przejdź do katalogu *`WMQ Install directory`*`\java\lib\soap` i wybierz wszystkie pliki `.jar` > **Otwórz** > **OK**.

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

    c) W eksploratorze projektów kliknij prawym przyciskiem myszy opcję `StockQuoteAxis\Java Resources\Libraries\com.ibm.mq.soap.jar\com.ibm.mq.soap.transport.jms\SimpleJavaListener.class\ SimpleJavaListener` > **Uruchom jako ...**> **Uruchom konfiguracje ...**

    Wskazówka:

    Jeśli nie ma konfiguracji dla produktu `SimpleJavaListener` , należy kliknąć ikonę **Nowa konfiguracja** na stronie **Tworzenie, zarządzanie i uruchamianie konfiguracji** kreatora **Uruchamianie konfiguracji** .
    `SimpleJavaListener` nie ma komendy, aby ją zatrzymać. Aby monitorować lub zatrzymać produkt **`SimpleJavaListener`**, otwórz **perspektywę debugowania** w środowisku Eclipse.

    d) Otwórz kartę **(x) = Argumenty** . W obszarze wejściowym **Argumenty programu** wpisz parametry do **`SimpleJavaListener`**.

    W tym przykładzie wpisz

    ```
    -u "jms:/queue?destination=REQUESTAXIS@QM1&connectionFactory=()
    &initialContextFactory=com.ibm.mq.jms.Nojndi
    &targetService=StockQuoteAxis
    &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE" -n 10
    ```

    Uwaga: Usługa docelowa to `StockQuoteAxis` , aby była zgodna z nazwą usługi docelowej utworzoną w deskryptorze wdrażania usługi `StockQuoteAxis\WebContent\WEB-INF\server-config.wsdd` . Usługa amqwdeployWMQService tworzy usługę docelową o nazwie `soap.server.StockQuoteAxis`. W tym przykładzie używany jest ten sam serwer `StockQuoteAxis.class` i serwer `service-config.wsdd` , co serwer HTTP.

    e) Na tej samej karcie należy skonfigurować opcję **Katalog roboczy** , aby odwołać się do pliku `server-config.wsdd` :
    `${workspace_loc:StockQuoteAxis/WebContent/WEB-INF}`

    a) **Uruchom**

    Błędy są zapisywane w konsoli. Jeśli konsola pozostanie pusta, program **`SimpleJavaListener`** uruchomił ok.

    b) Aby przetestować wdrożenie, należy uruchomić klienta Axis StockQuote, który został opracowany w ramach zadania “Projektowanie klienta JAX-RPC dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 1002.

Przykład: przykładowy program StockQuoteAxis

Przykładowa usługa Java Web Service `StockQuoteAxis.java`jest instalowana w produkcie *`WMQ install directory`*`\tools\soap\samples\java\server` . Produkt `StockQuoteAxis.java`, Rysunek 170 na stronie 992, ma cztery metody:

1. `float getQuote(String symbol)`

2. `void getQuoteOneWay(String symbol).`
3. `int asyncQuote(int delay)`
4. `float getQuoteTran(String symbol)`

```
package soap.server;
import java.lang.Thread;
import java.io.FileWriter;
public class StockQuoteAxis {
    public float getQuote(String symbol) throws Exception {
        return ((float) 55.25);
    }
    public void getQuoteOneWay(String symbol) throws Exception {
        try {
            // Write the results for this service to a file
            FileWriter f = new FileWriter("getQuoteOneWay.txt", true);
            f.write("One way service result via proxy is: 44.44\n");
            f.close();
        } catch (Exception ee) {
            System.out.println("Error writing result file in getQuoteOneWay");
            ee.printStackTrace();
        }
    }
    public int asyncQuote(int delay) {
        try {
            Thread.sleep(delay);
        } catch (Exception e) {
            System.out.println("Exception in asyncQuote during sleep");
        }
        return delay;
    }

    public float getQuoteTran(String symbol) throws Exception {
        if (symbol.equalsIgnoreCase("ROLLBACK")) {
            System.out.println("Rollback was requested,
                exiting from service by calling System.exit().");
            System.exit(0);
        }
        return ((float) 55.25);
    }
}
```

Rysunek 170. Oś StockQuote

Co dalej

Przeprowadź wdrożenie usługi przy użyciu produktu WebSphere MQ Transport for SOAP, a nie HTTP przy użyciu komendy **amqwdeployWMQService**.

Komenda ta ma opcję `axisDeploy`, która wdraża usługę, tworząc deskryptor wdrażania Apache Axis 1.4 . Usługa nasłuchiwania SOAP produktu WebSphere MQ uruchamia usługę. Obiekt nasłuchiwania SOAP nosi nazwę SimpleJavaListener i jest dostarczany wraz z transportem WebSphere MQ dla protokołu SOAP.

Zadania pokrewne

Projektowanie usługi .NET 1 lub 2 dla transportu WebSphere MQ dla protokołu SOAP za pomocą programu Microsoft Visual Studio 2008
Utwórz usługę Web Service SampleStockQuote dla platformy .NET 1 lub .NET 2 przy użyciu programu Microsoft Visual Studio 2008.

Projektowanie usługi Web Service JAX-WS komponentu EJB dla usługi W3C SOAP over JMS
Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji JEE. Ta czynność to krok 2 połączenia klienta usługi Web Service Axis2 z usługą Web Service wdrożoną na serwerze WebSphere Application Server przy użyciu protokołu SOAP W3C SOAP over JMS.

Projektowanie usługi .NET 1 lub 2 dla transportu WebSphere MQ dla protokołu SOAP za pomocą programu Microsoft Visual Studio 2008

Utwórz usługę Web Service SampleStockQuote dla platformy .NET 1 lub .NET 2 przy użyciu programu Microsoft Visual Studio 2008.

O tym zadaniu

Utwórz usługę StockQuote z implementacją kodu za pomocą narzędzia Visual Studio 2008.

Procedura

1. Utwórz szablon dla usługi, a następnie sprawdź, czy działa on na serwerze HTTP.
   a) Uruchom program Visual Studio 2008 > **Plik** > **Nowy** > **Projekt ...**. Wybierz opcję **C#** Typ projektu, **, NET Framework 2**i **ASP.NET Web Service Application**. Wpisz **Nazwę:** i **Nazwa rozwiązania:** `StockQuoteDotNet` > **OK**
   b) Kliknij prawym przyciskiem myszy pozycję **Service1.asmx** w oknie **Eksplorator rozwiązań** > **Zmień nazwę** > `StockQuote.asmx`.
   c) Zmień fragment kodu `public class Service1` na `public class StockQuote`.
   d) Kliknij prawym przyciskiem myszy opcję **StockQuote.asmx** w oknie **Eksplorator rozwiązań** > **Otwórz za pomocą ...** > **Edytor XML**. Zmień `Class="StockQuoteDotNet.Service1"` na `Class="StockQuoteDotNet.StockQuote"`
   e) Zmień fragment kodu `[WebService(Namespace = "http://tempuri.org/")]` na `[WebService(Namespace = "http://stock.samples/")]`.
   f) Usuń wiersz kodu `[ToolboxItem(false)]`.
   g) Sprawdź, czy wszystko jest poprawne: **Debuguj** > **Uruchom debugowanie (F5)**. Sprawdź dane wyjściowe w eksploratorze.
2. Dodaj metody z przykładu `SQDNNonInline.asmx.cs`i przetestuj usługę na serwerze HTTP.
   a) Otwórz program *MQ_INSTALLATION_PATH*`\tools\soap\samples\dotnet\SQDNNonInline.asmx.cs` i zastąp metodę `HelloWorld` czterema metodami `Quote` . Patrz sekcja Rysunek 171 na stronie 995. *MQ_INSTALLATION_PATH* Reprezentuje katalog, w którym zainstalowano produkt WebSphere MQ .
   b) **Buduj** > **Przebuduj** rozwiązanie > Kliknij prawym przyciskiem myszy jeden z opcji **Wątek** w przypadku błędu > **Rozstrzygnij** > Za pomocą metody **System.Threading**.
   c) Naciśnij klawisz F5 , aby rozpocząć debugowanie.

      Usługa nie jest zgodny ze standardem WS-I Basic Profile v1.1. Użytkownik może zmienić adnotację WebMethod z programu `[SoapRpcMethod]` na wartość `[SoapDocumentMethod]`lub usunąć adnotację `[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]`.
   d) Naciśnij klawisz F5 , aby sprawdzić implementację za pomocą protokołu HTTP.
3. Wygeneruj plik WSDL, klienty i uruchom usługę przy użyciu transportu WebSphere MQ dla protokołu SOAP.
   a) Otwórz okno komend w drzewie katalogów projektu, w którym przechowywana jest `StockQuote.asmx` .
   b) (Opcjonalnie) Użyj komendy amqwdeployWMQService , aby wygenerować artefakty. Menedżer kolejek musi być uruchomiony:

      ```
      amqswdeployWMQService -f StockQuote.asmx
      -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
      &connectionFactory=()
      &destination=REQUESTDOTNET@QM1
      &targetService=StockQuote.asmx"
      StockQuote.asmx StockQuote.wsdl
      ```

      Wszystkie artefakty są tworzone w drzewie katalogów produktu `./generated` .

c) (Opcjonalnie) Wygeneruj tylko plik WSDL w celu wywołania usługi przy użyciu transportu WebSphere MQ dla protokołu SOAP.

```
amqswsdl -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=()
&destination=REQUESTDOTNET@QM1
&targetService=StockQuote.asmx"
StockQuote.asmx StockQuote.wsdl
```

a) Uruchom program nasłuchujący .NET. Użyj komendy `.\generated\server\startWMQNListener.cmd` lub wpisz komendę:

```
amqSOAPNETListener -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=()
&destination=REQUESTDOTNET@QM1
&targetService=StockQuote.asmx"
```

4. Przetestuj usługę przy użyciu klienta wygenerowanego z pliku WSDL lub korzystając z klientów wygenerowanych przez produkt **amqwdeployWMQService**.

Kod przykładowy

Przykładowa usługa WWW .NET (`StockQuoteDotNet`) jest instalowana w produkcie *MQ_INSTALLATION_PATH*`\tools\soap\samples\dotnet`. *MQ_INSTALLATION_PATH* Jest to katalog, w którym zainstalowano produkt WebSphere MQ . Powiązanie usługi Web Service opublikowanych przykładów różni się nieznacznie od powiązania użytego w zadaniu. W zadaniu używane są wartości domyślne używane w Visual Studio 2008.

Istnieją dwa przykłady usług Web Service środowiska .NET Framework 1 i .NET Framework 2. `StockQuoteDotNet.asmx`, jest usługą wstawianą. `SQDNNoninline.asmx`, jest kodem-za usługą Web Service zaimplementowaną przez produkt `SQDNNoninline.asmx.cs`.

Produkt `StockQuoteDotNet` ma cztery metody:

1. `float getQuote(String symbol)`
2. `void getQuoteOneWay(String symbol)`.
3. `int asyncQuote(int delay)`
4. `float getQuoteDOC(String symbol)`

```
<%@ WebService Language="C#" Class="StockQuoteDotNet" %>
using System;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Services.Description;
using System.Threading;
[WebService (Namespace="http://stock.samples")]
public class StockQuoteDotNet {
     [WebMethod] [ SoapRpcMethod(OneWay=true) ]
     public void getQuoteOneWay(String symbol) {
          if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(5000);
          System.Console.WriteLine("getQuoteOneWay was invoked.");
     }
     [WebMethod] [SoapRpcMethod]
     public float getQuote(String symbol) {
          if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(10000);
          return 88.88F;
     }
     [WebMethod] [SoapRpcMethod]
     public int asyncQuote(int delay) {
          Thread.Sleep(delay);
          return delay;
     }
     [WebMethod]
     public float getQuoteDOC(String symbol) {
          return 77.77F;
     }
}
```

Rysunek 171. Usługa wstawiana: StockQuoteDotNet.asmx

```
<%@ WebService Language="C#" Codebehind="SQDNNonInline.asmx.cs" Class="SQDNNonInline" %>
```

Rysunek 172. Kod-za: Projekt SQDNNonInline.asmx

```
using System;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Services.Description;
using System.Threading;

[WebService(Namespace = "http://stock.samples")]
public class SQDNNonInline : System.Web.Services.Protocols.SoapHttpClientProtocol
{

    [WebMethod]
    [SoapRpcMethod(OneWay = true)]
    public void getNonInlineQuoteOneWay(String symbol)
    {
        if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(5000);
        System.Console.WriteLine("getNonInlineQuoteOneWay was invoked.");
    }

    [WebMethod]
    [SoapRpcMethod]
    public float getNonInlineQuote(String symbol)
    {
        if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(10000);
        return 88.88F;
    }

    [WebMethod]
    [SoapRpcMethod]
    public int asyncNonInlineQuote(int delay)
    {
        Thread.Sleep(delay);
        return delay;
    }
    [WebMethod]
    public float getNonInlineQuoteDOC(String symbol)
    {
        return 77.77F;
    }
}
```

Rysunek 173. Kod-za: Implementacja: SQDNNonInline.asmx.cs

Zadania pokrewne

Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse
Zaprojektuj usługę Web Service Axis 1.4 , która będzie uruchamiana przy użyciu produktu WebSphere MQ jako dostawcy usług. Użyj normalnego środowiska programistycznego usług Web Service w celu utworzenia usługi dla wdrożenia w osi 1.4.

Projektowanie usługi Web Service JAX-WS komponentu EJB dla usługi W3C SOAP over JMS
Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji JEE. Ta czynność to krok 2 połączenia klienta usługi Web Service Axis2 z usługą Web Service wdrożoną na serwerze WebSphere Application Server przy użyciu protokołu SOAP W3C SOAP over JMS.

Projektowanie usługi Web Service JAX-WS komponentu EJB dla usługi W3C SOAP over JMS

Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji JEE. Ta czynność to krok 2 połączenia klienta usługi Web Service Axis2 z usługą Web Service wdrożoną na serwerze WebSphere Application Server przy użyciu protokołu SOAP W3C SOAP over JMS.

Zanim rozpoczniesz

Za pomocą produktu Rational Application Developer można utworzyć usługę Web Service komponentu EJB. Kreator usługi Web Service w produkcie Rational Application Developer ma możliwość tworzenia usługi Web Service przy użyciu rekomendacji kandydata W3C dla powiązania SOAP korzystającego

z usługi JMS. Wymagany jest produkt Rational Application Developer 7.54 . Ćwiczenie użyte w produkcie Rational Application Developer zawarte w produkcie Rational Software Architect dla oprogramowania WebSphere v7.5.5.1,

Komponent EJB jest wdrażany na serwerze WebSphere Application Server z poziomu produktu Rational Application Developer jako część tego zadania. Należy wykonać “Konfigurowanie serwera WebSphere Application Server do korzystania z protokołu W3C SOAP over JMS” na stronie 1035

Aby utworzyć plik WSDL rzeczywiście użyty w zadaniu, należy najpierw wykonać zadanie “Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 988. Następnie można zaimportować plik WSDL z dynamicznego projektu WWW w obszarze roboczym Galileo Eclipse lub z uruchomionej usługi Web Service HTTP wdrożonej w WASCE.

Serwer WebSphere Application Server może być nadal uruchomiony w wyniku działania produktu “Konfigurowanie zasobów serwera WebSphere Application Server” na stronie 1036. Jeśli tak nie jest, można go uruchomić z widoku Serwery w produkcie RAD.

O tym zadaniu

W ramach tego zadania usługa `StockQuoteAxis` jest wdrażana z poziomu działającego jako usługa Axis JAX-RPC uruchamiana przez produkt **`SimpleJavaListener`** przy użyciu transportu produktu WebSphere MQ dla protokołu SOAP w celu bycia usługą JAX-WS działającą na serwerze WebSphere Application Server przy użyciu protokołu SOAP W3C SOAP over JMS.

Istnieją dwie części do migracji usługi z serwera **`SimpleJavaListener`** do serwera WebSphere Application Server:

1. Wygeneruj interfejs usługi Web Service na podstawie pliku WSDL dla usługi przy użyciu kreatora usługi Web Service z góry w dół w produkcie Rational Application Developer.
2. Implementowanie usługi przez zaimportowanie przykładowego produktu WebSphere MQ SOAP `StockQuoteAxis.java`.

Alternatywnym podejściem byłoby wygenerowanie dolnej części usługi z poziomu `StockQuoteAxis.java`. Jednak aby mieć pewność, że interfejs do zmigrowanej usługi jest identyczny, podejście zstępujący jest lepsze, ponieważ korzysta z tego samego pliku WSDL.

Usługa Web Service jest rozwijana dla kontenera EJB, a nie kontenera WWW, ponieważ obsługa usługi JMS jest częścią kontenera EJB.

Procedura

1. Uruchom produkt Rational Application Developeri sprawdź, czy serwer aplikacji WebSphere jest uruchomiony.
   a) Uruchom produkt Rational Application Developer w nowym obszarze roboczym.
   b) Otwórz perspektywę Java EE .
   c) Otwórz kartę **Serwery** , a następnie sprawdź, czy serwer aplikacji WebSphere jest uruchomiony.
      - Jeśli w widoku nie ma serwera WebSphere Application Server v7.0 , kliknij prawym przyciskiem myszy w widoku > **Nowy** > **Serwer**. Aby utworzyć instancję serwera WebSphere Application Server v7.0 , należy wykonać następujące czynności w kreatorze.
      - Jeśli serwer jest obecny, ale nie został uruchomiony, kliknij przycisk ze strzałką, aby go uruchomić.
      - Aby sprawdzić właściwości i uzyskać szybki dostęp do dzienników serwera, kliknij prawym przyciskiem myszy opcję **WebSphere Application Server v7.0 at localhost** > **Właściwości** > **WebSphere Application Server**.
      - Aby administrować serwerem, należy użyć przeglądarki zewnętrznej i otworzyć adres URL, `http://localhost:9061/ibm/console/unsecureLogon.jsplub` kliknąć prawym przyciskiem myszy opcję **WebSphere Application Server v7.0 at localhost** > **Uruchom Konsolę administracyjną**.

- Ustawieniem domyślnym jest automatyczne publikowanie. Wiele osób preferuje ręczne wdrażanie aktualizacji na serwerze. Dwukrotnie kliknij pozycję **WebSphere Application Server v7.0 na hoście lokalnym**, a następnie rozwiń element graficzny **Publikowanie** w oknie**Przegląd** . Kliknij opcję **Nigdy nie publikuj automatycznie**.
- Inną wartością domyślną, która może być zmieniona, jest usunięcie zaznaczenia pola wyboru **Zakończ serwer w środowisku roboczym** w oknie **Przegląd** .

2. Tworzenie projektów JEE

   Należy utworzyć projekt aplikacji korporacyjnej (Enterprise Application Project-EAR) oraz projekt komponentu EJB (Enterprise Java Bean).

   a) **Plik** > **Nowy** > **Projekt aplikacji korporacyjnej**. Nadaj nazwę projektowi `W3CJMSEAR` > **Zakończ**.

      Wartości domyślne muszą identyfikować serwer WebSphere Application Server v7.0 jako docelowe środowisko wykonawcze, a plik EAR 5.0. Należy wybrać konfigurację domyślną.

   b) **Plik** > **Nowy** > **Projekt EJB**. Nadaj nazwę projektowi `W3CJMSEJB`. Wybierz opcję `W3CEARJMS` jako **nazwę projektu EAR** > **Dalej**.

      Domyślna wersja modułu EJB to 3.0 , a konfiguracja domyślna jest używana ponownie.

   c) Usuń zaznaczenie pola wyboru **Utwórz moduł JAR klienta EJB** > **Zakończ**.

3. Wygeneruj i wdróż usługę Web Service komponentu EJB przy użyciu pliku WSDL produktu `StockQuoteAxis` .

   a) **Uruchom** > **Uruchom eksplorator usług Web Services**.

   b) Wybierz stronę WSDL, korzystając z ikon w oknie**Eksplorator usług Web Services** > kliknij opcję **Główny plik WSDL** w oknie Navigator.

   c) W oknie **Działania** wpisz lub przejdź do adresu URL pliku WSDL dla produktu `StockQuoteAxis.wsdl`.

      Jeśli produkt WASCE został uruchomiony z produktem `StockQuoteAxis` wdrożonym jako usługa HTTP, adres URL jest następujący:

      ```
      http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl
      ```

      Jeśli w systemie plików znajduje się plik WSDL, adres URL może być następujący:

      ```
      File:\Dirpath\StockQuoteAxis\WebContent\wsdl\StockQuoteAxis.wsdl
      ```

   d) Kliknij wiersz zawierający zaimportowany adres URL w drzewie Navigator .

      Jest to wiersz bezpośrednio następujący po **WSDL Main**, jeśli jest to pierwszy plik WSDL, który został zaimportowany do programu Web Services Explorer.

   e) W oknie **Actions** (Działania) kliknij opcję **Launch Web Service Wizard** (Uruchom kreator usług Web Service) > **Web Service Skeleton** > **Go** (Wykonaj).

   f) W kreatorze usług Web Service wybierz opcję **Zstępujący komponent EJB Web Service** .

      Wybierz lub zweryfikuj konfigurację, korzystając z informacji z Tabela 141 na stronie 998 Zaznacz opcję **Nadpisz pliki bez ostrzeżenia** > **Dalej**.

Tabela 141. Konfiguracja usługi Web Service najwyższego poziomu komponentów EJB

Pole	**Wartość**
Serwer	`WebSphere Application Server v7.0`
Czas wykonywania usługi Web Service	`IBM WebSphere JAX-WS`
projekt usługi	`W3CJMSEJB`
Projekt EAR usługi	`W3CJMSEAR`
Konfiguracja:	`No client generation`

g) Na stronie podzatytułowanej **Określ opcje tworzenia górnej części usługi Web Service komponentu EJB WebSphere JAX-WS**zaznacz pole wyboru **Przełącz na powiązanie JMS**. Zaznacz także opcję **Włącz styl opakowania**, **Kopiuj plik WSDL do projektu** i **Generuj deskryptor wdrażania usługi Web Service** > **Dalej**.

h) Na stronie zatytułowanej **WebSphere Konfiguracja powiązania JMS JAX-WS**zaznacz pole **Użyj protokołu współdziałania SOAP/JMS** i podaj wartości z Tabela 142 na stronie 999, pozostawiając inne pola puste > **Następny**.

Tabela 142. Konfiguracja powiązania JMS produktu WebSphere JAX-WS

Pole	**Wartość**
Miejsce docelowe JMS	`queue`
Nazwa JNDI miejsca docelowego:	`requestaxis`
Fabryka połączeń JMS	`qm1`
Odpowiedz na nazwę	`W3CJMSEAR`
Konfiguracja:	`replyaxis`

a) Na stronie zatytułowanej **Konfiguracja projektu routera JAX-WS produktu WebSphere**wpisz `qm1as` w polu **Nazwa JNDI serwera ActivationSpec** > **Następny**.

RAD zajmuje około 30 sekund na minutę, aby wygenerować i wdrożyć projekt.

b) Opcje można zignorować na stronie **Web Service Publication** (Publikacje usługi Web Service) > **Finish**(Zakończ).

4. Sprawdź wygenerowany plik WSDL.

Poproszono o wygenerowanie pliku WSDL specyficznego dla usługi i zapisanie go w projekcie.

a) W nawigatorze Eksplorator korporacyjny otwórz folder **W3CJMSEJB** > **ejbmodule** > **META-INF** > **wsdl**. Kliknij dwukrotnie opcję `StockQuoteAxis.wsdl` , aby otworzyć ją w edytorze WSDL.

Sprawdź powiązanie; zostanie wyświetlony adres URL usługi JMS:

```
jms:jndi:requestaxis?jndiConnectionFactoryName=qm1&targetService=StockQuoteAxis
```

5. Krok opcjonalny: powiązanie komponentu EJB z SOAP za pośrednictwem protokołu HTTP przy użyciu interfejsu JAX-WS.

Udostępnienie dwóch powiązań do komponentu EJB umożliwia klientom wybór powiązań SOAP w celu wywołania usługi Web Service. Udostępnia on również klientom możliwość wysyłania zapytań do serwera WWW w celu uzyskania pliku WSDL przy użyciu protokołu HTTP.

Kroki, które należy wykonać, aby powiązać komponent EJB z SOAP za pośrednictwem protokołu HTTP, nie są uwzględniane w ramach zadania.

6. Zaimplementuj i ponownie wdróż produkt `StockQuoteAxis` , korzystając z przykładu `StockQuoteAxis.java`

a) W nawigatorze eksploratora korporacyjnego otwórz folder **W3CJMSEJB** > **Usługi** dwukrotnie kliknij opcję `StockQuoteAxisService` , aby otworzyć klasę implementacji w edytorze Java.

b) Otwórz przykładowy program `StockQuoteAxis.java` w folderze *`WebSphere MQ Installation directory`*`\tools\soap\samples\java\server` > Wybierz wszystkie metody, ale nie nazwę klasy > **Kopiuj**.

c) W programie `StockQuoteAxisSoapBindingImpl.java` należy wybrać wszystkie metody, ale nie nazwę klasy, a następnie wkleić je w metodach z programu `StockQuoteAxis.java`.

d) Dodaj instrukcję print do wyjścia do konsoli serwera WebSphere Application Server, gdy wywoływana jest usługa.
Zmień metodę getQuote(String symbol):

```
public float getQuote(String symbol) {
    System.out.println("StockQuoteAxisSoapBindingImpl called with symbol: "
                       + symbol);
    return ((float) 55.25);
```

e) Popraw importy: **Źródło** > **Organizuj instrukcje importu** > **Zapisz**.

f) Popraw trzy błędy z powodu implementacji, która nie jest zgodna z interfejsem.

Błędy wynikają z trzech metod w produkcie `StockQuoteAxis.java` zgłaszających wyjątki, a plik WSDL dla usługi nie zawierający komunikatów o błędach. Problem jest diagnozowany jako niezgodność między sygnaturami metod i adnotacjami usługi Web Service metody.

Utwórz adnotację metod z adnotacją @WebFault i ponownie wygeneruj plik WSDL lub zachowaj niezmieniony interfejs, a następnie usuń wyjątki.

Aby interfejs był taki sam, należy usunąć trzy `throws exception` z sygnatur metod: > **Zapisz**.

Co dalej

“Wdrażanie na kliencie Axis2 przy użyciu protokołu W3C SOAP over JMS” na stronie 1046

Zadania pokrewne

Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse

Zaprojektuj usługę Web Service Axis 1.4 , która będzie uruchamiana przy użyciu produktu WebSphere MQ jako dostawcy usług. Użyj normalnego środowiska programistycznego usług Web Service w celu utworzenia usługi dla wdrożenia w osi 1.4.

Projektowanie usługi .NET 1 lub 2 dla transportu WebSphere MQ dla protokołu SOAP za pomocą programu Microsoft Visual Studio 2008

Utwórz usługę Web Service SampleStockQuote dla platformy .NET 1 lub .NET 2 przy użyciu programu Microsoft Visual Studio 2008.

Projektowanie klientów usług Web Service produktu WebSphere MQ dla produktu WebSphere MQ dla protokołu SOAP

Aby utworzyć klienty usługi Web Service w celu użycia z transportem WebSphere MQ dla protokołu SOAP, należy użyć normalnego środowiska programistycznego.

Zanim rozpoczniesz

Utwórz usługę. W produkcie “Projektowanie usług Web Service dla transportu WebSphere MQ dla protokołu SOAP” na stronie 987można użyć jednego z przykładów.

Należy wybrać sposób tworzenia, wdrażania i używania klientów, a także miejsca, w którym można uzyskać dostęp do pliku WSDL na potrzeby generowania klienta.

Zdecyduj o podejściu do projektowania klientów i usług dla transportu WebSphere MQ dla protokołu SOAP.

Istnieją dwa podejścia.

1. Użyj standardowych narzędzi programistycznych, opracuj usługę HTTP i klienta, a następnie użyj adresu URL dla transportu WebSphere MQ dla protokołu SOAP.
2. Użyj narzędzi i przykładów dostarczanych razem z transportem WebSphere MQ dla protokołu SOAP.

W przypadku korzystania z trasy HTTP można uruchomić usługę na serwerze HTTP, a następnie uruchomić ją przy użyciu transportu WebSphere MQ dla protokołu SOAP. Aby uruchomić ten produkt przy użyciu transportu WebSphere MQ dla protokołu SOAP, należy skonfigurować odpowiedni program nasłuchujący WebSphere MQ dla protokołu SOAP i skonfigurować ścieżki i deskryptory wdrażania w celu uruchomienia usługi. Narzędzia udostępniane przez transport produktu WebSphere MQ dla protokołu SOAP są przeznaczone dla użytkownika. Alternatywnie można skonfigurować środowisko w taki sposób, aby uruchamiał obiekty nasłuchiwania.

Narzędzia dostarczane z produktem WebSphere MQ transport for SOAP są przydatne podczas rozpoczynania pracy i uczenia się, jak wdrażać transport. W przypadku pracy produkcyjnej istnieją korzyści z używania standardowych narzędzi i wdrażania tej samej usługi dostępnej dla różnych transportów SOAP.

Zdecyduj o typie klienta, który ma zostać opracowany

Należy zdecydować, jaki typ klienta usługi Web Service ma zostać opracowany. Wybór zależy od tego, czy znasz interfejs usługi, jak i adres usługi.

Jeśli interfejs jest znany, należy użyć narzędzi Axis lub .NET w celu wygenerowania klas klienta proxy z interfejsu usługi. Klasy klienta proxy ułatwiają napisanie klienta w celu wywołania usługi. Jeśli miejsce usługi jest znane podczas tworzenia klienta, należy użyć statycznego interfejsu proxy. Jeśli miejsce zmiany usługi zostanie zmienione, na przykład wtedy, gdy usługa zostanie ponownie wdrożona na serwerze produkcyjnym, należy użyć dynamicznego interfejsu proxy.

Jeśli interfejs usługi nie jest znany w czasie tworzenia klienta, na osi Axis można utworzyć klienta DII (Dynamic Invocation Interface) dla osi 1.4. Klient DII używa ogólnego interfejsu do wywołania dowolnej usługi. Aby poprawnie przekazać parametry do konkretnej usługi, należy programowo zbudować konkretny interfejs usługi. Buduj interfejs programowo w kliencie lub ładuj plik WSDL dla usługi do klienta. W systemie Axis2można utworzyć klienta dyspozytorskiego. Klient rozsyłania używa modelu dokumentu do opisania żądania klienta, podczas gdy klient DII korzysta z modelu wywołania. Zarówno praca nad dynamicznym budowaniem żądania.

Uzyskaj plik WSDL dla usługi

Z wyjątkiem przypadku programowego budowania interfejsu usługi, należy najpierw uzyskać plik WSDL usługi w celu utworzenia klienta usługi Web Service. Plik WSDL usługi jest dostępny z trzech różnych źródeł:

1. Bezpośrednio z implementacji usługi Web Service przy użyciu narzędzia, takiego jak **java2wsdl** (Axis) lub **disco** (.NET).
2. Wysyłając zapytanie do usługi Web Service przy użyciu adresu URL: *Web service http url*?wsdl.
3. Z pliku, albo w systemie plików, albo z rejestru, takiego jak UDDI lub WebSphere Service Registry and Repository.

Uwaga: Jeśli usługa nie jest dostępna za pomocą protokołu HTTP, to zapytanie WSDL nie będzie działać. Sama usługa może być dostępna tylko przy użyciu transportu WebSphere MQ dla protokołu SOAP.

Plik WSDL wygenerowany przez produkt **amqwdeployWMQService** nie jest taki sam, jak plik WSDL wygenerowany przy użyciu produktu **java2wsdl** lub **disco**. Wygenerowany plik WSDL jest również inny niż dowolny plik WSDL, który mógł zostać uruchomiony w celu utworzenia usługi "Góra w dół". W przypadku osi deskryptor wdrażania server-config.wsdd odwzorowuje komunikat SOAP wygenerowany przez klienta na operację i usługę. Produkt **amqwdeployWMQService** generuje inny deskryptor wdrażania ze środowiska Eclipse.

Plik WSDL używany do budowania klientów zależy od tego, w jaki sposób usługa jest wdrażana:

Wdrożone za pomocą amqwdeployWMQService
Użyj pliku WSDL wygenerowanego przez produkt **amqwdeployWMQService** . Podaj opcję -w i wybierz plik WSDL rpcLiteral . W celu zapewnienia zgodności można wybrać plik WSDL rpcEncoded . Plik WSDL rpcEncoded działa tylko w przypadku klientów .NET i Axis 1.4 .

Ręczne wdrażanie przy użyciu produktu SimpleJavaListener
Użyj jednego z następujących plików WSDL:

1. Plik WSDL używany do definiowania usługi lub przechowywania w repozytorium.
2. Plik WSDL wygenerowany z usługi przez produkt **java2wsdl** .
3. Plik WSDL, którego dotyczy zapytanie, przy użyciu adresu URL *Web service http url* ?wsdl, jeśli jest dostępny z serwera HTTP. Aby zaimportować definicję usługi bezpośrednio do środowiska Eclipse, można uruchomić narzędzie, takie jak eksplorator usług Web Services.

Może być konieczna zmiana identyfikatora URI usługi. Zmień adres z adresu usługi HTTP na identyfikator URI transportu produktu WebSphere MQ dla protokołu SOAP.

Ręczne wdrażanie przy użyciu produktu `amqSOAPNETListener`.
Użyj jednego z następujących plików WSDL:

1. Plik WSDL używany do definiowania usługi lub przechowywania w repozytorium.
2. Plik WSDL uzyskany z klasy usługi .NET (.asmx). za pomocą **`disco`**.
3. Za pomocą adresu URL *`Web service http url`* `?wsdl`zapytanie WSDL, jeśli jest dostępne, jest wykonywane. Aby zaimportować definicję usługi bezpośrednio do środowiska Eclipse, można uruchomić narzędzie, takie jak eksplorator usług Web Services.
4. Plik WSDL uzyskany przez uruchomienie komendy **`amqswsdl`** dla klasy usługi .NET (.asmx).

Może być konieczna zmiana identyfikatora URI usługi. Zmień adres z adresu usługi HTTP na identyfikator URI transportu produktu WebSphere MQ dla protokołu SOAP.

Wdrożono w systemie Windows Communication Foundation
Uzyskaj plik WSDL usługi przy użyciu adresu URL *`Web service http url`*`?wsdl`. Usługa musi być zdefiniowana za pomocą konfiguracji zachowania `serviceMetaData` w ramach definicji usługi.

Wdrożenie na innej platformie serwerowej.
Aby uzyskać poprawny plik WSDL usługi, postępuj zgodnie z wskazówkami udostępnionym przez platformę.

O tym zadaniu

Tworzenie klientów przy użyciu standardowych narzędzi programistycznych. Poniższe zadania ilustrują sposób budowania klientów dla platformy .NET 1 i 2, Axis 1.4 (JAX-RPC) i Axis2 (JAX-WS). W przypadku programu Windows Communication Foundation należy zapoznać się z odsyłaczami do zadań pokrewnych.

Projektowanie klienta JAX-RPC dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse

Należy opracować klienta usługi Web Service firmy Axis 1.4 , który będzie uruchamiany przy użyciu transportu produktu WebSphere MQ dla protokołu SOAP.

Zanim rozpoczniesz

Musi być dostępna usługa. Jeśli wykonujesz zadanie jako ćwiczenie praktyczne, skorzystaj z obszaru roboczego i usługi utworzonej w zadaniu “Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 988. Serwer aplikacji może działać w środowisku Eclipse, który obsługuje usługi Web Services środowiska Axis 1.4 . W ramach tego zadania korzystamy z dostępnego w nim dostępnego oprogramowania WebSphere Application Server Community Edition w wersji 2.1.4. Jest on skonfigurowany jako część zadania “Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 988. Może być również używany serwer Tomcat 6, który jest mniejszym serwerem aplikacji open source.

O tym zadaniu

Zadanie przedstawia rozwój trzech typów klienta dla przykładowej usługi Axis dla produktu StockQuoteprzy użyciu środowiska Eclipse działającego w systemie Windows. Klientami są klient statyczny i dynamiczny, który został opracowany przy użyciu klienta proxy klienta oraz klienta DII.

Ilustrowane są dwa alternatywne podejścia do generowania proxy klienta z pliku WSDL:

1. Generowanie proxy klienta przy użyciu produktu **`amqwdeployWMQService`** .
2. Importowanie pliku WSDL do środowiska Eclipsei użycie kreatora usługi Web Service w celu wygenerowania proxy klienta.

Procedura

1. Uruchom środowisko Eclipse IDE dla programistów Java EE .
2. Utwórz projekt Java o nazwie `StockQuoteAxisClient`:
   a) Przejdź do perspektywy Java > **Plik** > **Nowy** > **Projekt Java**. W polu **Project name** w polu **Utwórz stronę projektu Java** wpisz `StockQuoteAxisEclipseClient`. Upewnij się, że środowisko wykonawcze ma wartość **J2SE1-1.4** lub **J2SE-1.5** > **Dalej** .
   b) On the **Ustawienia Java** page, select the **Biblioteki** tab > **Dodaj zewnętrzne pliki JAR ...**
   c) Przejdź do programu *MQ_INSTALLATION_PATH*`/java/lib` i wybierz wszystkie pliki `.jar` > **Otwórz**.
   MQ_INSTALLATION_PATH to katalog, w którym zainstalowano produkt WebSphere MQ .
   d) Przejdź do programu *MQ_INSTALLATION_PATH*`/java/lib/soap` i wybierz wszystkie pliki `.jar` > **Otwórz**. Produkt `axis.jar` musi być zainstalowany z nośnika instalacyjnego produktu WebSphere MQ do tego katalogu.
   MQ_INSTALLATION_PATH to katalog, w którym zainstalowany jest produkt WebSphere MQ .
   e) Karta **Biblioteka** odwołuje się teraz do wszystkich plików `.jar` potrzebnych do zbudowania klienta > **Zakończ**.
3. Aby utworzyć proxy w środowisku Eclipse dla przykładowej usługi Web Service StockQuoteAxis, należy wykonać jedną z tych dwóch metod tworzenia proxy:
   - Wygeneruj proxy klienta przy użyciu produktu **amqwdeployWMQService** .
     a. Utwórz menedżera kolejek: W przypadku zadania utwórz `QM1` jako domyślny menedżer kolejek.
     b. Utwórz katalog roboczy `samples`. Skopiuj przykładowy program `StockQuoteAxis.java` do programu `samples/soap/server`.
     c. Zmodyfikuj `amqwsetcp.cmd` w programie *MQ_INSTALLATION_PATH*`/bin` , aby uwzględnić bieżący katalog w ścieżce klasy. *MQ_INSTALLATION_PATH* Jest to katalog, w którym zainstalowano produkt WebSphere MQ .
     d. Otwórz okno komend w programie `samples` i uruchom zmodyfikowaną komendę **amqwsetcp** .
     e. Utwórz plik WSDL dla usługi Axis StockQuote, uruchamiając komendę,

```
amqwdeployWMQService -f soap/server/StockQuoteAxis.java -c genAxisWsdl
-u "jms:/queue?destination=REQUESTAXIS
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QM1)binding(auto))"
```

     Zapamiętaj: Jeśli używane są komendy Java, należy używać "`/`", a nie "`.`" lub "`\`" .

     Wskazówka: Zamiast importować wygenerowane serwery proxy do środowiska Eclipse, można zaimportować wygenerowany plik WSDL z produktu `.samples/generated`. Wynikowe serwery proxy różnią się na dwa sposoby:

     i) Nazwy pakietów są różne-co można refaktoryzować.
     ii) Wygenerowane serwery proxy produktu Eclipse zawierają dodatkową klasę pomocniczą, `StockQuoteAxisProxy.java`

     f. Utwórz serwery proxy klienta dla usługi Axis StockQuote, uruchamiając komendę:

```
amqwdeployWMQService -f soap/server/StockQuoteAxis.java -c genProxiestoAxis
-u "jms:/queue?destination=REQUESTAXIS
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QM1)binding(auto))"
```

     g. Zaimportuj proxy klienta do programu StockQuoteAxisClient:

     i) Kliknij prawym przyciskiem myszy opcję **StockQuoteAxisClient\src** > Wybierz opcję **System plików** > **Dalej** > **Przeglądaj ...** > znajdź folder `.\samples\generated\client\remote\soap\server` > **OK**.

ii) Zaznacz pole wyboru **serwer** na stronie**Import** > **Zakończ**.

h. Refaktoryzuj nazwę pakietu na `soap.server`.

i) Kliknij prawym przyciskiem myszy pakiet zawierający proxy klienta > **Refaktoryzuj** > **Zmień nazwę**. Wpisz **New name**: `soap.server` > pozostaw wybrane wartości domyślne dla innych opcji > **OK** . Wszystkie błędy są naprawiane.

- Wygeneruj proxy klienta przy użyciu środowiska Eclipse.

  Istnieje możliwość wyboru sposobu uzyskiwania dokumentu WSDL dla usługi. W tym przykładzie usługa została wdrożona na serwerze WebSphere Application Server Community Edition i uzyskana jest plik WSDL z serwera WWW. Wdrażanie zostało opisane w zadaniu “Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 988.

  a. W środowisku Eclipseprzejdź do perspektywy WWW i upewnij się, że serwer WebSphere Application Server Community Edition v2.1 Server jest uruchomiony, a produkt StockQuoteAxis jest wdrożony i zsynchronizowany.

  b. Zaimportuj plik WSDL do Eksploratora usług Web Services:

  i) Kliknij ikonę **Eksplorator usług Web Services** na pasku działań lub kliknij opcje **Uruchom** > **Uruchom eksplorator usług Web Services**.

  ii) Aby przełączyć się na stronę WSDL, należy kliknąć ikonę strony WSDL w Eksploratorze usług Web Services.

  iii) Kliknij opcję **WSDL Main** w oknie Navigator w Eksploratorze usług Web Services.

  iv) Wpisz adres URL usługi Web Service, a następnie `?WSDL` . Adres URL dla osi StockQuote, wdrożonego w zadaniu “Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 988, jest następujący:

  ```
  http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl
  ```

  c. Wygeneruj proxy klienta:

  i) W nawigatorze Eksploratora usług Web Services kliknij opcję **http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl** .

  ii) W oknie **Actions** (Działania) kliknij opcję **Launch Web Service Wizard** (Uruchom kreator usług Web Service) > leave **Web Service Client** selected > **Go**(Klient usługi Web Service)

  iii) Na pierwszej stronie kreatora kliknij odsyłacz do projektu **Klient** w konfiguracji > Wybierz projekt klienta **StockQuoteAxisClient** > **OK**.

  Wskazówka: Okno kreatora może zostać utracone. Konieczne jest ręczne wprowadzenie go do aktywnego obiektu.

  iv) Środowisko wykonawcze usługi Web Service musi być produktem Apache Axis w celu wygenerowania klienta JAX-RPC.

  v) Kliknij opcję **Zakończ**.

  vi) Zmień statyczny adres URL usługi tak, aby wskazywałon na transport WebSphere MQ dla adresu SOAP dla usługi StockQuoteAxis. Ten krok można pominąć, dopóki klient nie zostanie przetestowany z serwerem HTTP.

  a) Otwórz program `StockQuoteAxisServiceLocator.java` i znajdź deklarację dla produktu `StockQuoteAxis_address`.

  b) Zmień adres URL na

  ```
  "jms:/queue?destination=REQUESTAXIS
  &amp;initialContextFactory=com.ibm.mq.jms.Nojndi
  &amp;connectionFactory=(connectQueueManager(QM1)binding(auto))"
  ```

  Wskazówka: Środowisko Eclipse automatycznie transformuje & na `&`, a odwrotnie, kopiując i wklejać łańcuchy do kodu `.java` .

d. Utwórz trzy klasy klienta Java, z których każda ma główną metodę:

   i) Utwórz pakiet. Kliknij prawym przyciskiem myszy opcję **StockQuoteAxisClient/src** > **Nowy pakiet**. Nadaj mu nazwę `soap.client` > **Zakończ**.

   ii) Wybierz opcję **soap.client** > **Nowy** > **Klasa**. Nazwij klasę `SQAStaticClient` > Sprawdź **public static void main (string [] args)** > Zakończ

   iii) Powtórz procedurę, aby utworzyć `SQADynamicClient.java` i `SQADIIClient.java`

e. Napisz kod klienta.

   Produkt Rysunek 177 na stronie 1008 za pomocą programu Rysunek 181 na stronie 1010 udostępnia przykłady trzech stylów kodu klienta. W przykładach używany jest adres URL protokołu HTTP w celu przetestowania klienta przy użyciu usługi Axis StockQuotewdrożonej na serwerze HTTP. Aby uruchomić klienty dla usługi Axis (StockQuote) wdrożonej przy użyciu transportu WebSphere MQ dla protokołu SOAP, należy zmienić adres URL na:

```
"jms:/queue?destination=REQUESTAXIS
connectionFactory=(connectQueueManager(QM1)binding(auto))
initialContextFactory=com.ibm.mq.jms.Nojndi
targetService=soap.server.StockQuoteAxis.java
replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE"
```

   – Produkty Rysunek 177 na stronie 1008 i Rysunek 179 na stronie 1009 korzystają z serwera proxy wygenerowanego przez produkt Eclipse, który ma dodatkową klasę pomocniczą StockQuoteAxisproxy, która ułatwia kodowanie.

   – Produkty Rysunek 178 na stronie 1009 i Rysunek 180 na stronie 1009 korzystają z serwera proxy wygenerowanego przez produkt **amqwdeployWMQService** .

   – Produkt Rysunek 181 na stronie 1010 nie używa klas proxy.

   Każdy klient wywołał połączenie `com.ibm.mq.soap.Register.extension()` w celu połączenia się z transportem WebSphere MQ dla protokołu SOAP. Rozszerzenie jest zarejestrowane w deskryptorze wdrażania klienta. Wdrożenie klienta w firmie Axis 1.4 jest opisane w sekcji “Wdrażanie klienta usługi Web Service do środowiska Axis 1.4 w celu korzystania z transportu IBM WebSphere MQ dla protokołu SOAP” na stronie 1041.

f. Uruchom klienty, wysyłając żądanie SOAP do osi StockQuoteudostępnianej przez serwer WebSphere Application Server Community Edition skonfigurowany w obszarze roboczym.

   i) Sprawdź, czy serwer jest uruchomiony, produkt StockQuoteAxis został wdrożony i zsynchronizowany.

   ii) Wybierz lub otwórz klienta, który ma zostać przetestowany > Kliknij przycisk **Uruchom** na pasku działań. Można również kliknąć ikonę Uruchom zielony lub osiem kliknięcie klienta w nawigatorze > **Uruchom jako** > **Uruchom konfiguracje ...**. Skonfiguruj parametry wymagane do uruchomienia klienta.

g. Uruchom klienta, korzystając z transportu WebSphere MQ dla protokołu SOAP.

   Procedura korzysta z produktu **amqwdeployWMQService** w celu wdrożenia usługi i działa tylko z klientem, który korzysta z pliku WSDL lub proxy zbudowanych przez produkt **amqwdeployWMQService** . Aby uruchomić klienta przy użyciu oryginalnego pliku WSDL lub proxy zbudowanego przez środowisko Eclipse, należy wdrożyć usługę z jej deskryptorem wdrażania zbudowanym przez środowisko Eclipse. Ręcznie uruchom produkt **SimpleJavaListener** , korzystając z nazwy powiązania portu usługi jako `targetService` .

   i) Postępuj zgodnie z instrukcjami w sekcji “Wdrażanie usługi w firmie Axis 1.4 w celu użycia w transporcie WebSphere dla protokołu SOAP przy użyciu produktu amqwdeployWMQService” na stronie 1029 , aby wdrożyć usługę w programie nasłuchującym SOAP WebSphere MQ Simple Java SOAP. Wdrożenie usługi działa tylko dla klienta, korzystając z serwerów proxy WSDL lub klienta zbudowanych przez produkt **amqwdeployWMQService**.

   ii) W oknie komend uruchom program **amqwclientconfig** , aby utworzyć plik deskryptora wdrażania klienta `client-deploy.wsdd` .

iii) Zaimportuj plik `client-deploy.wsdd` do katalogu głównego projektu Java, który ma być testowany przy użyciu transportu produktu WebSphere MQ dla protokołu SOAP.

a) Kliknij prawym przyciskiem myszy projekt Java **StockQuoteAxisEclipseClient** > **Import** > **System plików** > **Dalej** > **Przeglądaj ...**

b) Przejdź do katalogu zawierającego opcje `client-deploy.wsdd` > **Otwórz** > Wybierz katalog na stronie kreatora **Importuj** > zaznacz pole wyboru `client-deploy.wsdd` w panelu po prawej stronie.

c) Sprawdź, czy w folderze **do folderu:** została wprowadzona wartość `StockQuoteAxisEclipseClient` > **Zakończ**.

iv) Upewnij się, że katalog roboczy na potrzeby uruchamiania aplikacji Java w tym projekcie to katalog `StockQuoteAxisEclipseClient` :

Kliknij prawym przyciskiem myszy projekt Java **StockQuoteAxisEclipseClient** > **Uruchom jako ....** > **Uruchom konfiguracje ...** > Wybierz kartę **(x) = argumenty** > Sprawdź, czy w katalogu roboczym zaznaczono przełącznik **Domyślny** , a ścieżka to `StockQuoteAxisEclipseClient` . Alternatywnie można dokonać jednej z następujących opcji, aby wybrać inne położenie lub plik zawierający konfigurację klienta:

- Zaznacz opcję **Inne:** > wpisz ścieżkę do katalogu wybranego przez użytkownika.
- W oknie **Argumenty maszyny VM** wpisz `-Daxis.ClientConfigFile=`*`full path to client deployment descriptor file`*

v) Upewnij się, że adres URL jest skonfigurowany w taki sposób, aby wskazywał na usługę wdrożoną przy użyciu transportu WebSphere MQ dla protokołu SOAP. Uruchom klienta zgodnie z opisem w kroku ii.

Wskazówka: Zwykle może wystąpić jeden z następujących błędów:

i) `Exception: No client transport named 'jms' found!.`

ii) Błąd połączenia JMS.

iii) `Exception: The AXIS engine could not find a target service to invoke! targetService is soap.server.StockQuoteAxis.java`

iv) `Exception: java.lang.InstantiationException: soap.server.StockQuoteAxis`

Wyjaśnienia:

i) `client-config.wsdd` nie został znaleziony lub zawiera wiersz `<transport name="jms" pivot="java:com.ibm.mq.soap.transport.jms.WMQSender"/>` w `client-config.wsdd`.

ii) Prawdopodobnie problem ze ścieżką budowania-nie zawierający plików .jar w produkcie *MQ_INSTALLATION_PATH*`/java/lib` . *MQ_INSTALLATION_PATH* to katalog, w którym zainstalowany jest produkt WebSphere MQ .

iii) Problem z wdrażaniem usługi przy użyciu produktu `server-config-wsdd` lub z parametrami przekazanych do produktu **`SimpleSoapListener`** .

iv) Niezgodność między deskryptorem wdrażania a implementacją usługi.

Jeśli masz problemy z uruchomieniem klienta w środowisku Eclipse, spróbuj użyć okna komend:

i) Przejdź do katalogu `StockQuoteAxisEclipseClient\bin` w drzewie katalogów obszaru roboczego.

ii) Uruchom **amqwsetcp** i **amqwclientconfig**

iii) Uruchom program `java soap/client/SQAStaticClient`.

Przykładowe klienty usługi Web Service JAX-RPC

Przykładowe klienty usługi Java Web Service dostarczane z produktem WebSphere MQ są instalowane w produkcie *MQ_INSTALLATION_PATH* \tools\soap\samples\java\clients. *MQ_INSTALLATION_PATH* Jest to katalog, w którym zainstalowano produkt WebSphere MQ .

SQAxis2Axis.java
SQAxis2Axis.java, Rysunek 174 na stronie 1007 , jest dynamicznym klientem proxy w celu wywołania usługi StockQuoteAxis . Adres URL usługi, który jest kompilowany do dynamicznego proxy, można przesłonić, udostępniając adres URL w wierszu komend.

SQAxis2DotNet.java
SQAxis2DotNet.java, Rysunek 175 na stronie 1007, jest dynamicznym klientem proxy w celu wywołania usługi StockQuoteDotNet . Adres URL usługi, który jest kompilowany do dynamicznego proxy, można przesłonić, udostępniając adres URL w wierszu komend.

WsdlClient.java
WsdlClient.java, Rysunek 176 na stronie 1008jest dynamicznym klientem wywołania w celu wywołania usługi StockQuoteDotNet lub StockQuoteAxis . Klient domyślnie wywołuje usługę StockQuoteAxis . Dodaj opcję wiersza komend -D wywołaj usługę StockQuoteDotNet i -w, aby udostępnić inny port w produkcie .\generated\StockQuoteDotNet_Wmq.wsdl .

```
package soap.clients;
import java.net.URL;
import soap.server.*;
public class SQAxis2Axis {
   public static void main(String[] args) {
      com.ibm.mq.soap.Register.extension();
      try {
         StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
         StockQuoteAxis service = null;
         if (args.length == 0)
            service = locator.getSoapServerStockQuoteAxis_Wmq();
         else
            service = locator.getSoapServerStockQuoteAxis_Wmq(
                new java.net.URL(args[0]));
         System.out.println("Response: " + service.getQuote("XXX"));
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING ProxyClient DEMO <<<\n");
                           e.printStackTrace();
         System.exit(2);
      }
   }
```

Rysunek 174. SQAxis2Axis.java

```
public class SQAxis2DotNet {
public static void main(String[] args) {
   com.ibm.mq.soap.Register.extension();
      try {
         StockQuoteDotNet locator = new StockQuoteDotNetLocator();
         StockQuoteDotNetSoap_PortType service = null;
         if (args.length == 0)
            service = locator.getStockQuoteDotNetSoap();
         else
            service = locator.getStockQuoteDotNetSoap(new java.net.URL(
                  args[0]));
         System.out.println("Response: " + service.getQuoteDOC("XXX"));
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING ProxyClient DEMO <<<\n");
                           e.printStackTrace();
         System.exit(2);
      }
   }
}
```

Rysunek 175. SQAxis2DotNet.java

```
package soap.clients;
import com.ibm.mq.soap.*;
import org.apache.axis.utils.Options;
import java.net.URL;
import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceFactory;
import javax.xml.namespace.QName;
public class WsdlClient {
public static void main(String[] args) {
      String wsdlService, wsdlPort, namespace, wsdlSource, wsdlTargetURI, s;
      try {
         Register.extension();
         Options opts = new Options(args);
         if (opts.isFlagSet('D') != 0) {
            wsdlService = "StockQuoteDotNet";
            wsdlPort = "StockQuoteDotNetSoap";
            namespace = "http://stock.samples";
            wsdlSource = "file:generated/StockQuoteDotNet_Wmq.wsdl";
         } else {
            wsdlService = "StockQuoteAxisService";
            wsdlPort = "soap.server.StockQuoteAxis_Wmq";
            namespace = "soap.server.StockQuoteAxis_Wmq";
            wsdlSource = "file:generated/soap.server.StockQuoteAxis_Wmq.wsdl";
         }
         if (null != (s = (opts.isValueSet('w'))))
            wsdlPort = s;
         System.out.println("start WsdlClient demo, wsdl port " + wsdlPort
               + " resolving uri to ...");
         QName servQN = new QName(namespace, wsdlService);
         QName portQN = new QName(namespace, wsdlPort);
         Service service = ServiceFactory.newInstance().createService(
               new URL(wsdlSource), servQN);
         Call call = (Call) service.createCall(portQN, "getQuote");
         wsdlTargetURI = call.getTargetEndpointAddress().toString();
         System.out.println("  '" + wsdlTargetURI + "'");
         Object ret = call.invoke(new Object[] { "XXX" });
         System.out.println("Response: " + ret);
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING WsdlClient DEMO <<<\n");
         e.printStackTrace();
         System.exit(2);
      }
   }
}
```

Rysunek 176. WsdlClient.java

Przykładowe klienty użyte w tym zadaniu:

```
package soap.client;
import soap.server.StockQuoteAxisProxy;
public class SQAStaticClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisProxy sqa = new StockQuoteAxisProxy();
      System.out.println("Static client synchronous result is:"
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Rysunek 177. Klient statyczny korzystający z wygenerowanego przez Eclipse proxy

```
package soap.client;
import soap.server.StockQuoteAxis;
import soap.server.StockQuoteAxisService;
import soap.server.StockQuoteAxisServiceLocator;
public class SQAStaticClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
      StockQuoteAxis sqa = locator.getSoapServerStockQuoteAxis_Wmq();
      System.out.println("Static client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Rysunek 178. Klient statyczny korzystający z wygenerowanego serwera proxy amqwdeployWMQService

```
package soap.client;
import soap.server.StockQuoteAxisProxy;
public class SQADynamicClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisProxy sqa = new StockQuoteAxisProxy(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      System.out.println("Dynamic client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Rysunek 179. Klient dynamiczny korzystający z wygenerowanego serwera proxy Eclipse

```
package soap.client;

import java.net.URL;
import soap.server.StockQuoteAxis;
import soap.server.StockQuoteAxisService;
import soap.server.StockQuoteAxisServiceLocator;
public class SQADynamicClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      URL sqaURL = new URL(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
      StockQuoteAxis sqa = locator.getSoapServerStockQuoteAxis_Wmq(sqaURL);
      System.out.println("Dynamic client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Rysunek 180. Klient dynamiczny korzystający z wygenerowanego przez usługę amqwdeployWMQService proxy

```
package soap.client;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceFactory;
public class SQADIIClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      URL wsdl = new URL(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl");
      Service SQAService = (ServiceFactory.newInstance()).createService(wsdl,
          new QName("http://server.soap", "StockQuoteAxisService"));
      Call SQACall = SQAService.createCall(new QName("http://server.soap",
          "StockQuoteAxis"), "getQuote");
      System.out.println("DII client synchronous result is "
          + SQACall.invoke(new Object[] { "ibm" }));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Rysunek 181. Klient DII (bez proxy)

Zadania pokrewne

Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse
Należy opracować klienta usługi Web Service Axis2 , który będzie uruchamiany przy użyciu transportu WebSphere MQ dla protokołu SOAP. Zostaną wyświetlone przykładowe klienty Axis2 dostarczane z produktem WebSphere MQ transport dla protokołu SOAP, a także komenda **wsimport** używana do generowania proxy.

Tworzenie klienta .NET 1 lub 2 dla transportu WebSphere dla protokołu SOAP za pomocą programu Microsoft Visual Studio 2008
Należy utworzyć klient usługi Web Service .NET 1 lub 2, który ma być uruchamiany przy użyciu transportu WebSphere MQ dla protokołu SOAP.

Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse

Należy opracować klienta usługi Web Service Axis2 , który będzie uruchamiany przy użyciu transportu WebSphere MQ dla protokołu SOAP. Zostaną wyświetlone przykładowe klienty Axis2 dostarczane z produktem WebSphere MQ transport dla protokołu SOAP, a także komenda **wsimport** używana do generowania proxy.

Zanim rozpoczniesz

Uzyskaj biblioteki Axis2 , a następnie skonfiguruj środowisko programistyczne i testowe, aby uruchomić klienta.

Uwaga: Nazewnictwo wersji i wydań używanych przez produkt Axis powoduje nieporozumienia. Zwykle nazwa Axis 1.4 odnosi się do implementacji JAX-RPC, a nazwa Axis2 do implementacji JAX-WS.

Axis 1.4 to poziom wersji. Jeśli wyrażenie Axis 1.4 zostanie wyszukane w Internecie, użytkownik zostanie skierowany na stronę http://ws.apache.org/axis/. Ta strona zawiera listę poprzednich wersji produktu Axis (1.2, 1.3) oraz ostatnią wersję Axis 1.4 z 22 kwietnia 2006 roku. Istnieją nowsze wersje produktu Axis 1.4, w których naprawiono błędy, ale wszystkie one są określane mianem Axis 1.4. Jest to jeden z tych wersji poprawek, które są dostarczane razem z produktem WebSphere MQ. W przypadku osi 1.4należy użyć wersji produktu `axis.jar` dostarczanej z produktem WebSphere MQ , a nie wersji produktu http://ws.apache.org/axis/, która może być dostępna.

W serwisie WWW Axis określenie Axis 1.1 odnosi się do wszystkich wersji, które są zwykle określane mianem Axis 1.4. Z kolei Axis 1.2 to określenie używane zwykle w przypadku produktu Axis2.

Wersja Axis 1.5 nie jest nowszą wersją produktu Axis 1.4, lecz wersją Axis2. W przypadku wyszukania wyrażenia Axis 1.5 użytkownik zostanie skierowany na stronę http://ws.apache.org/axis2/.https://

ws.apache.org/axis2/download.cgi zawiera listę wersji produktu Axis2, o etykiecie od 0.9 do 1.5.1 (łącznie z, co jest mylące, wersją 1.4). Wersja produktu Axis2 , która ma być używana z transportem WebSphere MQ dla protokołu SOAP, jest wersją 1.4.1. Produkt Axis2 1.4.1 należy pobrać ze strony http://ws.apache.org/axis2/download/1_4_1/download.cgi.

Istnieje możliwość wygenerowania proxy dla klientów usług Web Service dla transportu WebSphere MQ dla protokołu SOAP przy użyciu produktu **wsimport** lub narzędzia udostępnionego wraz z IDE. Produkt Eclipse IDE for Java EE Developer 3.5 SR1 używa produktu **wsdl2java**.Produkt **wsimport** jest dostarczany z językiem Java 6. Środowiska Java 5 można używać do uruchamiania proxy klienta wygenerowanych za pomocą produktu **wsimport** lub **wsdl2java**.

Przykładowe klienty usługi Web Service Axis2 dostarczane z produktem WebSphere MQ transport for SOAP zostały opracowane przy użyciu produktu **wsimport**. Patrz sekcja “Przykładowe klienty Axis2” na stronie 1016.

W poniższym zadaniu przedstawiono sposób generowania i używania serwerów proxy utworzonych przez kreator usług Web Service w pakiecie razem z produktem Eclipse IDE for Java EE Developers. Przykładowe klienty pokazują, w jaki sposób korzystać z serwerów proxy utworzonych przez produkt **wsimport**.

Aby korzystać z kreatora usług Web Services, należy dodać do środowiska roboczego serwer aplikacji obsługujący środowisko Axis2 . W poniższych krokach przedstawiono sposób konfigurowania produktu WASCE do obsługi środowiska Axis2 przy użyciu środowiska roboczego.

1. Skonfiguruj serwer aplikacji używany w środowisku Eclipse IDE for Java EE Developers do obsługi środowiska Axis2. W tym przykładzie należy skonfigurować serwer aplikacji WASCE 2.1.4, który jest częścią obszaru roboczego utworzonego w produkcie “Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 988.
   a. Otwórz preferencje obszaru roboczego, aby skonfigurować serwer: Otwórz okno **Okna** > **Preferencje**.
   b. Sprawdź, czy zainstalowane środowisko JRE to `Java50`: kliknij opcję **Zainstalowane środowiska JRE**.
   c. Dodaj serwer WASCE jako serwer: kliknij opcję **Serwer** > **Środowiska wykonawcze** > **Dodaj ...** > **IBM** > **WASCE v2.1** > Dalej. Środowisko JRE musi mieć wartość Java50 > Przejdź do katalogu instalacyjnego WASCE > **OK** > **Zakończ**. Konieczne jest zainstalowanie wtyczki WASCE dla środowiska IDE Eclipse Java EE for Web Developers.
   d. Dodaj element Axis2: kliknij opcję **Usługi Web Service** > **PreferencjeAxis2**. Na karcie **Środowisko wykonawcze Axis2** > **Przeglądaj ...** Otwórz katalog zawierający wiele plików jar produktu `Axis2` > **Zastosuj**.
   e. Powiąż produkt WASCE z Axis2: Kliknij opcję **Usługi Web Services** > **Serwer i środowisko wykonawcze**. W obszarze **`Server`** wybierz opcję **IBM WASCE v2.1 Server**, a w obszarze **Web `service runtime`**wybierz opcję **Apache Axis2** > **Zastosuj** > **OK** .
   f. Uruchom serwer: otwórz perspektywę WWW i otwórz widok Serwery. Kliknij prawym przyciskiem myszy w widoku Serwery > **Nowy** > **Serwer**. Opcja **IBM WASCE v2.1 Server** jest wybrana i skonfigurowana > **Finish**(Zakończ). Uruchom serwer.
2. Sprawdź, czy usługa Axis StockQuotezostała wdrożona w serwerze WASCE w celu uruchomienia kreatora usługi Web Service.
3. Aby przetestować usługę przy użyciu transportu produktu WebSphere MQ dla usługi SOAP, należy wdrożyć usługę w transporcie WebSphere MQ dla programu nasłuchującego SOAP dla osi 1.4. Patrz “Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 988.

O tym zadaniu

Produkt Eclipse IDE for Java EE Developers używa języka Java50 oraz kreatora usług Web Services do generowania klas proxy dla usługi. Klasy proxy są różne dla klas utworzonych za pomocą narzędzia **wsimport** udostępnionego w środowisku Java 6. Alternatywnym podejściem jest wygenerowanie klas

proxy przy użyciu produktu **wsimport** i zaimportowanie pakietów utworzonych przez niego do środowiska IDE Eclipse Java EE for Web Developers.

Kreator usług Web Service w środowisku Eclipse IDE for Java EE Developers buduje klienta usługi Web Service w projekcie WWW. Klient może być uruchamiany jako prosta aplikacja Java. Nie wymaga ona serwera aplikacji. Można także przesłać kod do projektu Java i skonfigurować ścieżkę budowania w taki sposób, aby zawierała pliki JAR Axis2 .

Procedura

1. Utwórz projekt WWW w nowym projekcie przedsiębiorstwa:
   a) Jeśli w eksploratorze projektów nie wybrano żadnych elementów, > kliknij prawym przyciskiem myszy biały znak > **Nowy** > **Projekt aplikacji korporacyjnej** > Nazwa, który `StockQuoteAxis2EAR` > **Zakończ**. Odpowiedz `No` na to okno, podając opcję otwierania perspektywy Java EE .

      Wartości domyślne są ustawione tak, aby używały WASCE.

   b) Kliknij prawym przyciskiem myszy opcję `StockQuoteAxis2EAR` > **Nowy** > **Dynamiczny projekt WWW**. Nadaj nazwę projektowi `StockQuoteAxis2WebClient` > Sprawdź, czy w polu przypisania do pliku EAR projekt ma zostać dodany do pliku **StockQuoteAxis2EAR**. WASCE 2.1 jest wybrane jako docelowe środowisko wykonawcze.

   c) W sekcji Konfiguracja na stronie **Nowy dynamiczny projekt WWW** > **Modyfikuj ...** > Sprawdź aspekt projektu usług Web Service Axis2 . **Dynamiczne moduły WWW 2.5**, **Java 6.0**i **Wdrożenie WASCE 1.2** są już zaznaczone. > **OK** > **Zakończ**. Odpowiedz `No` na to okno, podając opcję otwierania perspektywy Java EE .
2. Zaimportuj plik WSDL dla usługi do obszaru roboczego i wygeneruj proxy klienta:

   W tym przykładzie dokument WSDL zawiera powiązanie usługi HTTP i staje się celem dla statycznego proxy klienta WWW. Przed wygenerowaniem proxy klienta można zmodyfikować adres URL w powiązaniu usługi Web Service w taki sposób, aby wskazywało na transport produktu WebSphere MQ dla adresu URL SOAP. Statyczny serwer proxy klienta WWW to usługa, która jest wdrażana w transporcie WebSphere MQ dla protokołu SOAP.

   a) Uruchom eksplorator usług Web Services: użyj ikony na pasku działań lub wybierz opcję **Uruchom** > **Uruchom eksplorator usług Web Services**.

   b) Wybierz eksplorator WSDL, klikając ikonę WSDL w oknie **Eksplorator usług WWW** > Kliknij opcję **WSDL-główne** w oknie Navigator > Wpisz adres URL pliku WSDL środowiska Axis StockQuote> **Przejdź do**.
      W tym przykładzie można uzyskać plik WSDL bezpośrednio z usługi HTTP: `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl`

   c) W oknie Navigatorkliknij wiersz z adresem URL usługi Web Service. W oknie **Działania** kliknij opcję **Importuj plik WSDL do środowiska roboczego** > Wybierz opcję **StockQuoteAxis2WebClient** jako **Projekt środowiska roboczego** > Wpisz **nazwę pliku WSDL**, `StockQuoteAxisHTTP.wsdl` > **Przejdź**.

   d) Kliknij prawym przyciskiem myszy opcję **StockQuoteAxisHTTP.wsdl** > **Web Services** > **Generuj klienta**. Sprawdź, czy informacje o konfiguracji strony usług Web Service kreatora są następujące: Serwer: IBM WASCE v2.1 Server, środowisko wykonawcze usługi Web Service: Apache Axis2, projekt klienta: StockQuoteAxis2WebClient, projekt EAR klienta: StockQuoteAxisEAR. Aby poprawić konfigurację, kliknij niepoprawne wiersze.

   e) Kliknij przycisk **Dalej** > zweryfikuj ustawienia generowania kodu > **Zakończ**.

      Należy zauważyć, że tworzony jest nowy pakiet, `soap.server`, który zawiera wymagane proxy.
3. Skonfiguruj projekt, aby uruchomić transport produktu WebSphere MQ dla protokołu SOAP jako transport JMS.

   Transport produktu WebSphere MQ dla protokołu SOAP udostępnia obiekt `transportSender`, ale nie ma nazwy `transportReceiver`. Innymi słowy, transport produktu WebSphere MQ dla protokołu SOAP obsługuje klienty Axis2 . Obecnie usługi Axis2 nie są obsługiwane.

a) W projekcie **StockQuoteAxis2WebClient** kliknij prawym przyciskiem myszy opcję `WebContent\WEB-INF\conf\axis2.xml` > **Otwórz za pomocą ...** > **Edytor XML**.

b) Wyszukaj ostatni element `transportSender` (na końcu pliku) i odszukaj w komentarzu JMS `transportSender` > kliknij prawym przyciskiem myszy wiersz > **Dodaj przed ...** > **transportSender**.

c) Kliknij prawym przyciskiem myszy opcję **transportSender** > **Dodaj atrybut** > **Name** > prawym przyciskiem myszy kliknij opcję **transportSender** > **Dodaj atrybut** > **Class**.

d) Kliknij prawym przyciskiem myszy opcję **Nazwa** > **Edytuj atrybut** > Wpisz wartość **Wartość:** `jms`

e) Kliknij prawym przyciskiem myszy opcję **Klasa** > **Edytuj atrybut** > Wpisz wartość w polu **Wartość:** `com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender`. > Zapisz.

f) Dodaj plik `com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender` do ścieżki budowania: kliknij prawym przyciskiem myszy opcję **StockQuoteAxis2WebClient** > **Ścieżka budowania** > **Konfiguruj ścieżkę budowania ...** > Kliknij kartę **Biblioteki** > **Dodaj zewnętrzne pliki JAR ...**. Wybierz wszystkie pliki JAR w programie *MQ_INSTALLATION_PATH*`\java\lib` > **OK**.

MQ_INSTALLATION_PATH to katalog, w którym zainstalowany jest produkt WebSphere MQ .

4. Utwórz synchroniczny klient statyczny, przetestuj go przy użyciu protokołu HTTP, a następnie przekształć proxy w celu uruchomienia klienta statycznego przy użyciu transportu produktu WebSphere MQ dla protokołu SOAP.

a) Kliknij prawym przyciskiem myszy opcję **Zasoby Java: src** > **Nowy** > **Pakiet** > Nazwa pakietu `soap.client` > Zakończ

b) Kliknij prawym przyciskiem myszy opcję **soap.client** > **Nowy** > **Klasa** > Nazwa klasy `SQA2StaticClient` > **Zakończ**.

c) Zastąp klasę za pomocą następującego kodu, a następnie kliknij przycisk **Zapisz**.

Rysunek 182. SQA2DynamicClient.java

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
public class SQA2StaticClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub();
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("Response is: "
          + (stub.getQuote(request)).getGetQuoteReturn());
    } catch (Exception e) {
      System.out.println("Exception: " + e.getMessage());
      e.printStackTrace();
    }
  }
}
```

5. Przetestuj klienta za pomocą usługi Axis StockQuotewdrożonej w środowisku WASCE, a następnie z produktem WebSphere MQ transport for SOAP.

a) W eksploratorze projektów kliknij prawym przyciskiem myszy opcję **SQA2StaticClient** > **Uruchom jako ...** > **Aplikacja Java**.

Wynik, `Response is 55.25`, zostanie wyświetlony w widoku Konsola. Można również wybrać w widoku Konsola okno konsoli WASCE, a następnie wyświetlić dane wyjściowe na serwerze WASCE `StockQuoteAxis called with parameter: ibm`.

b) Serwer proxy został zbudowany przy użyciu adresu usługi `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis`, a więc klient statyczny wywołuje usługę działającą na serwerze HTTP. Istnieje możliwość zmiany klienta statycznego w taki sposób, aby wywoła usługę za pomocą transportu WebSphere MQ dla protokołu SOAP. Poniższe instrukcje zmieniają adres usługi w produkcie `StockQuoteAxisServiceStub.java` bez odbudowowania proxy i konfigurowania parametrów środowiska wykonawczego `SQA2StaticClient` na potrzeby

ładowania systemu `axis2.xml`. Skonfigurowanie produktu `axis2.xml` umożliwia skonfigurowanie produktu Axis2 w taki sposób, aby używany był transport produktu WebSphere MQ dla protokołu SOAP.

c) Otwórz program `StockQuoteAxisServiceStub.java` >
Zastąp dwa wystąpienia produktu `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis` ,

```
jms:/queue?destination=REQUESTAXIS@QM1
&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
&targetService=StockQuoteAxis
&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
```

d) Jeśli produkt `SQA2StaticClient` zostanie uruchomiony teraz, zgłaszany jest wyjątek, ponieważ nie znalazł on skonfigurowanego dla usługi JMS elementu `transportSender` .
Wyjątek:

```
Exception: null java.lang.NullPointerException at
soap.server.StockQuoteAxisServiceStub.getQuote(StockQuoteAxisServiceStub.java:547)
at soap.client.SQA2StaticClient.main(SQA2StaticClient.java:11)
```

e) W eksploratorze projektów kliknij prawym przyciskiem myszy opcję **SQA2StaticClient** > **Uruchom jako ...** > **Uruchom konfiguracje ...**. Przejdź do karty **(x) = Argumenty** , a w obszarze wejściowym **Argumenty maszyny VM** wpisz ścieżkę do pliku `axis2.conf` > **Zastosuj** > **Uruchom**.
Argument maszyny VM to: `-Daxis2.xml=${workspace_loc:StockQuoteAxis2WebClient/WebContent/WEB-INF/conf}/axis2.xml`. Można też podać standardową ścieżkę do pliku konfiguracyjnego Axis2 .

f) Ponownie uruchom program `SQA2StaticClient` . W tym przypadku używany jest transport produktu WebSphere MQ dla protokołu SOAP. Potwierdź go, sprawdzając, czy w konsoli WASCE nie ma żadnych nowych danych wyjściowych. Otwórz konsolę lub okno komend, które jest powiązane z programem nasłuchującym SimpleJava, a dane wyjściowe w tym oknie są `StockQuoteAxis called with parameter: ibm`.

6. Utwórz klienta dynamicznego dla transportu HTTP i produktu WebSphere MQ dla protokołu SOAP i przetestuj go.

a) Kliknij prawym przyciskiem myszy opcję **soap.client** > **Nowy** > **Klasa** > Nazwa klasy `SQA2DynamicClient` > **Zakończ**.

b) Zastąp klasę za pomocą następującego kodu, a następnie kliknij przycisk **Zapisz**.

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
public class SQA2DynamicClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("HTTP Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      stub = new StockQuoteAxisServiceStub(
          "jms:/queue?destination=REQUESTAXIS@QM1"
          + "&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&targetService=StockQuoteAxis&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE");
       System.out.println("JMS sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
     } catch (Exception e) {
       System.out.println("Exception: " + e.getMessage());
       e.printStackTrace();
     }
   }
 }
```

c) Utwórz konfigurację uruchomienia dla produktu `SQA2DynamicClient.java`i dodaj ścieżkę do produktu `axis2.xml`:

```
-Daxis2.xml=${workspace_loc:StockQuoteAxis2WebClient/WebContent/WEB-INF/
conf}/axis2.xml
```

d) Uruchom program SQA2DynamicClient. Sprawdź dane wyjściowe konsoli dla serwerów SQA2DynamicClient, WASCE i **SimpleJavaListener**.

7. Utwórz klienta asynchronicznego i uzyskaj dostęp do wyniku w procedurze obsługi wywołania zwrotnego, a także w głównym wątku programu.

Asynchroniczne proxy klienta utworzone przez kreator usług Web Service dla środowiska IDE Eclipse Java EE for Web Developers różnią się od proxy utworzonych przez produkt **wsimport**. Serwery proxy produktu **wsimport** używają typów ogólnych Future, Responsei AsyncHandler .

Kreator usług Web Service dla środowiska IDE Eclipse Java EE dla programistów WWW tworzy klasę abstrakcyjną StockQuoteAxisServiceCallbackHandler . Należy rozszerzyć produkt StockQuoteAxisServiceCallbackHandler i utworzyć procedurę obsługi wywołania zwrotnego.

a) Kliknij prawym przyciskiem myszy opcję **soap.client** > **Nowy** > **Klasa** > Nazwa klasy SQA2CallbackHandler > **Zakończ**.

b) Zastąp klasę następującym kodem.

```
package soap.client;
import soap.server.StockQuoteAxisServiceCallbackHandler;
import soap.server.StockQuoteAxisServiceStub.GetQuoteResponse;
public class SQA2CallbackHandler
    extends StockQuoteAxisServiceCallbackHandler {
  private boolean complete = false;
  SQA2CallbackHandler() {
    super();
    System.out.println("Callback constructor");
  }
  public void receiveResultgetQuote(GetQuoteResponse response) {
    System.out.println("Result in Callback " + response.getGetQuoteReturn());
    super.clientData = response;
    complete = true;
  }
  public boolean isComplete() {
    return complete;
  }
}
```

c) Kliknij prawym przyciskiem myszy opcję **soap.client** > **Nowy** > **Klasa** > Nazwa klasy SQA2AsyncClient > **Zakończ**.

d) Zastąp klasę następującym kodem.

Rysunek 183. SQA2AsyncClient.java

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
import soap.server.StockQuoteAxisServiceStub.GetQuoteResponse;
import soap.server.StockQuoteAxisServiceCallbackHandler;
@SuppressWarnings("unused")
public class SQA2AsyncClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("HTTP Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      SQA2CallbackHandler callback = new SQA2CallbackHandler();
      stub.startgetQuote(request, callback);
      do {
        System.out.println("Waiting for HTTP callback");
        Thread.sleep(2000);
      } while (!callback.isComplete());
      System.out.println("HTTP poll: "
          + ((GetQuoteResponse) (callback.getClientData()))
                .getGetQuoteReturn());
      stub = new StockQuoteAxisServiceStub(
```

```
          "jms:/queue?destination=REQUESTAXIS@QM1"
          + "&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&targetService=StockQuoteAxis&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE");
      System.out.println("JMS Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      callback = new SQA2CallbackHandler();
      stub.startgetQuote(request, callback);
      while (!callback.isComplete()) {
        System.out.println("Waiting for JMS callback");
        Thread.sleep(2000);
      }
      System.out.println("JMS poll: "
        + ((GetQuoteResponse) (callback.getClientData())).getGetQuoteReturn());
    } catch (Exception e) {
        System.out.println("Exception: " + e.getMessage());
        e.printStackTrace();
     }
   }
 }
```

Dane wyjściowe konsoli są następujące:

```
HTTP Sync: 55.25
Callback constructor
Waiting for HTTP callback
Result in Callback 55.25
HTTP poll: 55.25
JMS Sync: 55.25
Callback constructor
Waiting for JMS callback
Result in Callback 55.25
JMS poll: 55.25
```

Przykładowe klienty Axis2

Przykładowe serwery proxy są generowane przy użyciu narzędzia **wsimport** , które jest spakowane z językiem Java 6. Dostępnych jest sześć próbek:

1. DynamicProxyClientSync.java
2. DynamicProxyClientAsyncPolling.java
3. DynamicProxyClientAsyncCallback.java
4. DispatchClientSync.java
5. DispatchClientAsyncPolling.java
6. DispatchClientAsyncCallback.java

Przykłady klienta są generowane dla przykładowego serwera osi StockQuote. Wygeneruj plik WSDL za pomocą komendy **amqwdepoyWMQServer** , określając przełącznik -w , aby wybrać styl `rpcLiteral` . Aby wygenerować proxy dla przykładów, należy użyć następującej komendy:

```
wsimport soap.server.StockQuoteAxis_Wmq.wsdl -d generated -keep -p com.ibm.mq.axis2.samples
```

Rysunek 184. DynamicProxyClientSync.java

```
package com.ibm.mq.axis2.samples;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientSync {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientSync");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();
```

```
      System.out.println("Invoking getQuoteOneWay OneWay operation synchronously...");
      service.getQuoteOneWay("48");
      System.out.println(" > getQuoteOneWay has returned");

      System.out.println("Invoking getQuote Request Reply operation synchronously...");
      float result = service.getQuote("48");
      System.out.println(" > getQuote has returned result of " + result);

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Rysunek 185. DynamicProxyClientAsyncPolling.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.CancellationException;

import javax.xml.ws.Response;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientAsyncPolling {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientAsyncPolling");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      System.out
          .println("Invoking getQuoteAsync Request Reply operation asynchronously by
polling...");
      Response<Float> response = service.getQuoteAsync("49");

      /** Sleep main thread until response arrives **/
      System.out.println("Waiting for response to arrive...");
      while (!response.isDone()) {
        Thread.sleep(100);
      }
      System.out.println(" > Response received");

      /** Retrieve the result **/
      try {
        Float result = response.get();
        System.out.println(" > getQuoteAsync call has returned result of " + result);
      }
      catch (CancellationException ce) {
        // processing was cancelled via response.cancel()
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
```

```
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Rysunek 186. DynamicProxyClientAsyncCallback.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.Future;

import javax.xml.ws.AsyncHandler;
import javax.xml.ws.Response;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientAsyncCallback implements AsyncHandler<Float> {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientAsyncCallback");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      DynamicProxyClientAsyncCallback handler = new DynamicProxyClientAsyncCallback();

      System.out
          .println("Invoking getQuoteAsync Request Reply operation asynchronously using a
callback...");
      Future<?> monitor = service.getQuoteAsync("50", handler);
      System.out.println(" > Invoke call has returned");

      /** Sleep main thread until handler has been notified **/
      System.out.println("Waiting for handler to be called...");
      while (!monitor.isDone()) {
        Thread.sleep(100);
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
```

```
  public void handleResponse(Response<Float> response) {
    try {
      Float result = response.get();
      System.out.println(" > Async Handler has received a result of " + result);
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println("Exception in handleResponce");
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

}
```

Rysunek 187. DispatchClientSync.java

```
package com.ibm.mq.axis2.samples;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientSync {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientSync");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service **/
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /************************************
       * Create OneWay SOAPMessage request.
       ************************************/
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("\nCreating a OneWay SOAP Message");
      SOAPMessage request = mf.createMessage();
```

```
      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();

      /** Construct the message payload **/
      SOAPElement operation = body.addChildElement("getQuoteOneWay", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuoteOneWay OneWay operation synchronously...");
      dispatch.invokeOneWay(request);
      System.out.println(" > getQuoteOneWay call has returned");

      /*****************************************
       * Create Request Reply SOAPMessage request.
       *****************************************/
      mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("\nCreating a Request Reply SOAP Message");
      request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      part = request.getSOAPPart();
      env = part.getEnvelope();
      header = env.getHeader();
      body = env.getBody();

      /** Construct the message payload **/
      operation = body.addChildElement("getQuote", "ns1", "soap.server.StockQuoteAxis_Wmq");
      value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuote Request Reply operation synchronously...");
      SOAPMessage ans = dispatch.invoke(request);
      System.out.println(" > getQuote call has returned");

      /** Retrieve the result **/
      part = ans.getSOAPPart();
      env = part.getEnvelope();
      body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      System.out.println("Parsing SOAP response...");
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String message = responseElement.getValue();
      System.out.println(" > Response contains result of " + message);

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
```

```
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Rysunek 188. DispatchClientAsyncPolling.java

```
package com.ibm.mq.axis2.samples;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Response;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientAsyncPolling {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientAsyncPolling");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service.* */
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /** Create SOAPMessage request. * */
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("Creating a Request Reply SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();

      /** Construct the message payload **/
      SOAPElement operation = body.addChildElement("getQuote", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuote Request Reply operation asynchronously by
polling...");
      Response<SOAPMessage> response = dispatch.invokeAsync(request);
      System.out.println(" > getQuote call has returned");
```

```
      /** Sleep main thread until response arrives **/
      System.out.println("Waiting for response to arrive...");
      while (!response.isDone()) {
        Thread.sleep(100);
      }
      System.out.println(" > Response received");

      /** retrieve the result **/
      SOAPMessage ans = response.get();
      part = ans.getSOAPPart();
      env = part.getEnvelope();
      body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String message = responseElement.getValue();
      System.out.println(" > Response contains result of " + message);

      System.out.println("End of sample");

    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Rysunek 189. DispatchClientAsyncCallback.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.Future;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.AsyncHandler;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Response;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientAsyncCallback implements AsyncHandler<SOAPMessage> {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientAsyncCallback");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
```

```
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service.* */
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /** Create SOAPMessage request. * */
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("Creating a Request Reply SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();

      /** Construct the message payload. **/
      SOAPElement operation = body.addChildElement("getQuote", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint. **/
      DispatchClientAsyncCallback handler = new DispatchClientAsyncCallback();

      System.out
          .println("Invoking getQuote Request Reply operation asynchronously using a
callback...");
      Future<?> monitor = dispatch.invokeAsync(request, handler);
      System.out.println(" > getQuote call has returned");

      /** Sleep main thread until handler has been notified **/
      System.out.println("Waiting for handler to be called...");
      while (!monitor.isDone()) {
        Thread.sleep(100);
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

  public void handleResponse(Response<SOAPMessage> response) {
    try {
      // retrieve the result
      SOAPMessage ans = response.get();
      SOAPPart part = ans.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPBody body = env.getBody();
```

```
      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String result = responseElement.getValue();

      System.out.println(" > Async Handler has received a result of " + result);
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println("Exception in handleResponce");
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

}
```

Zadania pokrewne

Projektowanie klienta JAX-RPC dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse

Należy opracować klienta usługi Web Service firmy Axis 1.4 , który będzie uruchamiany przy użyciu transportu produktu WebSphere MQ dla protokołu SOAP.

Tworzenie klienta .NET 1 lub 2 dla transportu WebSphere dla protokołu SOAP za pomocą programu Microsoft Visual Studio 2008

Należy utworzyć klient usługi Web Service .NET 1 lub 2, który ma być uruchamiany przy użyciu transportu WebSphere MQ dla protokołu SOAP.

Tworzenie klienta .NET 1 lub 2 dla transportu WebSphere dla protokołu SOAP za pomocą programu Microsoft Visual Studio 2008

Należy utworzyć klient usługi Web Service .NET 1 lub 2, który ma być uruchamiany przy użyciu transportu WebSphere MQ dla protokołu SOAP.

Zanim rozpoczniesz

Rozwój klienta .NET 1 lub 2 można rozpocząć na wiele różnych sposobów:

1. Produkt **amqwdeployWMQService** służy do generowania kodów pośredniczących klientów z usługi Web Service i importowania ich do produktu Visual Studio.
2. Produkt **java2wsdl** służy do generowania pliku WSDL z implementacji Java usługi Web Service, a następnie użycia produktu `wsdl.exe`, który jest dostarczany razem z platformą .NET, w celu wygenerowania kodów pośredniczących klienta.
3. Wygeneruj plik WSDL na podstawie implementacji usługi .NET `.asmx` przy użyciu produktu **amqswsdl**, a następnie użyj produktu `wsdl.exe`.
4. Jeśli usługa dla protokołu HTTP została opracowana i wdrożona, należy użyć opcji **Dodaj odwołanie do sieci WWW ...** . Kreator w programie Visual Studio umożliwia skonfigurowanie klienta w celu uzyskania dostępu do usługi HTTP. Zmień adres URL w odwołaniu do usługi wdrożonej w produkcie WebSphere MQ transport dla protokołu SOAP.

Zadanie korzysta z usługi opracowanej w produkcie “Projektowanie usługi .NET 1 lub 2 dla transportu WebSphere MQ dla protokołu SOAP za pomocą programu Microsoft Visual Studio 2008” na stronie 993.

O tym zadaniu

Wykonaj poniższe kroki, aby utworzyć klient .NET 1 lub 2 dla protokołu HTTP i transportu WebSphere MQ dla protokołu SOAP.

Procedura

1. Utwórz aplikację konsoli klienta i zmodyfikuj ją w taki sposób, aby wywoływała usługę Web Service HTTP StockQuote .
   a) Kliknij prawym przyciskiem myszy opcję **Rozwiązanie 'StockQuoteDotNet'** w oknie **Eksplorator rozwiązań** > Dodaj ... > Nowy projekt. Wybierz typ projektu **C#** , **.Środowisko NET Framework 2.0**i **Aplikacja konsoli**. Nadaj nazwę projektowi `StockQuoteClientDotNet` > **OK**
   b) Kliknij prawym przyciskiem myszy opcję **Rozwiązanie 'StockQuoteDotNet'** w oknie **Eksplorator rozwiązań** > Dodaj ... > Nowy projekt. Wybierz typ projektu **C#** , **.Środowisko NET Framework 2.0**i **Aplikacja konsoli**. Nadaj nazwę projektowi `StockQuoteClientDotNet` > **OK**
   c) Kliknij prawym przyciskiem myszy opcję **StockQuoteClientDotNet** > **Set as Startup project**(Ustaw jako projekt startowy).
   d) Kliknij prawym przyciskiem myszy opcję **StockQuoteClientDotNet** > **Add Web Reference ...** > Browse to Web services in this solution > Wybierz kolejno opcje **StockQuote** > **Add Reference**. Należy zauważyć, że dodano odwołanie do sieci WWW do hosta lokalnego i nowego pliku konfiguracyjnego `app.config`.
   e) W eksploratorze rozwiązań zmień nazwę aplikacji konsoli z `Program.cs` na `StockQuoteClientDotNet.cs` > Kliknij przycisk **OK** , aby zmienić wszystkie zastosowania produktu `Program.cs` na wartość `StockQuoteClientDotNet.cs`.
   f) Zastąp zawartość produktu `StockQuoteClientDotNet.cs` kodem w produkcie Rysunek 190 na stronie 1025.

```
using System;
using StockQuoteClientDotNet.localhost;
namespace StockQuoteClientDotNet {
  class StockQuoteClientDotNet {
    static void Main(string[] args) {
      try {
        StockQuote stockobj = new StockQuote();
        Console.WriteLine("http reply is: "
          + stockobj.getNonInlineQuote("http request");
      }
      catch (System.Exception e) {
        Console.WriteLine("Exception thrown: " + e.ToString());
      }
      Console.ReadLine();
    }
  }
}
```

Rysunek 190. Program sieciowy HTTP StockQuoteClientDot

   g) Uruchom program StockQuoteClientDotNet w celu przetestowania na usłudze `StockQuote.asmx` :
      i) Naciśnij klawisz **F5**, kliknij zieloną strzałkę na pasku działań lub opcję **Debuguj** > **Uruchom debugowanie (F5)**.

      Jeśli projekt StockQuoteDOtNet znajduje się w tym samym rozwiązaniu, uruchamiany jest automatycznie. W przeciwnym razie należy najpierw uruchomić usługę.

      Okno komend z wynikami zostanie otwarte za obszarem roboczym. Instrukcja `Console.ReadLine();` zapobiega jej zamykaniu do momentu naciśania klawisza **Enter**.

Wskazówka: Upewnij się, że `StockQuote.asmx` jest stroną początkową w projekcie `StockQuoteDotNet` .

2. Zmodyfikuj plik StockQuoteClientDotNet, aby wywołać usługę `StockQuote.asmx` przy użyciu transportu WebSphere MQ dla protokołu SOAP.

   a) Dodaj wiersze oznaczone pogrubioną czcionką do klienta.

```
using System;
using StockQuoteClientDotNet.localhost;
namespace StockQuoteClientDotNet {
  class StockQuoteClientDotNet {
    static void Main(string[] args) {
      try {
        IBM.WMQSOAP.Register.Extension();
        StockQuote stockobj = new StockQuote();
        Console.WriteLine("http reply is: "
          + stockobj.getNonInlineQuote("http request");
        stockobj.Url = "jms:/queue?"
          + "initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&connectionFactory=()&destination=REQUESTDOTNET@QM1"
          + "&targetService=StockQuote.asmx";
        Console.WriteLine("jms reply is: "
          + stockobj.getNonInlineQuote("jms request"));
      }
      catch (System.Exception e) {
        Console.WriteLine("Exception thrown: " + e.ToString());
      }
      Console.ReadLine();
    }
  }
}
```

Rysunek 191. Zmodyfikowano program StockQuoteClientDotNet

   Alternatywnie zmodyfikuj domyślny adres URL. Otwórz kolejno opcje **StockQuoteClientDotNet** > **Properties** > **Settings.settings** i zmień wartość właściwości `StockQuoteClientDotNet_localhost_StockQuote` na transport WebSphere MQ dla adresu URL SOAP.

   b) Dodaj odwołanie do produktu `amqsoap.dll`

      i) W projekcie **StockQuoteClientDotNet** w **Eksploratorze rozwiązań**kliknij prawym przyciskiem myszy opcję **Odwołania** > **Dodaj odwołanie ...** > Kliknij kartę **Przeglądaj** > przejdź do opcji *MQ_INSTALLATION_PATH*\bin > Wybierz opcję **amqsoap.dll** > **OK**. *MQ_INSTALLATION_PATH* Jest to katalog, w którym zainstalowano produkt WebSphere MQ .

3. Przetestuj klient za pomocą usługi `StockQuote.asmx` przy użyciu transportu WebSphere MQ dla protokołu SOAP.

   a) Otwórz okno komend w katalogu projektu StockQuoteDotNet : `.\StockQuoteDotNet\StockQuoteDotNet` > Sprawdź, czy produkt `.bin\StockQuoteDotNet.dll` istnieje. Jeśli nie, odbuduj rozwiązanie.

   b) Wpisz komendę **amqwRegisterdotNet**.

   Produkt **amqwRegisterdotNet** należy uruchomić tylko raz na instalację.

   c) Jeśli produkt **amqwdeployWMQServer** został uruchomiony z produktem `genAsmxWMQBits`, uruchom program nasłuchujący SOAP .NET: `generated\server\startWMQNListener`

   d) Alternatywnie uruchom program nasłuchujący bezpośrednio:

```
amqwSOAPNETListener -u "jms:/queue?
destination=REQUESTDOTNET@QM1
&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi
&targetService=StockQuote.asmx&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE"
-w C:\IBM\ID\StockQuoteDotNet\StockQuoteDotNet -n 10
```

4. W programie Visual Studio 2008 naciśnij klawisz **F5** , aby uruchomić program StockQuoteClientDotNet.

Klienty usług Web Service środowiska .NET Framework 1 i .NET Framework 2

Przykładowe klienty .NET dostarczane z transportem WebSphere MQ dla SOAP korzystają z wygenerowanych kodów pośredniczących w celu wywołania przykładowych usług Axis i .NET.

W przypadku klientów środowiska .NET Framework 1 i .NET Framework 2 produkt WebSphere MQ zapewnia dostęp do usług Web Service przy użyciu klientów .NET. Komenda **amqwdeployWMQService** ma opcję `genProxiestoDotNet`, która generuje kody pośredniczące klienta .NET Framework 1 lub .NET Framework 2 dla usługi Web Service. Można również użyć kodów pośredniczących klienta wygenerowanych przez narzędzie .NET **wsdl** lub przez program Microsoft Visual Studio 2005 lub 2008.

Przykładowe klienty usług Web Service .NET Framework 1 i .NET zostały zainstalowane w produkcie *MQ_INSTALLATION_PATH*`\tools\soap\samples\dotnet`. *MQ_INSTALLATION_PATH* Jest to katalog, w którym zainstalowano produkt WebSphere MQ .

SQVB2Axis.vb
SQVB2Axis.vb(Rysunek 192 na stronie 1027) to klient Visual Basic, który może wywoływać usługę **StockQuoteAxisService** .

SQVB2DotNet.vb
QVB2DotNet.vb(Rysunek 193 na stronie 1027) to klient Visual Basic, który może wywoływać usługę **StockQuoteDotNet** .

SQCS2Axis.cs
SQCS2Axis.cs(Rysunek 194 na stronie 1028) to klient C#, który może wywołać usługę **StockQuoteAxisService** . Adres URL usługi można przesłonić, podając adres URL w wierszu komend.

SQCS2DotNet.cs
SQCS2DotNet.cs(Rysunek 195 na stronie 1028) to klient C#, który może wywołać usługę **StockQuoteDotNet** . Adres URL usługi można przesłonić, podając adres URL w wierszu komend.

```
Module SQVB2Axis
   Function Main(ByVal CmdArgs() As String) As Integer
      IBM.WMQSOAP.Register.Extension()
      Dim obj As New StockQuoteAxisService()
      Dim res As Single = obj.getQuote("fromcs")
      System.Console.WriteLine("SQVB2Axis: reply is: '{0}'", res)
   End Function
End Module
```

Rysunek 192. SQVB2Axis

```
Module SQVB2DotNet
   Function Main(ByVal CmdArgs() As String) As Integer
      IBM.WMQSOAP.Register.Extension()
      Dim obj as new StockQuoteDotNet()
      Dim res as Single = obj.getQuote("fromcs")
      System.Console.WriteLine("SQVB2DotNet: reply is: '{0}'", res)
   End Function
End Module
```

Rysunek 193. SQVB2DotNet

```
using System;
class SQCS2Axis {
   [STAThread]
   static void Main(string[] args) {
      try {
         IBM.WMQSOAP.Register.Extension();
         StockQuoteAxisService stockobj = new StockQuoteAxisService();
         if (args.GetLength(0) >= 1)
            stockobj.Url = args[0];
         System.Single res = stockobj.getQuote("XXX");
         Console.WriteLine("SQCS2Axis RPC reply is: " + res);
      }
      catch (System.Exception e) {
         Console.WriteLine("\n>>> EXCEPTION WHILE RUNNING SQCS2Axis DEMO <<<\n"
            + e.ToString());
      }
   }
}
```

Rysunek 194. SQCS2Axis

```
using System;
class SQCS2DotNet {
   [STAThread]
   static void Main(string[] args) {
      try {
         IBM.WMQSOAP.Register.Extension();
         StockQuoteDotNet stockobj = new StockQuoteDotNet();
         if (args.GetLength(0) >= 1)
            stockobj.Url = args[0];
         System.Single res = stockobj.getQuote("XXX");
         Console.WriteLine("RPC reply is: " + res);
         if (args.GetLength(0) == 0) {
            res = stockobj.getQuoteDOC("XXX");
            Console.WriteLine("DOC reply is: " + res);
         }
      }
      catch (System.Exception e) {
         Console.WriteLine("\n>>> EXCEPTION WHILE RUNNING SQCS2DotNet DEMO <<<\n"
            + e.ToString());
      }
   }
}
```

Rysunek 195. SQCS2DotNet

Zadania pokrewne

Projektowanie klienta JAX-RPC dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse

Należy opracować klienta usługi Web Service firmy Axis 1.4 , który będzie uruchamiany przy użyciu transportu produktu WebSphere MQ dla protokołu SOAP.

Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse

Należy opracować klienta usługi Web Service Axis2 , który będzie uruchamiany przy użyciu transportu WebSphere MQ dla protokołu SOAP. Zostaną wyświetlone przykładowe klienty Axis2 dostarczane z produktem WebSphere MQ transport dla protokołu SOAP, a także komenda **wsimport** używana do generowania proxy.

Wdrażanie usług Web Services przy użyciu transportu WebSphere MQ dla protokołu SOAP

Wdróż usługę Web Service w jednej z wielu różnych środowisk serwera i nawiążaj z nią połączenie przy użyciu transportu WebSphere MQ dla protokołu SOAP.

Zanim rozpoczniesz

Utwórz usługę Web Service i przetestuj ją przy użyciu protokołu SOAP za pośrednictwem protokołu HTTP w środowisku docelowym.

O tym zadaniu

Usługę Web Service można wdrożyć w celu uruchomienia z transportem WebSphere MQ dla protokołu SOAP w wielu różnych środowiskach środowiska wykonawczego SOAP. Usługę można wdrożyć w firmie Axis 1.4 , korzystając tylko z oprogramowania zainstalowanego w produkcie WebSphere MQ. W przypadku innych środowisk środowiska wykonawczego konieczne jest zainstalowanie dodatkowego oprogramowania.

Nie jest ograniczony do uruchamiania transportu produktu WebSphere MQ dla protokołu SOAP do serwerów, dla których istnieją instrukcje wdrażania. Użyj instrukcji, aby wdrożyć usługę w jednym z wymienionych środowisk.

Uwaga: Niektóre zintegrowane środowiska oferują protokół SOAP korzystający z usługi JMS przy użyciu rekomendowanego powiązania SOAP W3C , a także transportu WebSphere MQ dla powiązania SOAP. Wersje produktu WebSphere MQ, do wersji 7.0.1.2włącznie, obsługują tylko transport produktu WebSphere MQ do powiązania SOAP. Począwszy od wersji 7.0.1.3 można wdrażać klienty Axis2 przy użyciu identyfikatora URI zgodnego z rekomendacją kandydata W3C dla protokołu SOAP korzystającego z usługi JMS. Informacje na ten temat zawiera kurs Tworzenie aplikacji usług Web Service JAX-WS SOAP/JMS z produktami WebSphere Application Server V7 i Rational Application Developer V7.5.

Wdrażanie usługi w firmie Axis 1.4 w celu użycia w transporcie WebSphere dla protokołu SOAP przy użyciu produktu `amqwdeployWMQService`

Wdróż usługę Axis 1.4 w transporcie WebSphere MQ dla protokołu SOAP, tworząc katalog wdrożenia, uruchamiając komendę **`amqwdeployWMQService`** i uruchamiając program nasłuchujący Axis 1.4 .

Zanim rozpoczniesz

1. Postępuj zgodnie z instrukcjami dotyczącymi instalowania produktu WebSphere MQ transport for SOAP
2. Sprawdź poprawność instalacji i środowiska za pomocą komendy **`runivt`** .
3. Aby ponownie wdrożyć usługę:
   a. Usuń podkatalog `./generated` i wszystkie jego podkatalogi.
   b. Usuń żądania z kolejki docelowej i usuń je.
   c. Należy postępować zgodnie z instrukcjami z kroku “2” na stronie 1029.

O tym zadaniu

Te instrukcje służą do wdrażania usługi Axis 1.4 po raz pierwszy. Aby zrestartować usługę Axis 1.4 , ponownie uruchom program nasłuchujący SOAP Axis 1.4 : krok “11” na stronie 1030.

Aby wdrożyć nową usługę Axis 1.4 w transporcie WebSphere MQ dla protokołu SOAP, należy wykonać następujące instrukcje:

Procedura

1. Utwórz katalog *`deployDir`* , w którym będą przechowywane pliki wdrożenia.
   Program narzędziowy do wdrażania wymaga, aby każda usługa została wdrożona z osobnego katalogu.
2. Otwórz okno komend w systemie Windowslub powłokę komend, używając systemu X Window System w systemach UNIX and Linux , w programie *`deployDir`* , aby uruchomić program **`amqwdeployWMQService`**.
3. Uruchom program **`amqwsetcp`** , aby ustawić ścieżkę klasy.

Środowisko JRE i JDK musi znajdować się w ścieżce klasy (w wersji 5.0 lub nowszej) i na tym samym poziomie wersji.

4. Skopiuj źródło klasy *className*.javado produktu *deployDir* .
5. Skopiuj wszystkie pliki źródłowe Java z tego samego pakietu co *className* do produktu *deployDir*/*packageName*, gdzie *packageName* to drzewo katalogów odpowiadające nazwie pakietu.
6. Uruchom **javac** *packageName*.*className*.

   W celu znalezienia innych klas może być konieczne dodanie ścieżki do bieżącego katalogu "."lub do katalogu *packageName* (**javac**).
7. Utwórz plik WSDL środowiska Axis dla usługi:

```
amqwdeployWMQService -f packageName.className.java -c genAxisWsdl
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

8. Utwórz zasoby produktu WebSphere MQ dla usługi:

```
amqwdeployWMQService -f packageName.className.java -c genAxisWMQBits
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

   Wskazówka:

   Aby skonfigurować nowy menedżer kolejek i wymagane zasoby, aby wykonać programowanie i testowanie, uruchom program **setupWMQSOAP**.

   Jeśli nowy menedżer kolejek ma zostać skonfigurowany jako domyślny, należy wykonać kopię produktu **setupWMQSOAP** z katalogu *WMQ install directory*\tools\soap\samples i dodać parametr -q do wiersza.

```
call :try -q crtmqm %QMGR%
```

9. Utwórz program nasłuchujący Axis i wdróż usługę:

```
amqwdeployWMQService -f packageName.className.java -c AxisDeploy
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

10. Jeśli konieczne jest wygenerowanie pliku WSDL dla usługi, generowanie kodów pośredniczących klienta lub proxy klienta, należy uruchomić produkt **amqwdeployWMQService** z jednym z następujących parametrów:

    - genAsmxWsdl
    - genAxisWsdl
    - genProxiesToDotNet
    - genProxiestoOś

    Uwaga: Przed wygenerowaniem proxy należy wygenerować plik WSDL. Opcja AllAxis kończy się niepowodzeniem, jeśli zmienna CLASSPATH nie jest ustawiona w celu znalezienia wszystkich klas, które są importowane do kompilacji *className*.java. Jeśli w pakiecie zawierającym *className*.javaistnieje wiele plików Java, należy je skompilować najpierw za pomocą programu **javac**. Program **amqwdeployWMQService** -f *packageName*.*className*.java -c CompileJava kompiluje tylko produkt *className*.java.
11. Uruchom wygenerowany program nasłuchujący Axis.

```
.\generated\server\startWMQJListener.cmd
```

Zadania pokrewne

Wdrażanie usługi w usłudze .NET Framework 1 lub 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP

Wdróż usługę .NET Framework 1 lub 2 w produkcie WebSphere MQ transport dla protokołu SOAP. Utwórz katalog wdrażania, uruchom komendę **amqwdeployWMQService** i uruchom program nasłuchujący .NET.

Wdrażanie usługi na serwerze CICS Transaction Server w celu użycia produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z obsługą usług Web Services produktu CICS Transaction Server 4.1 .

Wdrażanie usługi na serwerze WebSphere Application Server w celu używania produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z magistralą integracji usług na serwerze aplikacji WebSphere Application Server.

Konfigurowanie serwera WebSphere Application Server do korzystania z protokołu W3C SOAP over JMS
Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji Java EE . To zadanie jest krokiem 1. połączenia klienta usługi Web Service Axis2 i usługi Web Service wdrożonej na serwerze WebSphere Application Server przy użyciu protokołu W3C SOAP over JMS. Skonfiguruj zasoby WebSphere MQ i WebSphere Application Server , aby projektować i wdrażać usługę Web Service powiązaną z usługą W3C SOAP przez JMS jako transport.

Wdrażanie usługi w produkcie WebSphere ESB i punkcie końcowym usługi Process Server w celu użycia produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP nie jest obsługiwany bezpośrednio przez produkty WebSphere ESB i Process Server. Należy skonfigurować niestandardowy eksport.

Wdrażanie usługi w usłudze .NET Framework 1 lub 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP

Wdróż usługę .NET Framework 1 lub 2 w produkcie WebSphere MQ transport dla protokołu SOAP. Utwórz katalog wdrażania, uruchom komendę **amqwdeployWMQService** i uruchom program nasłuchujący .NET.

Zanim rozpoczniesz

1. Postępuj zgodnie z instrukcjami dotyczącymi instalowania produktu WebSphere MQ transport for SOAP
2. Sprawdź poprawność instalacji i środowiska za pomocą komendy **runivt** .
3. Ścieżka do plików środowiska .NET `wsdl.exe` i `csc.exe` musi być ustawiona. Kopie produktów `wsdl.exe` i `csc.exe` identyfikowanych przez zmienną PATH muszą być na tym samym poziomie środowiska .NET. Jeśli jest zainstalowanych wiele środowisk .NET lub korzysta się z produktu Visual Studio, należy uważnie sprawdzić zmienną PATH .
4. Aby ponownie wdrożyć usługę:
   a. Usuń podkatalog `./generated` i wszystkie jego podkatalogi
   b. Usuń żądania z kolejki docelowej i usuń je.
   c. Należy postępować zgodnie z instrukcjami z kroku “2” na stronie 1031.

O tym zadaniu

Te instrukcje służą do wdrażania usługi .NET po raz pierwszy. Aby zrestartować usługę .NET, ponownie uruchom program nasłuchujący SOAP .NET, krok “9” na stronie 1032.

Wykonaj następujące czynności, aby wdrożyć nową usługę .NET Framework 1 lub .NET Framework 2 w transporcie WebSphere MQ dla protokołu SOAP:

Procedura

1. Utwórz katalog *deployDir* , w którym będą przechowywane pliki wdrożenia.
   Program narzędziowy do wdrażania wymaga, aby każda usługa została wdrożona z osobnego katalogu.
2. Otwórz okno komend w programie *deployDir* , aby uruchomić program **amqwdeployWMQService**.

```
C:\IBM\ID\QuoteClient>
```

3. Uruchom program **amqwsetcp** , aby ustawić ścieżkę klasy.

   Ścieżka klasy jest wymagana tylko dla klientów Axis.
4. Skopiuj usługę .NET *className*.asmxdo produktu *deployDir* .
5. Zbuduj implementację usługi w bibliotece (.dll).

   Implementacja usługi wstawianej znajduje się w produkcie *className*.asmx. Implementacja usługi za pomocą kodu może mieć wartość *className*.asmx.cs.

   Rysunek 196 na stronie 1032 przedstawia przykład komendy do zbudowania usługi .NET Framework V2 jako biblioteki.

```
c:\WINDOWS\Microsoft.NET\Framework\v3.5\Csc.exe /noconfig /nowarn:1701,1702
/errorreport:prompt /warn:4 /define:TRACE
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.configuration.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Data.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Drawing.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Web.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Web.Services.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Xml.dll
/debug:pdbonly /filealign:512 /optimize+
/out:obj\Quote.dll /target:library Properties\AssemblyInfo.cs Quote.asmx.cs
```

Rysunek 196. Komenda budowania dla środowiska .NET Framework V2

6. Skopiuj *className*.dll do programu *deployDir**bin*.
7. Skonfiguruj zasoby produktu WebSphere MQ i utwórz obiekt nasłuchiwania wymagany do wykonania usługi:

```
amqwdeployWMQService -f className.asmx -c genAsmxWMQBits
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))
&targetService=className.asmx"
```

8. Jeśli konieczne jest wygenerowanie pliku WSDL dla usługi, generowanie kodów pośredniczących klienta lub proxy klienta, należy uruchomić produkt **amqwdeployWMQService** z jednym z następujących parametrów:

   - genAsmxWsdl
   - genAxisWsdl
   - genProxiesToDotNet
   - genProxiestoOś

   Uwaga: Przed wygenerowaniem proxy należy wygenerować plik WSDL.
9. Uruchom wygenerowany program nasłuchujący .NET.

```
.\generated\server\startWMQNListener.cmd
```

Zadania pokrewne

Wdrażanie usługi w firmie Axis 1.4 w celu użycia w transporcie WebSphere dla protokołu SOAP przy użyciu produktu amqwdeployWMQService
Wdróż usługę Axis 1.4 w transporcie WebSphere MQ dla protokołu SOAP, tworząc katalog wdrożenia, uruchamiając komendę **amqwdeployWMQService** i uruchamiając program nasłuchujący Axis 1.4 .

Wdrażanie usługi na serwerze CICS Transaction Server w celu użycia produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z obsługą usług Web Services produktu CICS Transaction Server 4.1 .

Wdrażanie usługi na serwerze WebSphere Application Server w celu używania produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z magistralą integracji usług na serwerze aplikacji WebSphere Application Server.

Konfigurowanie serwera WebSphere Application Server do korzystania z protokołu W3C SOAP over JMS
Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji Java EE . To zadanie jest krokiem 1. połączenia klienta usługi Web Service Axis2 i usługi Web Service wdrożonej na serwerze WebSphere Application Server przy użyciu protokołu W3C SOAP over JMS. Skonfiguruj zasoby WebSphere MQ i WebSphere Application Server , aby projektować i wdrażać usługę Web Service powiązaną z usługą W3C SOAP przez JMS jako transport.

Wdrażanie usługi w produkcie WebSphere ESB i punkcie końcowym usługi Process Server w celu użycia produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP nie jest obsługiwany bezpośrednio przez produkty WebSphere ESB i Process Server. Należy skonfigurować niestandardowy eksport.

Wdrażanie usługi na serwerze CICS Transaction Server w celu użycia produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z obsługą usług Web Services produktu CICS Transaction Server 4.1 .

Zanim rozpoczniesz

Należy użyć tych samych narzędzi do programowania dla klienta lub usługi dla produktu WebSphere MQ, tak jak ma to miejsce w przypadku programowania dla protokołu HTTP. Program CICS ma narzędzia odpowiadające **Java2wsdl** i **wsdl2Java**:

- Produkt **DFHWS2LS** przyjmuje opis usługi Web Service jako punkt początkowy. Używa on opisów komunikatów i typów danych używanych w tych komunikatach w celu konstruowania struktur danych języka wysokiego poziomu. W strukturach aplikacji można używać w programach napisanych w różnych językach.
- Produkt **DFHLS2WS** przyjmuje strukturę danych języka wysokiego poziomu jako punkt początkowy. Używa on struktury do konstruowania opisu usług Web Services, który zawiera opisy komunikatów. Tworzy również schematy dla komunikatów ze struktury danych języka.

Aby utworzyć usługę Web Service, należy postępować zgodnie z instrukcjami w sekcji Tworzenie usługi Web Service w dokumentacji produktu CICS .

O tym zadaniu
Należy postępować zgodnie z instrukcjami w sekcji Konfigurowanie programu CICS do korzystania z transportu produktu WebSphere MQ w dokumentacji produktu CICS . Korzystając z instrukcji, można wdrożyć usługę Web Service w transporcie produktu WebSphere MQ dla protokołu SOAP.
Zadania pokrewne
Wdrażanie usługi w firmie Axis 1.4 w celu użycia w transporcie WebSphere dla protokołu SOAP przy użyciu produktu amqwdeployWMQService
Wdróż usługę Axis 1.4 w transporcie WebSphere MQ dla protokołu SOAP, tworząc katalog wdrożenia, uruchamiając komendę **amqwdeployWMQService** i uruchamiając program nasłuchujący Axis 1.4 .

Wdrażanie usługi w usłudze .NET Framework 1 lub 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP
Wdróż usługę .NET Framework 1 lub 2 w produkcie WebSphere MQ transport dla protokołu SOAP. Utwórz katalog wdrażania, uruchom komendę **amqwdeployWMQService** i uruchom program nasłuchujący .NET.

Wdrażanie usługi na serwerze WebSphere Application Server w celu używania produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z magistralą integracji usług na serwerze aplikacji WebSphere Application Server.

Konfigurowanie serwera WebSphere Application Server do korzystania z protokołu W3C SOAP over JMS
Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji Java EE . To zadanie jest krokiem 1. połączenia klienta usługi Web Service Axis2 i usługi Web Service wdrożonej na serwerze WebSphere Application Server przy użyciu protokołu W3C SOAP over JMS. Skonfiguruj zasoby WebSphere MQ i WebSphere Application Server , aby projektować i wdrażać usługę Web Service powiązaną z usługą W3C SOAP przez JMS jako transport.

Wdrażanie usługi w produkcie WebSphere ESB i punkcie końcowym usługi Process Server w celu użycia produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP nie jest obsługiwany bezpośrednio przez produkty WebSphere ESB i Process Server. Należy skonfigurować niestandardowy eksport.

Wdrażanie usługi na serwerze WebSphere Application Server w celu używania produktu WebSphere Transport for SOAP

Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z magistralą integracji usług na serwerze aplikacji WebSphere Application Server.

Zanim rozpoczniesz

Aby utworzyć usługę Web Service, należy użyć produktu Rational Application Developer, produktu WebSphere Integration Developer lub pakietu usług Web Service.

O tym zadaniu

Aby wdrożyć usługę przy użyciu transportu WebSphere MQ dla protokołu SOAP jako transportu SOAP na serwerze WebSphere Application Server, należy wykonać następujące instrukcje.

Procedura

1. Skonfiguruj produkt WebSphere MQ jako dostawcę przesyłania komunikatów JMS dla magistrali integracji usług na serwerze aplikacji WebSphere Application Server.
2. Skonfiguruj zasoby produktu WebSphere MQ wymagane przez usługę.
3. Postępuj zgodnie z instrukcjami, Konfigurowanie zasobów JMS dla procesu nasłuchiwania punktów końcowych synchronicznego protokołu SOAP przez JMSw dokumentacji produktu WebSphere Application Server Network Deployment.

   Istnieją odpowiednie instrukcje dla innych platform serwera WebSphere Application Server.
4. Zmodyfikuj identyfikator URI usługi w taki sposób, aby był zgodny z transportem WebSphere MQ dla identyfikatora URI SOAP.
5. Wdróż usługę na serwerze WebSphere Application Server.

Co dalej

Wdróż usługę przy użyciu protokołu HTTP jako transportu, aby klienci mogli wysyłać zapytania do usługi i odbierać plik WSDL w odpowiedzi.

Zadania pokrewne

Wdrażanie usługi w firmie Axis 1.4 w celu użycia w transporcie WebSphere dla protokołu SOAP przy użyciu produktu amqwdeployWMQService
Wdróż usługę Axis 1.4 w transporcie WebSphere MQ dla protokołu SOAP, tworząc katalog wdrożenia, uruchamiając komendę **amqwdeployWMQService** i uruchamiając program nasłuchujący Axis 1.4 .

Wdrażanie usługi w usłudze .NET Framework 1 lub 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP
Wdróż usługę .NET Framework 1 lub 2 w produkcie WebSphere MQ transport dla protokołu SOAP. Utwórz katalog wdrażania, uruchom komendę **amqwdeployWMQService** i uruchom program nasłuchujący .NET.

Wdrażanie usługi na serwerze CICS Transaction Server w celu użycia produktu WebSphere Transport for SOAP

Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z obsługą usług Web Services produktu CICS Transaction Server 4.1 .

Konfigurowanie serwera WebSphere Application Server do korzystania z protokołu W3C SOAP over JMS
Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji Java EE . To zadanie jest krokiem 1. połączenia klienta usługi Web Service Axis2 i usługi Web Service wdrożonej na serwerze WebSphere Application Server przy użyciu protokołu W3C SOAP over JMS. Skonfiguruj zasoby WebSphere MQ i WebSphere Application Server , aby projektować i wdrażać usługę Web Service powiązaną z usługą W3C SOAP przez JMS jako transport.

Wdrażanie usługi w produkcie WebSphere ESB i punkcie końcowym usługi Process Server w celu użycia produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP nie jest obsługiwany bezpośrednio przez produkty WebSphere ESB i Process Server. Należy skonfigurować niestandardowy eksport.

Konfigurowanie serwera WebSphere Application Server do korzystania z protokołu W3C SOAP over JMS

Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji Java EE . To zadanie jest krokiem 1. połączenia klienta usługi Web Service Axis2 i usługi Web Service wdrożonej na serwerze WebSphere Application Server przy użyciu protokołu W3C SOAP over JMS. Skonfiguruj zasoby WebSphere MQ i WebSphere Application Server , aby projektować i wdrażać usługę Web Service powiązaną z usługą W3C SOAP przez JMS jako transport.

Zanim rozpoczniesz

Zadanie to wymaga serwera WebSphere Application Server v7.0.0.9 i WebSphere MQ v7.0.1.3.

O tym zadaniu

Zadanie składa się z dwóch kroków:

Procedura

1. “Konfigurowanie zasobów produktu WebSphere MQ” na stronie 1036
2. “Konfigurowanie zasobów serwera WebSphere Application Server” na stronie 1036

Co dalej

“Konfigurowanie zasobów produktu WebSphere MQ” na stronie 1036

Zadania pokrewne

Wdrażanie usługi w firmie Axis 1.4 w celu użycia w transporcie WebSphere dla protokołu SOAP przy użyciu produktu amqwdeployWMQService
Wdróż usługę Axis 1.4 w transporcie WebSphere MQ dla protokołu SOAP, tworząc katalog wdrożenia, uruchamiając komendę **amqwdeployWMQService** i uruchamiając program nasłuchujący Axis 1.4 .

Wdrażanie usługi w usłudze .NET Framework 1 lub 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP
Wdróż usługę .NET Framework 1 lub 2 w produkcie WebSphere MQ transport dla protokołu SOAP. Utwórz katalog wdrażania, uruchom komendę **amqwdeployWMQService** i uruchom program nasłuchujący .NET.

Wdrażanie usługi na serwerze CICS Transaction Server w celu użycia produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z obsługą usług Web Services produktu CICS Transaction Server 4.1 .

Wdrażanie usługi na serwerze WebSphere Application Server w celu używania produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z magistralą integracji usług na serwerze aplikacji WebSphere Application Server.

Wdrażanie usługi w produkcie WebSphere ESB i punkcie końcowym usługi Process Server w celu użycia produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP nie jest obsługiwany bezpośrednio przez produkty WebSphere ESB i Process Server. Należy skonfigurować niestandardowy eksport.

Konfigurowanie zasobów produktu WebSphere MQ

Zanim rozpoczniesz

W przypadku obsługi Axis2 wymagany jest produkt WebSphere MQ 7.0.1.3 lub nowszy.

O tym zadaniu

W przypadku uproszczenia zadanie zakłada, że program WebSphere MQ jest zainstalowany na tej samej stacji roboczej, co inne oprogramowanie, i korzysta z połączeń powiązań. Konfiguracje klienta WebSphere Application Server i Axis2 działają z połączeniami klientów. Aby uruchomić zadanie przy użyciu połączeń klienckich, należy sprawdzić, czy można umieścić i pobrać komunikaty do kolejek żądań i odpowiedzi zarówno z poziomu klienta Axis2 , jak i z komputerów serwera WebSphere Application Server.

Ponownie, dla uproszczenia, nie jest używana żadna konfiguracja zabezpieczeń. ID użytkownika ma pełne uprawnienia `mqm` .

Procedura

1. Utwórz domyślny menedżer kolejek `QM1`.

   Użyj programu WebSphere MQ Explorer, aby utworzyć `QM1` jako domyślny menedżer kolejek. Skonfiguruj go tak, aby uruchamiał się automatycznie i wybierz opcję tworzenia nasłuchiwania. Alternatywnie można użyć następujących komend:

   ```
   crtmqm -q -sa QM1
   strmqm
   echo define listener (LISTENER.TCP) trptype(tcp) ipaddr(localhost) port(1414)
              control(qmgr) replace | runmqsc
   echo start listener(LISTENER.TCP) | runmqsc
   ```

2. Zdefiniuj kolejkę żądań, `REQUESTAXIS`i kolejkę odpowiedzi `REPLYAXIS`.

   Użyj programu Explorer lub następujących komend:

   ```
   echo define ql(REQUESTAXIS) replace | runmqsc
   echo define ql(REPLYAXIS) replace | runmqsc
   ```

Co dalej

“Konfigurowanie zasobów serwera WebSphere Application Server” na stronie 1036

Konfigurowanie zasobów serwera WebSphere Application Server

Zanim rozpoczniesz

W przypadku obsługi protokołu W3C SOAP przez JMS wymagany jest serwer WebSphere Application Server v7. Ta konfiguracja została wykonana w produkcie WebSphere Application Server, wersja 7.0 Test Environment v7.0.0.9 , aktualizacja 1. Produkt WebSphere Application Server został dostarczony z produktem Rational Software Architect for WebSphere Software 7.5.4. Program Rational Software Architect został zaktualizowany do wersji v7.5.5.1, stosując najnowsze aktualizacje, które były dostępne.

W ramach procesu instalacji należy utworzyć profil dla serwera WebSphere Application Server. W zadaniu zabezpieczenia administracyjne nie są włączone. Domyślna nazwa profilu to `was70profile1`, a serwer to `server1`.

O tym zadaniu

Skonfiguruj serwer aplikacji WebSphere . Serwer można uruchomić z poziomu produktu Rational Application Developer, a następnie uruchomić Konsolę administracyjną z poziomu widoku Serwery lub uruchomić serwer przy użyciu pliku komend i administrować serwerem za pomocą przeglądarki. Zadanie korzysta z drugiej metody.

Pliki komend serwera znajdują się w folderze *Rational Installation Root*\SDP\runtimes\base_v7\profiles\was70profile1\bin. Pliki dziennika, które mają być zbadane, znajdują się w katalogu *Rational Installation Root*\SDP\runtimes\base_v7\profiles\was70profile1\logs\server1.

Zgodnie z konwencją wszystkie nazwy obiektów produktu WebSphere MQ są wielkimi literami, a wszystkie nazwy JNDI odwołujące się do obiektów WebSphere MQ są małe.

Procedura

1. Uruchom serwer.

   ```
   startServer server1
   ```

2. Uruchom przeglądarkę, otwórz konsolę administracyjną i zaloguj się.

   ```
   http://localhost:9061/ibm/console/unsecureLogon.jsp
   ```

   Wpisz dowolny łańcuch w polu ID użytkownika.

3. Tworzenie fabryki połączeń, qm1
   a) W nawigatorze otwórz opcję **Zasoby** > **JMS** > **Fabryki połączeń**.
   b) W oknie Connection Factory (Fabryki połączeń) wybierz zasięg **Node =*nazwa_węzła***, a następnie kliknij opcję **New**(Nowy).
   c) Wybierz opcję **Dostawca przesyłania komunikatów produktuWebSphere MQ** > **OK**.
   d) Podaj informacje o połączeniu menedżera kolejek z Tabela 143 na stronie 1037 > **Dalej**.

   Tabela 143. Informacje o połączeniu menedżera kolejek

Nazwa pola	**Wartość**
Nazwa	qm1
Nazwa JNDI	qm1

   e) Wybierz opcję **Wprowadź wszystkie wymagane informacje do tego kreatora** jako metodę połączenia > **Dalej**.
   f) Wpisz QM1 jako szczegóły połączenia kolejki > **Dalej**.
   g) Wprowadź szczegóły połączenia z programu Tabela 144 na stronie 1037 > **Dalej**.

   Tabela 144. Informacje o połączeniu

Nazwa pola	**Wartość**
Transport	Bindings, then client
Nazwa hosta	localhost
Port	1414
Kanał połączenia z serwerem	SYSTEM.DEF.SVRCONN

   h) **Testuj połączenie** > **Dalej** > **Zakończ** > **Zapisz**.

4. Utwórz kolejkę żądań JMS requestaxis.
   a) W oknie Navigatorotwórz opcje **Zasoby** > **JMS** > **Kolejki**.

b) W oknie Connection Factory (Fabryki połączeń) wybierz zasięg **Node =*nazwa_węzła***, a następnie kliknij opcję **New**(Nowy).

c) Wybierz opcję **Dostawca przesyłania komunikatów produktuWebSphere MQ** > **OK**.

d) Wprowadź szczegóły kolejki z menu Tabela 145 na stronie 1038 > **OK** > **Zapisz**.

Tabela 145. Szczegóły kolejki	
Nazwa pola	**Wartość**
Nazwa	`requestaxis`
Nazwa JNDI	`requestaxis`
Nazwa kolejki	`REQUESTAXIS`
Nazwa menedżera kolejek	`QM1`

5. Powtórz krok “4” na stronie 1037 , aby utworzyć kolejkę odpowiedzi JMS `replyaxis`.
6. Utwórz specyfikację aktywowania `qm1as`.

   Specyfikacja aktywowania wyzwala komponent bean sterowany komunikatami routera usług Web Service (MDB) po nadejściu komunikatu w kolejce żądań. Komponent MDB jest zdefiniowany w deskryptorze wdrażania usługi Web Service, która jest tworzona przez kreator usług Web Service produktu Rational Application Developer.

   a) W oknie Navigatorotwórz opcje **Zasoby** > **JMS** > **Specyfikacje aktywowania**.

   b) W oknie Connection Factory (Fabryki połączeń) wybierz zasięg **Node =*nazwa_węzła***, a następnie kliknij opcję **New**(Nowy).

   c) Wybierz opcję **Dostawca przesyłania komunikatów produktuWebSphere MQ** > **OK**.

   d) Wprowadź podstawowe atrybuty specyfikacji aktywowania z programu Tabela 146 na stronie 1038 > **Dalej**.

Tabela 146. Nazwa specyfikacji aktywowania	
Nazwa pola	**Wartość**
Nazwa	`qm1as`
Nazwa JNDI	`qm1as`

   e) Określ informacje dotyczące komponentu MDB z programu Tabela 147 na stronie 1038 > **Dalej**.

Tabela 147. Informacje o MDB	
Nazwa pola	**Wartość**
Nazwa JNDI miejsca docelowego	`requestaxis`
selektor komunikatów	*Puste pole*
Typ miejsca docelowego	`Queue`

   f) Wybierz opcję **Wprowadź wszystkie wymagane informacje do tego kreatora** jako metodę połączenia > **Dalej**.

   g) Wpisz `QM1` jako szczegóły połączenia kolejki > **Dalej**.

   h) Wprowadź szczegóły połączenia z programu Tabela 144 na stronie 1037 > **Dalej**.

Tabela 148. Informacje o połączeniu	
Nazwa pola	**Wartość**
Transport	`Bindings, then client`
Nazwa hosta	`localhost`

Tabela 148. Informacje o połączeniu (kontynuacja)	
Nazwa pola	**Wartość**
Port	1414
Kanał połączenia z serwerem	SYSTEM.DEF.SVRCONN

i) **Testuj połączenie** > **Dalej** > **Zakończ** > **Zapisz**.

7. Utwórz fabrykę połączeń kolejki (jms/WebServicesReplyQCF) dla kolejki odpowiedzi.

   Router usług Web Service używa fabryki połączeń kolejki w celu uzyskania dostępu do kolejki odpowiedzi. W deskryptorze wdrażania usługi Web Service fabryka połączeń kolejki ma domyślną nazwę JNDI serwera jms/WebServicesReplyQCF. Nazwę można zmienić w deskryptorze wdrażania. W tym zadaniu należy dodać nazwę domyślną do definicji zasobów JMS.

   a) W oknie Navigatorotwórz opcje **Zasoby** > **JMS** > **Fabryki połączeń kolejki**.

   b) W oknie Connection Factory (Fabryki połączeń) wybierz zasięg **Node =*nazwa_węzła***, a następnie kliknij opcję **New**(Nowy).

   c) Wybierz opcję **Dostawca przesyłania komunikatów produktuWebSphere MQ** > **OK**.

   d) Wprowadź podstawowe atrybuty fabryki połączeń kolejki z programu Tabela 149 na stronie 1039 > **Dalej**.

Tabela 149. Nazwa fabryki połączeń kolejki	
Nazwa pola	**Wartość**
Nazwa	WebServicesReplyQCF
Nazwa JNDI	jms/WebServicesReplyQCF

   e) Wybierz opcję **Wprowadź wszystkie wymagane informacje do tego kreatora** jako metodę połączenia > **Dalej**.

   f) Wpisz QM1 jako szczegóły połączenia kolejki > **Dalej**.

   g) Wprowadź szczegóły połączenia z programu Tabela 144 na stronie 1037 > **Dalej**.

Tabela 150. Informacje o połączeniu	
Nazwa pola	**Wartość**
Transport	Bindings, then client
Nazwa hosta	localhost
Port	1414
Kanał połączenia z serwerem	SYSTEM.DEF.SVRCONN

   h) **Testuj połączenie** > **Dalej** > **Zakończ** > **Zapisz**.

Co dalej

“Projektowanie usługi Web Service JAX-WS komponentu EJB dla usługi W3C SOAP over JMS” na stronie 996

Wdrażanie usługi w produkcie WebSphere ESB i punkcie końcowym usługi Process Server w celu użycia produktu WebSphere Transport for SOAP

Transport produktu WebSphere MQ dla protokołu SOAP nie jest obsługiwany bezpośrednio przez produkty WebSphere ESB i Process Server. Należy skonfigurować niestandardowy eksport.

O tym zadaniu

Produkt WebSphere Integration Developer udostępnia transformację danych SOAP, którą można powiązać z eksportem JMS produktu WebSphere MQ w celu utworzenia niestandardowego eksportu SOAP WebSphere MQ JMS.

Postępuj zgodnie z instrukcjami, aby utworzyć niestandardowy eksport, aby otrzymywać żądania SOAP za pośrednictwem transportu WebSphere MQ dla protokołu SOAP.

Procedura

1. Zapoznaj się z informacjami w sekcji Przegląd importów i eksportów oraz W jaki sposób nawiązać połączenie z produktem WebSphere MQ w dokumentacji produktu WebSphere Process Server for Multiplatforms V6.2 .
2. Wykonaj następujące czynności: Generowanie powiązania eksportu JMS produktu MQ w dokumentacji produktu IBM Business Process Manager, wersja 8.6 .

   Użyj powiązania danych SOAP opisanego w sekcji Transformowane transformacje formatu danych JMS , aby sformatować komunikat SOAP.

Zadania pokrewne

Wdrażanie usługi w firmie Axis 1.4 w celu użycia w transporcie WebSphere dla protokołu SOAP przy użyciu produktu amqwdeployWMQService
Wdróż usługę Axis 1.4 w transporcie WebSphere MQ dla protokołu SOAP, tworząc katalog wdrożenia, uruchamiając komendę **amqwdeployWMQService** i uruchamiając program nasłuchujący Axis 1.4 .

Wdrażanie usługi w usłudze .NET Framework 1 lub 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP
Wdróż usługę .NET Framework 1 lub 2 w produkcie WebSphere MQ transport dla protokołu SOAP. Utwórz katalog wdrażania, uruchom komendę **amqwdeployWMQService** i uruchom program nasłuchujący .NET.

Wdrażanie usługi na serwerze CICS Transaction Server w celu użycia produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z obsługą usług Web Services produktu CICS Transaction Server 4.1 .

Wdrażanie usługi na serwerze WebSphere Application Server w celu używania produktu WebSphere Transport for SOAP
Transport produktu WebSphere MQ dla protokołu SOAP jest zintegrowany z magistralą integracji usług na serwerze aplikacji WebSphere Application Server.

Konfigurowanie serwera WebSphere Application Server do korzystania z protokołu W3C SOAP over JMS
Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji Java EE . To zadanie jest krokiem 1. połączenia klienta usługi Web Service Axis2 i usługi Web Service wdrożonej na serwerze WebSphere Application Server przy użyciu protokołu W3C SOAP over JMS. Skonfiguruj zasoby WebSphere MQ i WebSphere Application Server , aby projektować i wdrażać usługę Web Service powiązaną z usługą W3C SOAP przez JMS jako transport.

Wdrażanie klientów usług Web Service w celu używania transportu produktu WebSphere MQ dla protokołu SOAP

Wdrożenie klienta usługi Web Service do jednego z wielu różnych środowisk klienta i nawiązanie połączenia z usługą przy użyciu protokołu transportowego WebSphere MQ dla protokołu SOAP.

Zanim rozpoczniesz

Utwórz usługę Web Service i wdróż ją w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP.

O tym zadaniu

Klient usługi Web Service można wdrożyć w taki sposób, aby był uruchamiany z transportem WebSphere MQ dla protokołu SOAP w wielu różnych środowiskach klienckich. Klient Java można wdrożyć w środowisku Axis 1.4 , korzystając tylko z oprogramowania zainstalowanego razem z produktem WebSphere MQ. W przypadku innych środowisk klienckich należy zainstalować dodatkowe oprogramowanie.

Nie jest ograniczony do uruchamiania transportu produktu WebSphere dla protokołu SOAP w środowiskach klienckich, dla których istnieją instrukcje wdrażania. Użyj instrukcji w celu wdrożenia klienta w jednym z obsługiwanych środowisk.

Uwaga: Niektóre zintegrowane środowiska oferują protokół SOAP korzystający z usługi JMS przy użyciu rekomendowanego powiązania SOAP W3C , a także transportu WebSphere MQ dla powiązania SOAP. Wersje produktu WebSphere MQ, do wersji 7.0.1.2włącznie, obsługują tylko transport produktu WebSphere MQ do powiązania SOAP. Począwszy od wersji 7.0.1.3 można wdrażać klienty Axis2 przy użyciu identyfikatora URI zgodnego z rekomendacją kandydata W3C dla protokołu SOAP korzystającego z usługi JMS. Informacje na ten temat zawiera kurs Tworzenie aplikacji usług Web Service JAX-WS SOAP/JMS z produktami WebSphere Application Server V7 i Rational Application Developer V7.5.

Wdrażanie klienta usługi Web Service do środowiska Axis 1.4 w celu korzystania z transportu IBM WebSphere MQ dla protokołu SOAP

Przygotuj katalog wdrażania i deskryptor wdrażania dla klienta. Podaj proxy klienta i klasę klienta, a następnie skonfiguruj zmienną CLASSPATH. Skonfiguruj kolejki i kanały IBM WebSphere MQ , uruchom usługę i przetestuj klient.

Zanim rozpoczniesz

Wskazówka: Wdróż usługę na potrzeby protokołu HTTP, opracuj i przetestuj klient dla protokołu HTTP, a następnie zmodyfikuj klient dla transportu IBM WebSphere MQ dla protokołu SOAP:

1. Dodaj wywołanie Register.extension() do klienta.
2. Zmień statyczny adres usługi Web Service w klasie wskaźnika proxy klienta tak, aby używany był identyfikator URI dla transportu IBM WebSphere MQ dla protokołu SOAP.

O tym zadaniu

Wdrożenie klienta Axis 1.4 w celu użycia transportu produktu IBM WebSphere MQ dla protokołu SOAP wymaga jednego dodatkowego kroku wdrożenia w porównaniu do klienta HTTP. Aby odwzorować transport produktu `jms:` na klasę nadawcy `com.ibm.mq.soap.transport.jms.WMQSender`, należy utworzyć deskryptor wdrażania klienta `client-config.wsdd`.

Jeśli komenda **amqwdeployWMQService** jest używana do generowania proxy klienta, można wdrożyć klienta przy użyciu katalogów generowanych przez komendę.

Procedura

1. Utwórz katalog *deployDir* , aby przechowywać pliki wdrażania klienta.
2. Otwórz okno komend w systemach Windows lub powłokę komend za pomocą systemu X Window System w systemach UNIX and Linux , w katalogu *deployDir*.
3. Uruchom komendę **amqwsetcp.cmd** , aby ustawić zmienną CLASSPATH .
4. Uruchom komendę **amqwclientconfig.cmd** , aby utworzyć deskryptor wdrażania klienta Axis 1.4 , `client-config.wsdd` w katalogu *deployDir*.
5. Upewnij się, że klasy w pakiecie klienta, klasy proxy klienta oraz biblioteki używane przez klienta znajdują się w zmiennej CLASSPATH.

   Produkt **amqwdeployWMQService** umieszcza proxy klienta .NET w produkcie `./generated/server/soap/client/remote/dotnetService` i proxy osi 1.4 w produkcie `./generated/server/soap/client/remote/`*`client package`*.

Przykład

The example shows the configuration and output, Rysunek 199 na stronie 1042, from an Axis 1.4 Java client. Klient, Rysunek 198 na stronie 1042, wywołuje usługę Web Service, która odbija się na jej parametrze wejściowym. Definicja usługi, Rysunek 197 na stronie 1042, przedstawia identyfikator URI pobranego z pliku WSDL usługi.

```
<wsdl:service name="QuoteSOAPImplService">
    wsdl:port binding="intf:org.example.www.QuoteSOAPImplBindingSoap"
                      name="org.example.www.QuoteSOAPImpl_Wmq">
       <wsdlsoap:address location="jms:/queue?destination=REQUESTAXIS
           &connectionFactory=(connectQueueManager(QM1)binding(server))
           &initialContextFactory=com.ibm.mq.jms.Nojndi
           &targetService=org.example.www.QuoteSOAPImpl.java
           &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE" />
    </wsdl:port>
</wsdl:service>
```

Rysunek 197. definicja usługi

```
package org.example.www;
import com.ibm.mq.soap.Register;
public class QuoteClient {
   public static void main(String[] args) {
      try {
         Register.extension();
         QuoteSOAPImplServiceLocator locator = new QuoteSOAPImplServiceLocator();
         System.out.println("Response = "
                  + locator.getOrgExampleWwwQuoteSOAPImpl_Wmq().getQuote("IBM"));
      } catch (Exception e) {
         System.out.println("Exception = " + e.getMessage());
      }
   }
}
```

Rysunek 198. Klient produktu Axis 1.4 Java

```
C:\IBM\ID\Test>dir /s /b
C:\IBM\ID\Test\client-config.wsdd
C:\IBM\ID\Test\org
C:\IBM\ID\Test\org\example
C:\IBM\ID\Test\org\example\www
C:\IBM\ID\Test\org\example\www\GetQuoteFaultMsg.class
C:\IBM\ID\Test\org\example\www\OrgExampleWwwQuoteSOAPImplBindingSoapStub.class
C:\IBM\ID\Test\org\example\www\QuoteClient.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImpl.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImplService.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImplServiceLocator.class

C:\IBM\ID\Test>amqwsetcp
C:\IBM\ID\Test>java org.example.www.QuoteClient.class
Response = IBM
```

Rysunek 199. Konfiguracja i wyjście klienta

Co dalej

1. Jeśli klient jest wdrażany jako klient IBM WebSphere MQ , należy skonfigurować kanał połączenia klienta i serwera.
2. Jeśli klient jest wdrażany do innego menedżera kolejek w usłudze, należy udostępnić kolejkę docelową dla klienta. Skonfiguruj kolejkę docelową w menedżerze kolejek usług jako kolejkę klastra lub w menedżerze kolejek klienta jako definicję kolejki zdalnej.

Zadania pokrewne

Wdrażanie klienta usługi Web Service w produkcie Axis2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP
Przygotuj katalog wdrażania i plik konfiguracyjny Axis2 dla klienta. Podaj proxy klienta i klasę klienta, a następnie ustaw zmienną CLASSPATH. Skonfiguruj kolejki i kanały produktu WebSphere MQ , uruchom usługę i przetestuj klient.

Wdrażanie na kliencie Axis2 przy użyciu protokołu W3C SOAP over JMS
Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji Java EE . Ta czynność to krok 4 połączenia klienta usługi Web Service Axis2 z usługą Web Service wdrożoną na serwerze WebSphere Application Server przy użyciu protokołu SOAP W3C SOAP over JMS. Zmodyfikuj adres URL w kliencie Axis2 opracowanym dla transportu WebSphere MQ dla protokołu SOAP w celu użycia rekomendacji kandydata W3C dla protokołu SOAP korzystającego z usługi JMS.

Wdrażanie klienta usługi Web Service w środowisku .NET Framework 1 i 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP
Przygotuj katalog wdrażania i deskryptor wdrażania dla klienta. Podaj proxy klienta i klasę klienta. Skonfiguruj kolejki i kanały produktu WebSphere MQ , uruchom usługę i przetestuj klient.

Wdrażanie klienta usługi Web Service w produkcie Axis2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP

Przygotuj katalog wdrażania i plik konfiguracyjny Axis2 dla klienta. Podaj proxy klienta i klasę klienta, a następnie ustaw zmienną CLASSPATH. Skonfiguruj kolejki i kanały produktu WebSphere MQ , uruchom usługę i przetestuj klient.

Zanim rozpoczniesz

Wskazówka: Wdróż usługę na serwerze HTTP. Utwórz i przetestuj klient dla protokołu HTTP, a następnie zmodyfikuj adres URL w taki sposób, aby odwoływał się do usługi przy użyciu transportu produktu WebSphere MQ dla protokołu SOAP.

W tym zadaniu przedstawiono sposób wdrożenia niezarządzanego klienta Axis2 w środowisku Java Standard Edition. Może być konieczne wdrożenie klienta Axis2 w kontenerze WWW. W produkcie “Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 1010opracowano klienta w kontenerze WWW i wdrożono go w produkcie WebSphere Application Server Community Edition. W ramach konfiguracji serwera włączono aspekt Axis2 i uwzględnił on aspekt w konfiguracji kontenera WWW. Aby skonfigurować kontenery WWW na innych serwerach aplikacji, należy zapoznać się z dokumentacją Axis2 , http://ws.apache.org/axis2/1_4_1/installationguide.html#servlet_containerlub dokumentacją dostarczonej z serwerem WWW.

Uwaga: Axis2 używa terminu, kontenera serwletów. Kontener serwletu jest taki sam, jak kontener WWW.

O tym zadaniu

Wdrażanie klienta Axis2 w celu korzystania z transportu WebSphere MQ dla protokołu SOAP jest podobne do wdrożenia klienta Axis2 w celu użycia protokołu HTTP. Aby udostępnić ścieżkę klasy do plików JAR produktu WebSphere MQ oraz zmodyfikować plik konfiguracyjny Axis2 , wymagane są dodatkowe kroki. Plik konfiguracyjny Axis2 wymaga dodatkowego wpisu dla usługi JMS. Ta pozycja odwołuje się do transportu WebSphere MQ dla pliku JAR SOAP, który implementuje interfejs JMS `transportSender`.

Axis2 udostępnia skrypt, produkt `axis2.bat` lub `axis2.sh`, który upraszcza wdrażanie klientów. Patrz przykłady w sekcji Rysunek 203 na stronie 1045 i Rysunek 204 na stronie 1046.

Uwaga:

1. `axis2.bat` ma błąd, który musi zostać poprawiony. Łańcuch `-Djava.ext.dirs="%AXIS2_HOME%\lib\"` musi zostać zmieniony na `-Djava.ext.dirs="%AXIS2_HOME%\lib\\"`.
2. In `axis2.bat` and `axis2.sh`, `-Djava.ext.dirs` is used as a quick way to reference all the Axis2 JAR files, instead of adding them separately to the classpath. Niestety takie podejście jest wadliwe i działa tylko z niektórymi JREs. Nie działa on w przypadku środowisk JRE firmy IBM .

Parametr maszyny JVM `-Djava.ext.dirs="%AXIS2_HOME%\lib\\"` powoduje udostępnienie plików JAR środowiska Axis dla maszyny JVM. Środowisko JVM próbuje utworzyć instancję niektórych plików JAR środowiska Axis i prowadzi do błędu, którego szczegóły zależą od maszyny JVM. Zwykle w stosie wywołań może zostać wyświetlony jeden z następujących wierszy:

```
org.apache.axiom.om.util.UUIDGenerator.getInitialUUID(UUIDGenerator.java:76)
```

lub wersji `org.apache.axis2.deployment.DeploymentException: java.security.NoSuchAlgorithmException: MD5 MessageDigest not available`

Poprawny sposób uruchamiania niezarządzanego klienta Axis2 polega na dodaniu plików JAR Axis2 do ścieżki klasy. Ścieżka klasy jest dostępna tylko dla aplikacji klienckiej, a nie dla maszyny JVM.

W tej procedurze opisano ogólne kroki uruchamiania niezarządzanego klienta Axis2 bez użycia skryptu axis2 . Przykłady w produkcie Rysunek 201 na stronie 1045 i Rysunek 202 na stronie 1045 to skrypty dla systemów Windows i Linux.

Procedura

1. Pobierz plik Axis2 1.4.1 z serwisu http://ws.apache.org/axis2/download/1_4_1/download.cgi i rozpakuj go do folderu, `Axis2-1.4.1`.
2. Zaktualizuj produkt `axis2.xml` w produkcie `Axis2-1.4.1\conf`.
   a) Zaktualizuj produkt `axis2.xml` w produkcie `Axis2-1.4.1\conf`. Dodaj transport produktu WebSphere MQ dla protokołu SOAP jako element `transportSender`:

   ```
   <transportSender name="jms"
   class="com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender"/>
   ```

   b) Jeśli jest to wymagane, zmień wielkość puli połączeń z wartości domyślnej 10.

   ```
   <transportSender name="jms"
   class="com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender">
   <parameter name="ResourcePoolCapacity">20</parameter>
   </transportSender>
   ```

   `ResourcePoolPojemność` określa liczbę pozycji punktów końcowych usługi, które są przechowywane w pamięci podręcznej. Wartość musi wynosić co najmniej 1. Jeśli liczba pozycji punktów końcowych usługi przekroczy wielkość pamięci podręcznej, pozycje zostaną usunięte, aby zrobić miejsce dla nowych pozycji. Zmienia się wielkość pozycji punktu końcowego. Ustaw liczbę, która jest wystarczająco duża, aby uniknąć thrashing pamięci podręcznej.

   Patrz krok 3 w sekcji “Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 1010.
3. Utwórz katalog *deployDir*. W tym katalogu skopiuj strukturę folderów zawierającą proxy klienta i klienta. Element *deployDir* jest odpowiednikiem folderu *`project`*`\bin` w projekcie Java platformy Eclipse .
4. Otwórz okno komend w systemie Windowslub powłokę komend za pomocą systemu X Window System w systemach UNIX and Linux , w katalogu *deployDir*.
5. Zaktualizuj ścieżkę klasy, tak aby uwzględniała bieżący katalog, pliki JAR Axis2 , `com.ibm.mqjms.jar` i `com.ibm.mq.axis2.jar`.

   Produkt `com.ibm.mqjms.jar` odwołuje się do wszystkich pozostałych wymaganych plików JAR produktu WebSphere MQ .
6. Aby uruchomić program kliencki, należy użyć komendy **Java** .

Przykłady

Cztery przykłady uruchomienia klienta Axis2 są wymienione w sekcji Rysunek 202 na stronie 1045 na Rysunek 204 na stronie 1046. Program Rysunek 200 na stronie 1045 wyświetla dane wyjściowe z uruchomionego klienta asynchronicznego wymienionego w sekcji Rysunek 183 na stronie 1015.

```
cd C:\IBM\ID\Workspaces\Axis2docs\StockQuoteAxis2PojoClient\bin>
runpojo soap/client/SQA2AsyncClient

HTTP Sync: 55.25
Callback constructor
Waiting for HTTP callback
Result in Callback 55.25
HTTP poll: 55.25
JMS: Sync: 55.25
Callback constructor
Waiting for JMS callback
Result in Callback 55.25
JMS poll: 55.25
Press any key to continue . . .
```

Rysunek 200. Dane wyjściowe z uruchomionego klienta SQA2AsyncClient

```
@echo off
set AXIS2_HOME=C:\OpenSource\axis2-1.4.1
set JAVA_HOME=C:\IBM\Java50
set WMQ_HOME=C:\IBM\MQ\java\lib

setlocal EnableDelayedExpansion
set CLASSPATH=
set AXIS2_CLASS_PATH=
FOR %%c in ("%AXIS2_HOME%\lib\*.jar") DO set AXIS2_CLASS_PATH=!AXIS2_CLASS_PATH!;%%c

"%JAVA_HOME%\bin\java" -Daxis2.repo="%AXIS2_HOME%\repository"
-Daxis2.xml="%AXIS2_HOME%\conf\axis2.xml" -cp
".;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar;%AXIS2_CLASS_PATH%" %1

pause
```

Rysunek 201. runpojo.bat: Windows, korzystając ze ścieżki klasy

```
export AXIS2_HOME=/home/OpenSource/axis2-1.4.1
export JAVA_HOME=/usr/lib/j2sdk1.5-ibm
# update classpath
AXIS2_CLASSPATH=""
for f in "$AXIS2_HOME"/lib/*.jar
do
  AXIS2_CLASSPATH="$AXIS2_CLASSPATH":$f
done
AXIS2_CLASSPATH="$AXIS2_HOME":"$JAVA_HOME/lib/tools.jar":"$AXIS2_CLASSPATH":"$CLASSPATH"
java -cp /home/alex/dev/sandbox/Soap/axis2:/opt/mqm/java/lib/com.ibm.mqjms.jar:
/opt/mqm/java/lib/com.ibm.mq.axis2.jar:$AXIS2_CLASSPATH
-Daxis2.xml=/home/alex/dev/sandbox/axis2-1.4.1/conf/axis2.xml %1
```

Rysunek 202. runpojo.sh: Linux, korzystając ze ścieżki klasy.

```
@echo off
set AXIS2_HOME=C:\OpenSource\axis2-1.4.1
set JAVA_HOME=C:\IBM\Java50
set WMQ_HOME=C:\IBM\MQ\java\lib

%AXIS2_HOME%\bin\axis2 -cp .;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar; %1
pause
```

Rysunek 203. runaxis2.bat: Windows, za pomocą pliku axis2.bat

Uwaga

```
export AXIS2_HOME=/home/OpenSource/axis2-1.4.1
export JAVA_HOME=/usr/lib/j2sdk1.5-ibm

%AXIS2_HOME%\bin\axis2 -cp .;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar; %1
```

Rysunek 204. runaxis2.sh: Linux, używając axis2.sh

Uwaga

Zadania pokrewne

Wdrażanie klienta usługi Web Service do środowiska Axis 1.4 w celu korzystania z transportu IBM WebSphere MQ dla protokołu SOAP

Przygotuj katalog wdrażania i deskryptor wdrażania dla klienta. Podaj proxy klienta i klasę klienta, a następnie skonfiguruj zmienną CLASSPATH. Skonfiguruj kolejki i kanały IBM WebSphere MQ , uruchom usługę i przetestuj klient.

Wdrażanie na kliencie Axis2 przy użyciu protokołu W3C SOAP over JMS

Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji Java EE . Ta czynność to krok 4 połączenia klienta usługi Web Service Axis2 z usługą Web Service wdrożoną na serwerze WebSphere Application Server przy użyciu protokołu SOAP W3C SOAP over JMS. Zmodyfikuj adres URL w kliencie Axis2 opracowanym dla transportu WebSphere MQ dla protokołu SOAP w celu użycia rekomendacji kandydata W3C dla protokołu SOAP korzystającego z usługi JMS.

Wdrażanie klienta usługi Web Service w środowisku .NET Framework 1 i 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP

Przygotuj katalog wdrażania i deskryptor wdrażania dla klienta. Podaj proxy klienta i klasę klienta. Skonfiguruj kolejki i kanały produktu WebSphere MQ , uruchom usługę i przetestuj klient.

Wdrażanie na kliencie Axis2 przy użyciu protokołu W3C SOAP over JMS

Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji Java EE . Ta czynność to krok 4 połączenia klienta usługi Web Service Axis2 z usługą Web Service wdrożoną na serwerze WebSphere Application Server przy użyciu protokołu SOAP W3C SOAP over JMS. Zmodyfikuj adres URL w kliencie Axis2 opracowanym dla transportu WebSphere MQ dla protokołu SOAP w celu użycia rekomendacji kandydata W3C dla protokołu SOAP korzystającego z usługi JMS.

Zanim rozpoczniesz

Najpierw należy wykonać zadanie “Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 1010 , aby wywołać produkt **SimpleJavaListener** przy użyciu klienta Axis2 i protokołu transportowego WebSphere MQ dla protokołu SOAP.

W poprzednich zadaniach należy również utworzyć usługę Web Service i skonfigurować produkt WebSphere MQ oraz serwer WebSphere Application Server :

1. “Konfigurowanie zasobów produktu WebSphere MQ” na stronie 1036.
2. “Konfigurowanie zasobów serwera WebSphere Application Server” na stronie 1036.
3. “Projektowanie usługi Web Service JAX-WS komponentu EJB dla usługi W3C SOAP over JMS” na stronie 996.

W ramach zadania klient działa w systemie Eclipse Galileo. Klient może uruchomić klienta z wiersza komend, modyfikując plik `Axis2.bat` dostarczony z Axis2.

O tym zadaniu

Jedyną zmianą, którą należy wprowadzić, aby istniejący klient statyczny Axis2 `StockQuoteAxis` wywoła usługę Axis StockQuoteudostępnianą przez serwer aplikacji WebSphere , jest zmiana adresu URL przekazanego do klienta. Ponieważ plik WSDL nie został zmieniony, można użyć tych samych klas proxy w pakiecie `soap.server` .

Istnieją dwa podejścia do definiowania adresu URL do przekazania do klienta. Można użyć tego samego adresu URL, co w wygenerowanym `StockQuoteAxis.wsdl`. Aby uzyskać dostęp do katalogu JNDI serwera WebSphere Application Server, należy dodać parametry `jndiInitialContextFactory` i `jndiURL` . Innym podejściem jest zmiana adresu URL i nadanie klientowi bezpośredniego dostępu do kolejek produktu `REQUESTAXIS` i `REPLYAXIS` w produkcie `QM1`bez użycia wyszukiwania JNDI.

Parametry połączenia zdefiniowane w adresie URL przekazytym do klienta Axis2 są używane do nawiązywania połączenia z menedżerem kolejek produktu WebSphere MQ i kolejkami wymaganymi do wysyłania i odbierania komunikatów SOAP. Parametry połączenia przekazane do klienta Axis2 nie muszą być używane przez usługę. Można użyć możliwości rozproszonego kolejkowania produktu WebSphere MQ w celu oddzielającego klienta i usługi od używania tego samego menedżera kolejek lub tego samego serwera nazw.

Procedura

1. Zapisz adres URL z wygenerowanej bazy danych `StockQuoteAxis.wsdl` i zamknij produkt Rational Application Developer, aby zapisać go w pamięci.

   Jeśli konfiguracja serwera nie została zmieniona, zamknięcie produktu Rational Application Developer spowoduje zatrzymanie serwera aplikacji. W takim przypadku uruchom serwer za pomocą komendy:

   ```
   startserver server1
   ```

2. Otwórz program Eclipse Galileo w obszarze roboczym, korzystając z projektu klienta Axis2 .
3. Otwórz program `SQA2StaticClient.java`.

   Patrz SQA2StaticClient.java.
4. Wywołaj usługę, korzystając z wariantu `queue` identyfikatora URI.
   a) Zmodyfikuj adres URL.

   Nowy identyfikator URI to:

   ```
   jms:queue:REQUESTAXIS
             ?replyToName=REPLYAXIS
             &connectionFactory=connectQueueManager(QM1)Bind(Server)
             &targetService=StockQuoteAxis;
   ```

   Porównaj tę wartość z adresem URL z produktu `StockQuoteAxis.wsdl`:

   ```
   jms:jndi:requestaxis
            ?jndiConnectionFactoryName=qm1
            &targetService=StockQuoteAxis
   ```

   Rysunek 205. Adres URL z `StockQuoteAxis.wsdl`

   - Produkt `REQUESTAXIS` jest teraz pisany wielkimi literami, ponieważ jest to nazwa kolejki, a nie nazwa JNDI.
   - Połączenie z produktem `QM1` jest proste.
   - Identyfikator URI nie zawiera nazwy odpowiedzi do miejsca docelowego. Klient musi zdefiniować kolejkę, na której oczekuje odpowiedzi.

   b) Uruchom program `SQA2StaticClient.java` przy użyciu tej samej opcji **Uruchom jako ...** konfiguracji, jak to zostało wykonane w zadaniu “Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 1010.
5. Wywołaj usługę, korzystając z wariantu identyfikatora URI `jndi` , używając serwera WebSphere Application Server jako serwera nazw.
   a) Użyj adresu URL z `StockQuoteAxis.wsdl`, Rysunek 205 na stronie 1047, podając brakujące parametry, aby użyć usługi nazw w produkcie WebSphere Application Server.

   Brakujące parametry i wartości, które należy podać, to:

Tabela 151. Dodatkowe parametry JNDI

Parametr	Wartość użyta w tym przykładzie	Opis
&jndiURL	iiop://localhost:2810 lub wersji corbaname:iiop:localhost:2810	Identyfikator URI dostawcy nazewnictwa. W przypadku serwera WebSphere Application Server wartością domyślną jest 2809. Jest on również znany jako numer portu konektora RMI, a także port programu startowego. Wartość jest wyświetlana w SystemOut.log `00000000 NameServerImp A NMSV0018I: Name server available on bootstrap port 2810`
&jndiInitialContextFactory	com.ibm.websphere.naming.WsnInitialContextFactory	Nazwa początkowej fabryki kontekstów używanej przez serwer WebSphere Application Server.
&replyToName	replyaxis	Nazwa JNDI kolejki produktu REPLYAXIS .

```
jms:jndi:requestaxis?
         &jndiURL=iiop//localhost:2810
         &jndiConnectionFactoryName=qm1
         &jndiInitialContextFactory=com.ibm.websphere.naming.WsnInitialContextFactory
         &targetService=StockQuoteAxis
         &replyToName=replyaxis;
```

b) Dodaj pliki JAR wymagane przez wyszukiwanie JNDI.

W tej konfiguracji do ścieżki budowania dodano następujące pliki JAR w celu uruchomienia zadania przy użyciu wariantu jndi adresu URL usługi JMS:

- com.ibm.jaxws.thinclient_7.0.0.jar z *Rational install directory*\SDP\runtimes\base_v7\runtimes.
- com.ibm.ws.runtime.jar z wersji *Rational install directory*\SDP\runtimes\base_v7\plugins

W przypadku innego dostawcy JNDI wymagane są różne pliki JAR.

Pozostałe pliki JAR w ścieżce budowania są następujące:

i) Wszystkie pliki JAR w produkcie *WebSphere MQ Install directory*\java\lib.

ii) Wszystkie pliki JAR w produkcie *Axis2-1.5.1*\lib.

iii) Środowisko JRE Java 6.0 .

c) Uruchom program SQA2StaticClient.java przy użyciu tej samej opcji **Uruchom jako ...** konfiguracji, jak to zostało wykonane w zadaniu "Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse" na stronie 1010.

Wyniki

W obu przypadkach odpowiedź z usługi jest wyświetlana w widoku konsoli klienta.

Zadania pokrewne

Wdrażanie klienta usługi Web Service do środowiska Axis 1.4 w celu korzystania z transportu IBM WebSphere MQ dla protokołu SOAP

Przygotuj katalog wdrażania i deskryptor wdrażania dla klienta. Podaj proxy klienta i klasę klienta, a następnie skonfiguruj zmienną CLASSPATH. Skonfiguruj kolejki i kanały IBM WebSphere MQ , uruchom usługę i przetestuj klient.

Wdrażanie klienta usługi Web Service w produkcie Axis2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP
Przygotuj katalog wdrażania i plik konfiguracyjny Axis2 dla klienta. Podaj proxy klienta i klasę klienta, a następnie ustaw zmienną CLASSPATH. Skonfiguruj kolejki i kanały produktu WebSphere MQ , uruchom usługę i przetestuj klient.

Wdrażanie klienta usługi Web Service w środowisku .NET Framework 1 i 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP
Przygotuj katalog wdrażania i deskryptor wdrażania dla klienta. Podaj proxy klienta i klasę klienta. Skonfiguruj kolejki i kanały produktu WebSphere MQ , uruchom usługę i przetestuj klient.

Wdrażanie klienta usługi Web Service w środowisku .NET Framework 1 i 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP

Przygotuj katalog wdrażania i deskryptor wdrażania dla klienta. Podaj proxy klienta i klasę klienta. Skonfiguruj kolejki i kanały produktu WebSphere MQ , uruchom usługę i przetestuj klient.

Zanim rozpoczniesz

Wskazówka: Opracowanie i przetestowanie usługi oraz klienta przy użyciu Visual Studio. Następnie należy zmodyfikować klienta dla transportu WebSphere MQ dla protokołu SOAP.

1. Jeśli usługa jest wdrażana przy użyciu środowiska .NET Framework 1 lub 2, należy zbudować usługę jako bibliotekę (`.dll`). Wdrażaj przy użyciu transportu WebSphere MQ dla protokołu SOAP.
2. Dodaj wywołanie Register.Extension() do klienta.
3. Dodaj odwołanie do pliku amqsoap.dll, który znajduje się w katalogu *MQ_Install*`\bin`.
4. Zmień statyczną właściwość `Url` w konstruktorze klasy proxy klienta na identyfikator URI produktu `jms:/` , w przypadku transportu WebSphere MQ dla protokołu SOAP.

O tym zadaniu

Wdrożenie klienta usługi Web Service dla środowiska .NET Framework 1 lub 2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP wymaga dodatkowego kroku wdrożenia. Konieczne jest zarejestrowanie produktu `amqsoap.dll` w środowisku .NET Framework. Produkt `amqsoap.dll` jest rejestrowany automatycznie jako część procesu instalowania transportu WebSphere MQ dla protokołu SOAP, ale może być konieczne jego ponowne zarejestrowanie.

Jeśli komenda **`amqwdeployWMQService`** jest używana do generowania proxy klienta, można wdrożyć klienta przy użyciu katalogów generowanych przez komendę.

Procedura

1. Utwórz katalog *deployDir* , aby przechowywać pliki wdrażania klienta.
2. Otwórz okno komend w katalogu *deployDir*.
3. Uruchom program **`amqwsetcp`** , aby ustawić zmienną CLASSPATH , jeśli usługa ma być uruchamiana na osi 1.4.
4. Jeśli to konieczne, uruchom program **`amqwRegisterDotNet`** , aby zarejestrować produkt `amqsoap.dll` w środowisku .NET Framework.

Przykład

W przykładzie przedstawiono konfigurację i dane wyjściowe Rysunek 208 na stronie 1050z klienta środowiska .NET Framework V2 . Klient, Rysunek 207 na stronie 1050, wywołuje usługę Web Service, która odbija się na jej parametrze wejściowym. Statyczna definicja adresu URL, Rysunek 206 na stronie 1050, przedstawia konstruktor dla proxy klienta.

```
public Quote() {
    this.Url = "jms:/queue?destination=REQUESTDOTNET
               &connectionFactory=(connectQueueManager(QM1)binding(server))
               &initialContextFactory=com.ibm.mq.jms.Nojndi
               &targetService=Quote.asmx
               &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE";
}
```

Rysunek 206. Konstruktor proxy statycznego klienta

```
using System;
namespace QuoteClientProgram {
    class QuoteMain {
        static void Main(string[] args) {
            try {
                IBM.WMQSOAP.Register.Extension();
                Quote q = new Quote();
                Console.WriteLine("Response is: " + q.getQuote("ibm"));
            } catch (Exception e) {
                Console.WriteLine("Exception is: " + e);
            }
        }
    }
}
```

Rysunek 207. Program kliencki

```
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram>dir /s /b
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram\QuoteClientProgram.exe
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram>quoteclientprogram
Response is: IBM
```

Rysunek 208. Konfiguracja i wyjście

Co dalej

1. Jeśli klient jest wdrażany jako klient MQI produktu WebSphere MQ , należy skonfigurować kanał połączenia klienta i serwera.
2. Jeśli klient jest wdrażany do innego menedżera kolejek w usłudze, należy udostępnić kolejkę docelową dla klienta. Skonfiguruj kolejkę docelową w menedżerze kolejek usług jako kolejkę klastra lub w menedżerze kolejek klienta jako definicję kolejki zdalnej.

Zadania pokrewne

Wdrażanie klienta usługi Web Service do środowiska Axis 1.4 w celu korzystania z transportu IBM WebSphere MQ dla protokołu SOAP
Przygotuj katalog wdrażania i deskryptor wdrażania dla klienta. Podaj proxy klienta i klasę klienta, a następnie skonfiguruj zmienną CLASSPATH. Skonfiguruj kolejki i kanały IBM WebSphere MQ , uruchom usługę i przetestuj klient.

Wdrażanie klienta usługi Web Service w produkcie Axis2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP
Przygotuj katalog wdrażania i plik konfiguracyjny Axis2 dla klienta. Podaj proxy klienta i klasę klienta, a następnie ustaw zmienną CLASSPATH. Skonfiguruj kolejki i kanały produktu WebSphere MQ , uruchom usługę i przetestuj klient.

Wdrażanie na kliencie Axis2 przy użyciu protokołu W3C SOAP over JMS
Usługa Web Service powiązana z kandydacką rekomendacją W3C dla protokołu SOAP over JMS musi być uruchamiana w kontenerze EJB serwera aplikacji Java EE . Ta czynność to krok 4 połączenia klienta usługi Web Service Axis2 z usługą Web Service wdrożoną na serwerze WebSphere Application Server przy użyciu protokołu SOAP W3C SOAP over JMS. Zmodyfikuj adres URL w kliencie Axis2 opracowanym dla

transportu WebSphere MQ dla protokołu SOAP w celu użycia rekomendacji kandydata W3C dla protokołu SOAP korzystającego z usługi JMS.

Połączenie klienta Axis2 z usługą JAX-WS przy użyciu protokołu W3C SOAP over JMS oraz serwera WebSphere Application Server.

Po zakończeniu tego zadania zostanie wywołana usługa Web Service JAX-WS działająca na serwerze WebSphere Application Server z poziomu klienta Axis2 . Klient Axis2 i serwer WebSphere Application Server korzystają z rekomendacji kandydata W3C dla protokołu SOAP korzystającego z protokołu JMS działającego w produkcie WebSphere MQ. Użyj programów Eclipse Galileo i Rational Application Developer, aby zbudować odpowiednio klienta usługi Web Service i usługę Web Service.

Zanim rozpoczniesz

Zadanie wymaga wersji 7 środowiska Rational Software Development Environment oraz serwera WebSphere Application Server. Zadanie zostało utworzone przy użyciu produktu Rational Application Developer spakowanego z produktem Rational Software Architect for WebSphere Software v7.5.5.1, and WebSphere Application Server , wersja 7.0 Test Environment v7.0.0.9 Update 1. Wymagany jest również produkt WebSphere MQ v7.0.1.3.

Zadanie jest oparte na dwóch innych zadaniach: “Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 988i “Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 1010. Aby wykonać te zadania, w środowisku programistycznym jest już zainstalowane środowisko Eclipse Galileo, WASCE, wtyczka Eclipse dla WASCE i Axis2 1.4.1 . Nie jest wymagane WASCE dla tego zadania.

Niektóre z etapów są złożone. W tych krokach przyjęto założenie, że niektóre aplikacje usług Web Service dla serwera WebSphere Application Server są dostępne przy użyciu produktu Rational Application Developer. Wymagania dotyczące procesora i pamięci związane z zadaniem są duże. Zadanie zostało wykonane na maszynie wirtualnej VMWare Windows XP SP3 przydzieloną 1.8GB pamięci.

Przed uruchomieniem zadania zainstaluj całe oprogramowanie. Oprogramowanie zajmuje około dzień do pobrania i dzień instalacji, w zależności od przepustowości. Zadanie trwa co najmniej pół dnia.

O tym zadaniu

Scenariusz dla tego zadania polega na tym, że opracowano usługę Web Service notowań akcji `StockQuoteAxis`przy użyciu narzędzia open source, Eclipse Galileo. Produkt `StockQuoteAxis` jest wdrażany przy użyciu protokołu SOAP korzystający z protokołu HTTP działającego na otwartym serwerze źródłowym, WASCE.

Użytkownik chce powiązać usługi Web Service wdrażane z transportem przesyłania komunikatów opartym na standardach, takim jak SOAP-JMS lub z niezawodnym przesyłaniem komunikatów usług Web Service, a także z SOAP-za pośrednictwem protokołu HTTP. Użytkownik chce, aby klient i usługa używały interfejsów opartych na standardach. Z tego powodu, mimo że przyszły zespół projektowy wdrożyliśmy rozwiązanie korzystające z transportu WebSphere MQ dla protokołu SOAP, nie zostało ono wprowadzone do środowiska produkcyjnego.

Klient Axis2 usunował problem, który klient SOAP dla transportu WebSphere MQ dla protokołu SOAP wymagał zmiany z klienta HTTP. Problem nadal utrzymywał, że usługa połączona z transportem IBM WebSphere MQ dla protokołu SOAP jest obsługiwana przez specjalny program nasłuchujący udostępniany przez produkt WebSphere MQ: `SimpleJavaListener`.

W przypadku standardu SOAP W3C SOAP over JMS w statusie rekomendacji kandydata, niektórzy dostawcy zapewniają obsługę protokołu SOAP W3C SOAP przez JMS. Obsługa umożliwia wdrożenie usługi Web Service na serwerze aplikacji i nawiąże połączenie z tą samą usługą przy użyciu różnych protokołów połączeń. Wsparcie udostępniane przez serwer WebSphere Application Server v7 usuwa problem z tym, że usługa Web Service jest udostępniana osobno, aby można było używać transportu SOAP opartego na komunikatach. Korzystanie z interfejsu transportowego komunikatów opartego na standardach, JMS, oznacza, że można tworzyć rozwiązania przy użyciu narzędzi różnych dostawców.

Użytkownik ma nadzieję, że narzędzia usług Web Service w środowisku Eclipse będą zawierać powiązania SOAP-JMS w przyszłości.

Większość kroków jest wykonywanych przy użyciu środowiska Eclipselub narzędzi zarządzania dostarczonych z produktami WebSphere . Kroki zostały opisane w przypadku środowiska Windows . Po wprowadzeniu niewielkich zmian w niektórych komendach można wykonać kroki na innych platformach.

Wyświetlane są wstępne kroki tworzenia usługi Web Service HTTP i łączenia się z nim za pomocą komendy Axis2 . Klient i plik WSDL z tych kroków są używane do tworzenia rozwiązania.

Procedura

1. Nawiąże połączenie z usługą Web Service produktu StockQuoteAxis przy użyciu klienta Axis2 i transportu IBM WebSphere MQ dla protokołu SOAP.
   a) “Projektowanie usługi JAX-RPC dla transportu produktu WebSphere MQ dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 988
   b) “Tworzenie klienta JAX-WS dla transportu WebSphere dla protokołu SOAP przy użyciu środowiska Eclipse” na stronie 1010
   c) “Wdrażanie klienta usługi Web Service w produkcie Axis2 w celu użycia transportu produktu WebSphere MQ dla protokołu SOAP” na stronie 1043
2. Połącz się z usługą Web Service środowiska Axis StockQuoteprzy użyciu klienta Axis2 i rekomendacji kandydata W3C dla protokołu SOAP korzystającego z usługi JMS.
   a) “Konfigurowanie zasobów produktu WebSphere MQ” na stronie 1036
   b) “Konfigurowanie zasobów serwera WebSphere Application Server” na stronie 1036
   c) “Projektowanie usługi Web Service JAX-WS komponentu EJB dla usługi W3C SOAP over JMS” na stronie 996
   d) “Wdrażanie na kliencie Axis2 przy użyciu protokołu W3C SOAP over JMS” na stronie 1046

Most produktu WebSphere MQ dla protokołu HTTP

Za pomocą mostu WebSphere MQ dla protokołu HTTP aplikacje klienckie mogą wymieniać komunikaty z produktem WebSphere MQ bez konieczności instalowania klienta MQI produktu WebSphere MQ . Produkt WebSphere MQ można wywołać z dowolnej platformy lub języka z możliwościami HTTP.

Wprowadzenie do mostu WebSphere MQ dla protokołu HTTP

Most produktu WebSphere MQ dla protokołu HTTP jest aplikacją WWW Java, Enterprise Environment (JEE). Klienty HTTP mogą wysyłać do niego żądania **POST**, **GET**i **DELETE** w celu umieszczania, przeglądania i usuwania komunikatów z kolejek produktu WebSphere MQ . Most WebSphere MQ dla protokołu HTTP nie jest odpowiedni do użycia z komunikatami, jeśli wymagana jest zapewniona dostawa.

Korzyści

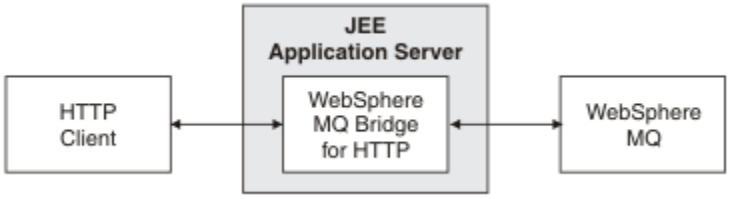
Za pomocą mostu WebSphere MQ dla protokołu HTTP można wysyłać i odbierać komunikaty produktu WebSphere MQ przy użyciu protokołu HTTP z wielu różnych środowisk:

- Środowiska, które obsługują protokół HTTP, ale nie są obsługiwane przez produkt WebSphere MQ.
- Środowiska, w których nie ma wystarczającej ilości miejsca w pamięci masowej, aby zainstalować klienta MQI produktu WebSphere MQ .
- Środowiska, które są zbyt liczne, aby zainstalować klienta MQI produktu WebSphere MQ w każdym systemie, który wymaga dostępu do produktu WebSphere MQ.
- Aplikacje z interfejsem WWW, z których mają być wysyłane lub odbierane komunikaty bez konieczności kodowania własnego mostu do produktu WebSphere MQ.
- Aplikacje z interfejsem WWW, które mają zostać ulepszone, przy użyciu technik asynchronicznych, takich jak AJAX. Most produktu WebSphere MQ dla protokołu HTTP powoduje, że kolejki i tematy

produktu WebSphere MQ są dostępne przy użyciu usługi REST (Representation State Transfer) przy użyciu protokołu HTTP.

Obsługa protokołu HTTP może być używana zarówno z topologiami przesyłania komunikatów typu punkt z punktem, jak i publikowania/subskrypcji.

Jak działa obsługa protokołu HTTP?



Rysunek 209. Most produktu WebSphere MQ dla protokołu HTTP

Most produktu WebSphere MQ dla aplikacji WWW HTTP odbiera żądania HTTP od jednego lub większej liczby klientów. Współdziała z produktem WebSphere MQ w ich imieniu i zwraca do nich odpowiedzi HTTP.

Most WebSphere MQ dla protokołu HTTP to serwlet JEE, który jest połączony z produktem WebSphere MQ przy użyciu adaptera zasobów. Serwlet HTTP obsługuje trzy różne typy żądań HTTP: **POST**, **GET**i **DELETE**.

Tabela 152. Most produktu WebSphere MQ dla komend HTTP

Żądanie HTTP	**Wynik**
POST	Umieszcza komunikat w kolejce lub temacie.
GET	Przegląda pierwszy komunikat w kolejce. W wierszu z protokołem HTTP produkt **GET** nie usuwa komunikatu z kolejki. Nie należy używać produktu **GET** z przesyłaniem komunikatów w trybie publikowania/subskrypcji.
USUŃ	Pobiera i usuwa komunikat z kolejki lub tematu.

Przykład metody HTTP POST

Metoda HTTP **POST** umożliwia umieszczenie komunikatu w kolejce lub jego opublikowanie w temacie. Przykład komendy **HTTPPOST** Java to przykład żądania HTTP **POST** dotyczącego umieszczenia komunikatu w kolejce. Zamiast używać języka Java, można utworzyć żądanie HTTP **POST** przy użyciu formularza przeglądarki lub przybornika AJAX.

Rysunek 210 na stronie 1053 przedstawia żądanie HTTP w celu umieszczenia komunikatu w kolejce o nazwie `myQueue`. To żądanie zawiera nagłówek HTTP `x-msg-correlId` , aby ustawić identyfikator korelacji komunikatu WebSphere MQ .

```
POST /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
Content-Type: text/plain
x-msg-correlID: 1234567890
Content-Length: 50

Here is my message body that is posted on the queue.
```

*Rysunek 210. Przykład żądania HTTP **POST** w kolejce*

Rysunek 211 na stronie 1054 wyświetla odpowiedź wysłanej z powrotem do klienta. Brak treści odpowiedzi.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 0
```

Rysunek 211. Przykład odpowiedzi HTTP POST

Przykład metody HTTP DELETE

Metoda HTTP **DELETE** umożliwia pobranie komunikatu z kolejki i jego usunięcie lub pobranie i usunięcie publikacji. Przykład komendy **HTTPDELETE** języka Java to przykład, w którym wykonywany jest odczyt komunikatu z kolejki przez żądanie HTTP **DELETE**. Zamiast używać języka Java, można utworzyć żądanie HTTP **DELETE** przy użyciu formularza przeglądarki lub przybornika AJAX.

Rysunek 212 na stronie 1054 jest żądaniem HTTP do usunięcia następnego komunikatu w kolejce o nazwie `myQueue`. W odpowiedzi treść komunikatu jest zwracana do klienta. W terminach WebSphere MQ HTTP **DELETE** jest destrukcyjnym dostaniem.

Żądanie zawiera nagłówek żądania HTTP `x-msg-wait`, który instruuje most WebSphere MQ dla protokołu HTTP, jak długo czekać na pojaw się komunikatu w kolejce. Żądanie zawiera również nagłówek żądania `x-msg-require-headers`, który wskazuje, że klient ma otrzymać w odpowiedzi identyfikator korelacji komunikatu.

```
DELETE /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Rysunek 212. Przykład żądania HTTP **DELETE***

Rysunek 213 na stronie 1054jest odpowiedzią zwracaną do klienta. Identyfikator korelacji jest zwracany do klienta zgodnie z żądaniem w nagłówku `x-msg-require-headers` żądania.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that is retrieved from the queue.
```

*Rysunek 213. Przykład odpowiedzi HTTP **DELETE***

Przykład metody HTTP GET

Metoda HTTP **GET** pobiera komunikat z kolejki. Komunikat pozostaje w kolejce. W terminach WebSphere MQ HTTP **GET** jest żądaniem przeglądania. Żądanie metody HTTP **GET** można utworzyć przy użyciu klienta Java, formularza przeglądarki lub przybornika AJAX.

Rysunek 214 na stronie 1055 jest żądaniem HTTP do przeglądania następnego komunikatu w kolejce o nazwie `myQueue`.

Żądanie zawiera nagłówek żądania HTTP `x-msg-wait`, który instruuje most WebSphere MQ dla protokołu HTTP, jak długo czekać na pojaw się komunikatu w kolejce. Żądanie zawiera również nagłówek żądania `x-msg-require-headers`, który wskazuje, że klient ma otrzymać w odpowiedzi identyfikator korelacji komunikatu.

```
GET /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Rysunek 214. Przykład żądania HTTP **GET***

Rysunek 215 na stronie 1055 jest odpowiedzią zwracaną do klienta. Identyfikator korelacji jest zwracany do klienta zgodnie z żądaniem w nagłówku `x-msg-require-headers` żądania.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that appears on the queue.
```

*Rysunek 215. Przykład odpowiedzi HTTP **GET***

Instalowanie, konfigurowanie i weryfikowanie mostu WebSphere MQ dla protokołu HTTP

Uzyskaj most WebSphere MQ dla protokołu HTTP, instalując opcję "Java Messaging and Web Services" (Przesyłanie komunikatów języka Java i usługi Web Service) z klienta MQI produktu WebSphere MQ lub z materiałów instalacyjnych serwera. Wdróż most WebSphere MQ dla protokołu HTTP na odpowiednim serwerze aplikacji.

Zanim rozpoczniesz

Sprawdź wstępnie wymagane produkty pod adresem Wymagania systemowe produktu IBM WebSphere MQ. Proces instalacji nie sprawdza obecności i dostępności wstępnie wymaganego oprogramowania dla uruchomionego mostu WebSphere MQ dla protokołu HTTP. Należy sprawdzić, czy zostały zainstalowane wymagania wstępne.

Most WebSphere MQ dla protokołu HTTP działa na dowolnym serwerze aplikacji zgodnym z Java EE 1.4 , instalując adapter zasobów produktu WebSphere MQ . Możliwe jest również uruchomienie mostu WebSphere MQ dla protokołu HTTP na serwerze WebSphere Application Server w wersji wcześniejszej niż 6.0.2.1. Użyj portu nasłuchiwania komunikatów serwera aplikacji WebSphere Application Server (MLP), aby zintegrować produkt WebSphere MQ z dostawcą JMS.

Obsługa mostu WebSphere MQ dla protokołu HTTP jest udostępniana tylko dla następujących serwerów aplikacji:

- WebSphere Application Server 6.0.2.1 i nowsze.
- WebSphere Application Server Community Edition w wersji 1.1 i nowszych.

O tym zadaniu

Most WebSphere MQ dla protokołu HTTP jest dostarczany jako plik `.war` (`WMQHTTP.war`).

- Na platformach UNIX i Linux
  - Produkt `WMQHTTP.war` jest częścią opcji instalacji "Java Messaging and Web Services" (Przesyłanie komunikatów języka Java i usługi Web Service). Opcja ta jest dostępna zarówno w materiałach instalacyjnych klienta, jak i na serwerze.

– Produkt `WMQHTTP.war` jest zainstalowany w katalogu *<mqmtop>*`/java/http/WMQHTTP.war`. *<mqmtop>* to katalog, w którym zainstalowano produkt WebSphere MQ .
– Produkt `WMQHTTP.samples` jest zainstalowany w katalogu *<mqmtop>*`/java/http/samples`. *<mqmtop>* to katalog, w którym zainstalowano produkt WebSphere MQ .

Wykonaj następujące czynności instalacyjne, aby zainstalować most WebSphere MQ dla protokołu HTTP, wdrażaj go i skonfiguruj, a następnie sprawdź konfigurację. Szczegółowe informacje na temat kroków konfiguracji różnią się w zależności od serwerów aplikacji. Użyj "Wdrażanie i weryfikowanie mostu WebSphere MQ dla protokołu HTTP na serwerze WebSphere Application Server V6.1.0.9" na stronie 1056 jako szablonu dla kroków, które należy wykonać na serwerze aplikacji.

Procedura

1. Uzyskaj produkt `WMQHTTP.war` , instalując klienta lub klienta MQI produktu WebSphere MQ .
2. Skopiuj `WMQHTTP.war` na serwer, z którego można go wdrożyć na serwerze aplikacji.
3. Wdróż produkt `WMQHTTP.war` na serwerze aplikacji.
4. Jeśli to konieczne, zainstaluj produkt WebSphere MQ jako adapter zasobów na serwerze aplikacji.

   Dowiedz się, czy produkt WebSphere MQ jest już skonfigurowany jako dostawca przesyłania komunikatów na serwerze aplikacji. Użyj narzędzia administracyjnego lub narzędzia do zarządzania dostarczanego z serwerem aplikacji, aby wyszukać produkt WebSphere MQ. Produkt WebSphere MQ może zostać znaleziony w następującej ścieżce: **Zasoby > JMS > Dostawcy przesyłania komunikatów**.
5. Konfigurowanie fabryki połączeń na serwerze aplikacji w celu nawiązania połączenia z menedżerem kolejek, który korzysta z transportu klienta MQI produktu WebSphere MQ .[12].
6. Skonfiguruj aplikację WWW produktu `WMQHTTP.war` na serwerze aplikacji, aby korzystała z fabryki połączeń.
7. Zweryfikuj konfigurację.
   a) Skonfiguruj menedżer kolejek o nazwie określonej w fabryce połączeń i kolejce lokalnej.
   b) Umieść komunikat w kolejce lokalnej.
   c) Utwórz kanał połączenia z serwerem o nazwie podanej w fabryce połączeń z uprawnieniem do odczytu i zapisu do kolejki lokalnej.
   d) Uruchom menedżer kolejek i program nasłuchujący.
   e) Uruchom serwer aplikacji i produkt `WMQHTTP.war`, jeśli nie są one jeszcze uruchomione.
   f) Otwórz przeglądarkę i wpisz `http://`*`hostname`*`:`*`web port`*`/`*`Context root`*`/msg/queue/`*`local queue`*

Wyniki

W oknie przeglądarki zostanie wyświetlony komunikat umieszczony w kolejce lokalnej.

Co dalej

1. Spróbuj użyć przykładu "Wdrażanie i weryfikowanie mostu WebSphere MQ dla protokołu HTTP na serwerze WebSphere Application Server V6.1.0.9" na stronie 1056.
2. Uruchom przykładowe aplikacje HTTP Java.

Wdrażanie i weryfikowanie mostu WebSphere MQ dla protokołu HTTP na serwerze WebSphere Application Server V6.1.0.9

W poniższym przykładzie przedstawiono przygotowanie wdrożenia mostu WebSphere MQ dla protokołu HTTP w celu uruchomienia przykładowych programów Java Java. Wdrożenie znajduje się na serwerze WebSphere Application Server V6.1.0.9.

[12] Początkowo, przynajmniej, skonfiguruj transport klienta. Niektóre serwery aplikacji mogą łączyć się z produktem WebSphere MQ za pomocą połączeń w trybie bezpośrednim lub w trybie powiązań.

Zanim rozpoczniesz

1. Postępuj zgodnie z instrukcjami w sekcji “Instalowanie, konfigurowanie i weryfikowanie mostu WebSphere MQ dla protokołu HTTP” na stronie 1055, aby skopiować produkt `WMQHTTP.war` na serwer dostępny dla danej instalacji serwera WebSphere Application Server.
2. Skonfiguruj menedżer kolejek i kolejkę do użycia w celu przetestowania konfiguracji:
   - W tym przykładzie menedżer kolejek jest skonfigurowany jako używany z wartościami w programie Tabela 153 na stronie 1057:

Tabela 153. Konfiguracja menedżera kolejek	
Obiekt	**Wartość**
Nazwa hosta	`itso-01`
Menedżer kolejek	`QM1`
Kolejka lokalna	`HTTPTESTQ`
Kanał połączenia z serwerem	`MYSVRCON`. Skonfiguruj identyfikator użytkownika agenta MCA z uprawnieniami wystarczą do odczytu i zapisu w programie `HTTPTESTQ`.
Port procesu nasłuchującego	`1414`

3. Uruchamianie menedżera kolejek i programu nasłuchującego
4. Umieść komunikat testowy na serwerze HTTPTESTQ. Na przykład:
   a. Uruchom program WebSphere MQ Explorer.
   b. Na liście kolejek lokalnych dla QM1kliknij prawym przyciskiem myszy opcję **HTTPTESTQ > Umieść komunikat testowy > wpisz `First Message` > Umieść komunikat > Zamknij** .
5. Uruchom serwer aplikacji i zaloguj się do konsoli Integrated Solutions Console.

O tym zadaniu

W przykładzie przedstawiono kroki, które należy wykonać, jeśli serwer aplikacji WebSphere Application Server V6.1.0.9 jest uruchomiony jako serwer aplikacji. Jeśli używana jest inna wersja serwera aplikacji WebSphere Application Server lub inny serwer aplikacji, kroki są różne. Serwer WebSphere Application Server V6.1.0.9 jest wstępnie konfigurowany przy użyciu WebSphere MQ zainstalowanego jako dostawca komunikatów, przy użyciu bibliotek klienta MQI produktu WebSphere MQ . Jeśli produkt WebSphere MQ nie jest wstępnie skonfigurowany jako dostawca przesyłania komunikatów lub jeśli wymagane jest użycie powiązań serwera WebSphere MQ , należy zainstalować i skonfigurować adapter zasobów produktu WebSphere MQ na potrzeby środowiska JEE na serwerze aplikacji.

Postępuj zgodnie z instrukcjami, aby wdrożyć most WebSphere MQ dla protokołu HTTP na serwerze WebSphere Application Server V6.1.0.9, a następnie zweryfikuj wdrożenie za pomocą przeglądarki:

Procedura

1. W panelu nawigacyjnym kliknij opcję **Zasoby > Dostawcy JMS > Dostawca przesyłania komunikatów WebSphere MQ**.

   Konfigurację można skonfigurować na poziomie węzła, komórki lub serwera, w zależności od wdrożenia serwera WebSphere Application Server. W przykładzie użyto wdrożenia na poziomie serwera.
2. W obszarze **Właściwości dodatkowe**kliknij opcję **Fabryki połączeń > Nowy**.
3. W formularzu dostawcy JMS podaj informacje zawarte w sekcji Tabela 154 na stronie 1058lub kliknij przycisk **Zastosuj > Zapisz**, aby wybrać inne opcje.

Tabela 154. Ustaw lub zmodyfikuj następujące pola	
Pole	**Wartość**
Nazwa	`WMQHTTPBridge`
Nazwa JNDI	`jms/WMQHTTPJCAConnectionFactory`
Menedżer kolejek	`QM1`
Host	`itso-01`
Port	`1414`
Kanał	`MYSVRCON`
Typ transportu	KLIENT

4. W panelu nawigacyjnym kliknij opcję **Aplikacje > Zainstaluj nową aplikację**.
5. Wstaw ścieżkę do produktu `WMQHTTP.war` w formularzu i podaj kontekstowy katalog główny, a następnie kliknij przycisk **Dalej**.
   a) Kontekstowy katalog główny jest opcjonalny. `mq` to domyślny kontekstowy katalog główny dla przykładowych aplikacji HTTP.
   b) Kontekstowy katalog główny stanowi część identyfikatora URI identyfikującego most produktu WebSphere MQ dla protokołu HTTP. Kontekstowy katalog główny można pominąć lub zmienić w późniejszym czasie.
6. Na stronie **Wybór opcji instalacji** kreatora instalacji nie ma potrzeby zmiany żadnych wartości domyślnych, a następnie kliknij przycisk **Dalej**.
7. Na stronie **Odwzoruj moduły na serwery** wybierz klaster lub serwer, zaznacz pole wyboru, a następnie kliknij przycisk **Zastosuj > Dalej**.
8. Na stronie **Odwzorowanie odwołań do zasobów na zasoby** w formularzu **javax.jms.ConnectionFactory** kliknij przycisk **Przeglądaj ...** . w przypadku mostu IBM WebSphere MQ dla wiersza HTTP.
9. Na stronie **Aplikacje korporacyjne > Dostępne zasoby** wybierz opcję **WMQHTTPBridge**, a następnie kliknij przycisk **Zastosuj**.
10. Wróć do formularza **javax.jms.ConnectionFactory** , wybierz metodę uwierzytelniania.
    a) Dla przykładu wybierz opcję **Brak**, a następnie kliknij przycisk **Zastosuj**. Pozostałe opcje wymagają dodatkowej konfiguracji.
11. Zaznacz pole wyboru **Wybierz** dla mostu IBM WebSphere MQ dla protokołu HTTP, a następnie kliknij przycisk **Dalej > Dalej > Zakończ > Zapisz** .
12. W panelu nawigacyjnym kliknij opcję **Aplikacje > Aplikacje korporacyjne**.
13. Zaznacz pole wyboru `WMQHTTP.war`, a następnie kliknij przycisk **Uruchom**.
14. Otwórz okno przeglądarki. Wpisz `http://itso-01:9080/mq/msg/queue/HTTPTESTQ`, używając odpowiedniej nazwy hosta i portu.

Wyniki

Jeśli konfiguracja zakończy się pomyślnie, w oknie przeglądarki zostanie wyświetlone okno `First Message`.

Co dalej

Uruchom przykładowe aplikacje HTTP Java.

Publikowanie/subskrypcja przy użyciu mostu WebSphere MQ dla protokołu HTTP

Most WebSphere MQ dla protokołu HTTP korzysta z klas WebSphere MQ dla interfejsu publikowania/subskrypcji JMS. Protokół HTTP **POST** tworzy publikację. Protokół HTTP **DELETE** tworzy nietrwałą subskrypcję zarządzaną. Przed użyciem identyfikatora URI tematu należy skonfigurować publikowanie/subskrybowanie usługi JMS.

Publikowanie/subskrypcja jest w pełni zintegrowana z produktem WebSphere MQ w wersji 7. Przed wersją 7 osobny broker publikowania/subskrypcji obsługuje publikacje i subskrypcje. Jest on nazywany "umieszczonym w kolejce" publikowania/subskrybowaniem, co umożliwia odróżnienie go od w pełni zintegrowanej wersji publikowania/subskrypcji w wersji 7. Wersja 7 emuluje w kolejce publikowanie/subskrybowanie przy użyciu zintegrowanego publikowania/subskrypcji. Emulacja umożliwia współistnienie istniejących w kolejce aplikacji publikowania/subskrybowania ze zintegrowanymi aplikacjami działanianymi w tym samym menedżerze kolejek. Kolejkowane aplikacje publikowania/subskrypcji mogą również współdziałać z aplikacjami zintegrowanymi, współużytkowając te same tematy. W wersji 6 broker został dostarczony z produktem WebSphere MQ; przed wersją 6 był on dostępny jako pakiet serwisowy SupportPack.

Konfiguracja

Most produktu WebSphere MQ dla protokołu HTTP używa interfejsu JMS do publikowania i subskrybowania. W wersji 7 można sterować tym, czy klasy produktu WebSphere MQ dla usługi JMS są umieszczane w kolejce lub w zintegrowanym publikowania/subskrypcji, przy użyciu właściwości JMS produktu `PROVIDERVERSION` .

Dodatkową kwestią jest to, że można użyć bibliotek klienta MQI produktu WebSphere MQ z mostem WebSphere MQ dla protokołu HTTP lub bibliotek serwera. Biblioteki klienta w wersji 6 obsługują tylko w kolejce publikowania/subskrybowania, natomiast biblioteki w wersji 7 obsługują zarówno w kolejce, jak i zintegrowaną publikowanie/subskrybowanie. Większość serwerów WWW lub serwerów aplikacji korzystających z produktu WebSphere MQ jako dostawcy przesyłania komunikatów może używać bibliotek klienta. Aby można było używać zintegrowanego publikowania/subskrybowania, zarówno biblioteki klienta MQI produktu WebSphere MQ , jak i serwera muszą być co najmniej w wersji 7. Jeśli używana jest wcześniejsza wersja produktu WebSphere niż 7, należy skonfigurować publikowanie/subskrybowanie w kolejce. Patrz sekcja Tabela 155 na stronie 1059. Sprawdź, które biblioteki są zainstalowane lub skonfigurowane razem z serwerem WWW lub serwerem aplikacji, który jest używany.

Tabela 155. Tryby konfiguracji publikowania/subskrypcji

	Klient V6 lub wcześniejszy	**Klient V7 lub nowszy**
Serwer V6 lub wcześniejszy	1. Uruchom skrypt `\java\bin\MQJMS_PSQ.mqsc` .	Nieobsługiwane
Serwer V7 lub nowszy	1. Uruchom skrypt `\java\bin\MQJMS_PSQ.mqsc` . 2. Ustaw menedżera kolejek na wartość `PSMODE=ENABLED`	1. If PROVIDERVERSION = 7 a. Ustaw menedżera kolejek na wartość `PSMODE=ENABLED` lub `PSMODE=COMPAT` 2. If PROVIDERVERSION = 6 a. Ustaw menedżera kolejek na wartość `PSMODE=ENABLED`

Publikowanie

Wyślij żądanie HTTP **POST** z identyfikatorem URI:

```
http://hostname:port/context_root/msg/topic/topicString
```

Treść komunikatu jest publikowana przy użyciu łańcucha tematu *topicString*.

Subskrybowanie

Wyślij żądanie HTTP **DELETE** z identyfikatorem URI:

```
http://hostname:port/context_root/msg/topic/topicString
```

Most WebSphere MQ dla protokołu HTTP tworzy trwałą, nietrwałą subskrypcję dla łańcucha tematu *topicString*. Subskrypcja jest usuwana zaraz po zwróconej publikacji lub do momentu, gdy upłynie odstęp czasu oczekiwania ustawiony przez niestandardowy nagłówek jednostki, `x-msg-wait`.

Uruchamianie mostu produktu WebSphere MQ dla przykładów HTTP

Most WebSphere MQ dla przykładów HTTP jest dostępny do użycia tylko w systemie operacyjnym Windows . W przykładach przedstawiono sposób przesyłania komend HTTP **POST** i HTTP **DELETE** do mostu WebSphere MQ dla protokołu HTTP z programów Java.

Zanim rozpoczniesz

Zweryfikuj instalację mostu WebSphere MQ dla instalacji HTTP, wykonując krok “7” na stronie 1056 w sekcji “Instalowanie, konfigurowanie i weryfikowanie mostu WebSphere MQ dla protokołu HTTP” na stronie 1055.

Przykłady HTTP są instalowane w katalogach pokazanych w programie Tabela 156 na stronie 1060. W każdym przypadku kod źródłowy jest instalowany w podkatalogu `/src` .

Tabela 156. Położenie przykładów HTTP

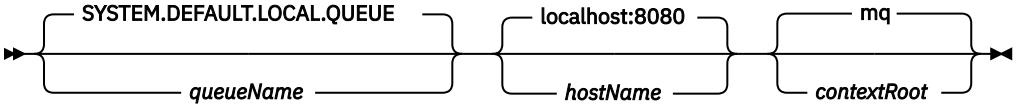
Platforma	**Położenie**
Windows	*MQ_INSTALLATION_PATH*/tools/http/samples
Wszystkie pozostałe platformy	*MQ_INSTALLATION_PATH*/samp/http
MQ_INSTALLATION_PATH reprezentuje katalog, w którym zainstalowany jest produkt WebSphere MQ .	

O tym zadaniu

Przykłady symulują przykładowe aplikacje produktu WebSphere MQ AMQSPUT i AMQSGET. Ilustrują one następujące funkcje w środowisku przesyłania komunikatów w trybie punkt z punktem:

- **HTTPPOST** -wysyła żądania HTTP **POST** w aplikacji Java w celu umieszczania komunikatów w kolejce produktu WebSphere MQ przy użyciu mostu WebSphere MQ dla protokołu HTTP i obsługuje odpowiedzi.
- **HTTPDELETE** -umożliwia wysyłanie żądań HTTP **DELETE** w aplikacji Java w celu pobrania komunikatów z kolejki produktu WebSphere MQ przy użyciu mostu WebSphere MQ dla protokołu HTTP i obsługuje odpowiedzi zawierające komunikat WebSphere MQ .

Parametry dla HTTPPOST i HTTPDELETE



Aby uruchomić przykład produktu **HTTPPOST** , wykonaj następujące kroki:

Procedura

1. W oknie komend przejdź do katalogu przykładów HTTP.
2. Uruchom przykład **HTTPPOST** .

   ```
   java -classpath . HTTPPOST [parameters]
   ```

   Po uruchomieniu przykładu **HTTPPOST** wyświetlane są następujące dane wyjściowe:

   ```
   HTTP POST Sample start
   Target server is 'hostName'
   Target queue is 'queueName'
   Target context-root is 'contextRoot'
   ```

3. W wierszu komend wpisz tekst, który ma utworzyć treść wiadomości.
4. Naciśnij klawisz Enter, aby opublikować komunikat w kolejce produktu WebSphere MQ .
   a) Jeśli chcesz wysłać kolejny komunikat, wprowadź jeszcze więcej tekstu.
      Teksty tworzą treść drugiego komunikatu produktu WebSphere MQ .
   b) Naciśnij klawisz Enter, aby opublikować komunikat w kolejce produktu WebSphere MQ .
5. Naciśnij klawisz Enter dwukrotnie, aby zakończyć **HTTPPOST**.
   Zostaną wyświetlone następujące dane wyjściowe:

   ```
   HTTP POST Sample end
   ```

Co dalej

Przykład **HTTPDELETE** wykonuje destrukcyjne pobieranie wszystkich komunikatów umieszczonych w kolejce produktu WebSphere MQ .

Uruchom przykład produktu **HTTPDELETE** , wykonując następujące kroki:

1. W oknie komend przejdź do sekcji *MQ_INSTALLATION_PATH*/tools/samples.
   MQ_INSTALLATION_PATH Reprezentuje katalog, w którym zainstalowano produkt WebSphere MQ .
2. Uruchom przykład **HTTPDELETE** .

   ```
   java -classpath . HTTPPOST [parameters]
   ```

   Po uruchomieniu przykładu **HTTPDELETE** wyświetlane są następujące dane wyjściowe:

   ```
   HTTP DELETE Sample start
   Target server is 'host:port'
   Target queue is 'your queue name'
   Target context-root is 'your context-root'
   message
   message
   ...
   ```

Uwagi dotyczące zabezpieczeń mostu WebSphere dla protokołu HTTP

W celu uwierzytelnienia klienta przeglądarki WWW stosowane są standardowe uwagi dotyczące zabezpieczeń WWW. Autoryzacja do zasobów WebSphere MQ jest na poziomie użytkownika uruchamianego przez most WebSphere Bridge dla serwletu HTTP, a nie indywidualnego klienta przeglądarki WWW. W produkcie WebSphere MQstosowane są standardowe zabezpieczenia produktu WebSphere MQ .

Dane przepływające z przeglądarki WWW do aplikacji WebSphere MQ za pomocą mostu WebSphere dla protokołu HTTP i z powrotem, są wykonywane trzy kroki:

Połączenie klienta
Z poziomu przeglądarki do mostu WebSphere dla protokołu HTTP przez połączenie TCP/IP przy użyciu protokołu HTTP.

Połączenie adaptera zasobów z produktem WebSphere MQ
Połączenie pochodzi z mostu WebSphere dla protokołu HTTP do menedżera kolejek produktu WebSphere MQ . Połączenie to połączenie klienta, za pośrednictwem protokołu TCP/IP lub lokalne połączenie powiązań produktu WebSphere MQ . Po nawiązaniu połączenia żądanie HTTP jest umieszczane w standardowej kolejce lokalnej lub w kolejce transmisji.

Z kolejki lokalnej produktu WebSphere MQ dla jednego lub większej liczby kanałów, do kolejki docelowej.
Zastosowanie standardowych technik zabezpieczania kolejek, tematów, menedżerów kolejek i kanałów.

Odpowiedź podejmuje kroki w odwrotnej kolejności.

Połączenie klienta

Bezpieczne połączenia między klientami HTTP i serwerem aplikacji przy użyciu kontenera WWW. Użyj standardowych technik serwera HTTP, takich jak HTTPS. Informacje na ten temat można znaleźć w dokumentacji serwera aplikacji.

Połączenie adaptera zasobów z produktem WebSphere MQ

Połączenie między adapterem zasobów i menedżerem kolejek jest autoryzowane przy użyciu tylko jednego identyfikatora użytkownika. Przypisz jeden ID użytkownika, aby zidentyfikować żądania z WebSphere Bridge for HTTP. ID użytkownika musi mieć ograniczone uprawnienia WebSphere MQ tylko do tych zasobów, do których użytkownicy zewnętrzni muszą mieć dostęp. Należy oddzielnie uwierzytelnić klienta rzeczywistego, a także ustanowić zaufanie do kolejnych interakcji z klientem przy użyciu standardowych technik zabezpieczeń WWW.

Zabezpieczy połączenie między adapterem zasobów i menedżerem kolejek przy użyciu pojedynczego identyfikatora użytkownika. Ogranicz uprawnienia, aby ID użytkownika nie miał więcej niż wymagany do odczytu i zapisu komunikatów do kolejek i tematów. Most WebSphere dla protokołu HTTP jest punktem ataku między Internetem a intranetem.

Sposób zabezpieczania połączenia między adapterem zasobów i produktem WebSphere MQ jest zależny od konkretnego adaptera zasobów. Refer to the documentation for the resource adapter.

Korzystanie z interfejsu modelu obiektu komponentu (klasy automatyzacji produktu WebSphere MQ dla elementu ActiveX)

Klasy automatyzacji produktu WebSphere MQ dla ActiveX (MQAX) to komponenty ActiveX , które udostępniają klasy, których można użyć w aplikacji w celu uzyskania dostępu do produktu WebSphere MQ.

Produkt MQAX wymaga środowiska WebSphere MQ i odpowiedniej aplikacji produktu WebSphere MQ , z którą ma być komuniowany.

Dzięki temu aplikacja ActiveX może uruchamiać transakcje i uzyskiwać dostęp do danych w dowolnym systemie korporacyjnym, do którego można uzyskać dostęp za pośrednictwem produktu WebSphere MQ.

Klasy automatyzacji produktu WebSphere MQ dla elementu ActiveX:

- Dostęp do funkcji i funkcji interfejsu API produktu WebSphere MQ można uzyskać, zezwalając na pełne połączenie z innymi platformami WebSphere MQ .
- Zgodne z normalnymi konwencjami, które są oczekiwane w przypadku komponentu ActiveX .
- Jest on zgodny z modelem obiektu WebSphere MQ , który jest również dostępny dla środowisk .NET, C + +, Java i LotusScript.

Dostępne są przykłady startowe MQAX. Tych przykładów można użyć początkowo w celu sprawdzenia, czy instalacja produktu MQAX powiodła się i czy jest to podstawowe środowisko produktu WebSphere MQ . Przykłady pokazują również, w jaki sposób można użyć produktu MQAX.

Skrypty COM i ActiveX

Model obiektu komponentu (Component Object Model-COM) jest modelem programistycznym opartym na obiektowym systemie zdefiniowanym przez firmę Microsoft. Określa on, w jaki sposób komponenty oprogramowania mogą być udostępniane w sposób umożliwiający ich zlokalizowanie i komunikację niezależnie od języka komputera, w którym są one napisane lub w którym znajdują się ich położenie.

ActiveX to zestaw technologii opartych na technologii COM, który integruje projektowanie aplikacji, komponenty wielokrotnego użytku oraz technologie internetowe na platformach Microsoft Windows . Komponenty ActiveX udostępniają interfejsy, które mogą być dostępne dynamicznie przez aplikacje. Klient skryptowy ActiveX to aplikacja, na przykład kompilator, która może zbudować lub wykonać program lub skrypt, który korzysta z interfejsów udostępnianych przez komponenty ActiveX (lub COM).

Obsługa środowiska WebSphere MQ

Klasy automatyzacji produktu WebSphere MQ dla elementu ActiveX mogą być używane tylko z **32-bitowych** klientami skryptowymi ActiveX .

Komponent COM może być używany tylko dla aplikacji **32-bitowych** . Jeśli chcesz napisać 64-bitową aplikację COM, możesz użyć interfejsu .NET.

Aby uruchomić produkt MQAX w środowisku serwera WebSphere MQ , w systemie musi być zainstalowany system Windows 2000 lub nowszy.

Aby uruchomić program MQAX w środowisku klienta MQI produktu WebSphere MQ , w systemie Okna 2000 lub nowszym w systemie musi być zainstalowany klient MQI produktu WebSphere MQ :

Klient MQI produktu WebSphere MQ wymaga dostępu do co najmniej jednego serwera produktu WebSphere MQ . Jeśli w systemie są zainstalowane zarówno klienta MQI produktu WebSphere MQ , jak i serwer WebSphere MQ , aplikacje MQAX zawsze działają na serwerze. Interfejs ActiveX do interfejsu MQAI jest dostępny tylko w środowiskach serwerów WebSphere MQ .

Projektowanie i programowanie za pomocą klas automatyzacji WebSphere MQ dla ActiveX

Projektowanie aplikacji MQAX, które uzyskują dostęp do aplikacji innych niżActiveX

Klasy automatyzacji produktu WebSphere MQ zapewniają dostęp do funkcji interfejsu API produktu WebSphere MQ . W związku z tym można skorzystać ze wszystkich zalet, które przy użyciu produktu WebSphere MQ mogą być używane w aplikacji systemu Windows .

Ogólna konstrukcja aplikacji jest taka sama, jak w przypadku dowolnej aplikacji produktu WebSphere MQ , dlatego należy rozważyć wszystkie aspekty projektu opisane w sekcji “Projektowanie aplikacji” na stronie 7 .

Aby używać klas automatyzacji produktu WebSphere MQ , należy zakodować programy w systemie Windows w aplikacji przy użyciu języka, który obsługuje tworzenie i używanie obiektów COM. Na przykład Visual Basic, Java i inne klienty skryptowe ActiveX . Klasy mogą być następnie łatwo zintegrowane z aplikacją, ponieważ potrzebne obiekty WebSphere MQ mogą być kodowane przy użyciu rodzimej składni języka implementacji.

Korzystanie z klas automatyzacji produktu WebSphere MQ dla elementu ActiveX

Podczas projektowania aplikacji ActiveX , która korzysta z klas automatyzacji WebSphere MQ dla ActiveX, najważniejszym elementem informacji jest komunikat wysyłany lub odbierany ze zdalnego systemu WebSphere MQ . Dlatego należy znać format elementów, które są wstawiane do komunikatu. W przypadku skryptu MQAX do pracy, zarówno ta, jak i aplikacja WebSphere MQ , która pobiera lub wysyła komunikat, musi znać strukturę komunikatu.

Jeśli wysyłany jest komunikat z aplikacją MQAX i ma zostać wykonana konwersja danych na końcu MQAX, należy również wiedzieć, że:

- Strona kodowa używana przez system zdalny
- Kodowanie używane przez system zdalny

Aby zachować przenośny kod, zaleca się ustawienie strony kodowej i kodowania, nawet jeśli są one takie same w systemach wysyłających i odbierających.

Rozważając strukturę implementacji projektowanego systemu, należy pamiętać, że skrypty MQAX działają na tym samym komputerze, na którym jest zainstalowany menedżer kolejek produktu WebSphere MQ lub klient WebSphere MQ .

Wskazówki dotyczące programowania

Następujące wskazówki i wskazówki nie są w znaczącym porządku. Są to tematy, które, jeśli mają znaczenie dla pracy, którą wykonujesz, mogą zaoszczędzić czas.

Właściwości deskryptora komunikatu

W przypadku manipulowania właściwościami deskryptora komunikatu w programie może być lepiej użycie szesnastkowych odpowiedników pól.

Informacje zawarte w tej sekcji odnoszą się do następujących właściwości:

- AccountingToken
- CorrelationId
- GroupId
- MessageId

Jeśli aplikacja WebSphere MQ jest inicjatorem komunikatu, a produkt WebSphere MQ generuje te właściwości, lepiej jest użyć właściwości AccountingTokenHex, CorrelationIdHex, GroupIdHex i MessageIdHex, jeśli mają być wyświetlane ich wartości lub manipulować nimi w dowolny sposób, z uwzględnieniem przekazywania ich z powrotem w komunikacie do produktu WebSphere MQ. Wynika to z tego, że wygenerowane wartości produktu WebSphere MQ są łańcuchami bajtów, które mają dowolną wartość z zakresu od 0 do 255 włącznie, nie są łańcuchami znaków drukowalnych.

W przypadku, gdy skrypt MQAX jest inicjatorem komunikatu, można użyć właściwości AccountingToken, CorrelationId, GroupIdi MessageId lub ich odpowiedników Hex.

Stałe produktu WebSphere MQ

Stałe produktu WebSphere MQ są udostępniane jako elementy typu enum WebSphere MQ w bibliotece MQAX200.

Stałe łańcuchowe produktu WebSphere MQ

WebSphere MQ -stałe łańcuchowe i odpowiadające im łańcuchy znaków.

Stałe łańcuchowe produktu WebSphere MQ nie są dostępne w przypadku używania klas automatyzacji produktu WebSphere MQ dla elementu ActiveX. Należy użyć jawnego łańcucha znaków dla tych, które znajdują się na poniższej liście, a także innych, które mogą być potrzebne. Komendy muszą być dopełniane do ośmiu znaków za pomocą spacji:

MQFMT_NONE	" "
ADMINISTRATOR MQFMT_ADMIN	"MQADMIN"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM"
MQFMT_CICS	"MQCICS"
MQFMT_COMMAND_1	"MQCMD1 "

MQFMT_COMMAND_2	"MQCMD2 "
MQFMT_DEAD_LETTER_HEADER	"MQDEAD"
MQFMT_DIST_HEADER	"MQHDIST"
Zdarzenie MQFMT_EVENT	"MQEVENT"
MQFMT_IMS	"MQIMS"
MQFMT_IMS_VAR_STRIsNG	"MQIMSVS"
MQFMT_MD_EXTENSION	"MQHMDE"
MQFMT_PCF	"MQPCF"
MQFMT_REF_MSG_HEADER	"MQHREF"
MQFMT_RF_HEADER	"MQHRF"
MQFMT_STRING	"MQSTR"
MQFMT_TRIGGER	"MQTRIG"
Nagłówek MQFMT_WORK_INFO_HEADER	"MQHWIH"
MQFMT_XMIT_Q_HEADER	"MQXMIT"

Stałe łańcuchowe o wartości NULL

Stałe produktu WebSphere MQ używane do inicjowania czterech właściwości MQMessage, MQMI_NONE (24 znaki NULL), MQCI_NONE (24 znaki NULL), MQGI_NONE (24 znaki NULL) i MQACT_NONE (32 znaki NULL) nie są obsługiwane przez klasy automatyzacji WebSphere MQ dla ActiveX. Ustawienie ich na puste łańcuchy ma taki sam efekt.

Na przykład, aby ustawić różne identyfikatory komunikatu MQMessage na następujące wartości: *mymessage*.**MessageId** = "" *mymessage*.**CorrelationId** = "" *mymessage*.**AccountingToken** = ""

Odbieranie komunikatu z produktu WebSphere MQ

Istnieje kilka sposobów otrzymywania komunikatu z produktu WebSphere MQ:

- Odpytywanie przez wydanie komendy GET, po której następuje oczekiwanie, przy użyciu funkcji Visual Basic TIMER.
- Wydanie komendy GET z opcją oczekiwania; należy określić przedział czasu oczekiwania, ustawiając właściwość WaitInterval . Należy wziąć pod uwagę, że nawet jeśli system jest uruchamiany w środowisku wielowątkowym, wówczas oprogramowanie działające w danym momencie może działać tylko w postaci jednowątkowej. Pozwala to na uniknięcie blokady systemu w nieskończoność.

  Nie ma wpływu na inne wątki. Jeśli jednak inne wątki wymagają dostępu do produktu WebSphere MQ, wymagają one drugiego połączenia z produktem WebSphere MQ za pomocą dodatkowego menedżera kolejek MQAX i obiektów kolejki.

Wydanie komendy GET z opcją oczekiwania i ustawienie parametru WaitInterval na wartość MQWI_UNLIMITED powoduje, że system zablokuje się do momentu zakończenia operacji GET, jeśli proces jest jednowątkowy.

Używanie konwersji danych

WebSphere MQ Automation Classes for ActiveX -kodowanie liczbowe i konwersja zestawu znaków są obsługiwane przez dwie formy konwersji danych.

Kodowanie liczbowe

Jeśli właściwość Kodowanie komunikatu MQMessage zostanie ustawiona, następujące metody są przekształcane między różnymi numerycznymi systemami kodowania:

- Metoda ReadDecimal2
- Metoda ReadDecimal4
- Metoda ReadDouble
- Metoda ReadDouble4
- Metoda ReadFloat
- Metoda ReadInt2
- Metoda ReadInt4
- Metoda ReadLong
- Metoda ReadShort
- Metoda ReadUInt2
- Metoda WriteDecimal2
- Metoda WriteDecimal4
- Metoda WriteDouble
- Metoda WriteDouble4
- Metoda WriteFloat
- Metoda WriteInt2
- Metoda WriteInt4
- Metoda WriteLong
- Metoda WriteShort
- Metoda WriteUInt2

Właściwość Kodowanie może być ustawiona i interpretowana przy użyciu dostarczonych stałych WebSphere MQ . Rysunek 216 na stronie 1066 przedstawia przykład następujących elementów:

```
/* Encodings for Binary Integers */
        MQENC_INTEGER_UNDEFINED
        MQENC_INTEGER_NORMAL
        MQENC_INTEGER_REVERSED

/* Encodings for Decimals */
        MQENC_DECIMAL_UNDEFINED
        MQENC_DECIMAL_NORMAL
        MQENC_DECIMAL_REVERSED

/* Encodings for Floating-Point Numbers */
        MQENC_FLOAT_UNDEFINED
        MQENC_FLOAT_IEEE_NORMAL
        MQENC_FLOAT_IEEE_REVERSED
        MQENC_FLOAT_S390
```

Rysunek 216. Dostarczane stałe produktu WebSphere MQ do kodowania

Na przykład, aby wysłać liczbę całkowitą z systemu Intel do systemu operacyjnego System/390 w kodowaniu System/390 :

```
Dim msg As New MQMessage   'Define a WebSphere MQ message for our use..
Print msg.Encoding    'Currently 546 (or X'222')
                                          'Set the encoding property
                                           to 785 (or X'311')
msg.Encoding = MQENC_INTEGER_NORMAL OR MQENC_DECIMAL_NORMAL
                        OR MQENC_FLOAT_S390
Print msg.Encoding    'Print it to see the change
Dim local_num As long 'Define a long integer
local_num = 1234                          'Set it
msg.WriteLong(local_num)         'Write the number into the message
```

Konwersja zestawu znaków

Konwersja zestawu znaków jest niezbędna, gdy wysyłany jest komunikat z jednego systemu do innego systemu, w którym strony kodowe są różne. Konwersja strony kodowej jest używana przez:

- Metoda ReadString
- Metoda ReadNullTerminatedString
- Metoda WriteString
- Metoda WriteNullTerminatedString
- Właściwość MessageData

Właściwość MQMessage CharacterSet należy ustawić na obsługiwaną wartość zestawu znaków (CCSID).

Klasy automatyzacji WebSphere MQ dla produktu ActiveX korzystają z tabel konwersji w celu wykonania konwersji zestawu znaków.

Na przykład, aby automatycznie przekształcić łańcuchy w stronę kodową 437:

```
Dim msg As New MQMessage                'Define a WebSphere MQ message
msg.CharacterSet = 437                 'Set code page required
msg.WriteString "A character string"'Put character string in message
```

Metoda WriteString pobiera dane łańcuchowe (w tym przykładzie łańcuch znaków) jako łańcuch Unicode. Następnie przekształca te dane z kodu Unicode na stronę kodową 437 przy użyciu tabeli konwersji 34B001B5.TBL.

Znaki w łańcuchu Unicode, które nie są obsługiwane przez stronę kodową 437, otrzymują standardowy znak zastępczy ze strony kodowej 437.

Podobnie, jeśli używana jest metoda ReadString , komunikat przychodzący ma zestaw znaków ustalony przez wartość WebSphere MQ Message Descriptor (MQMD), a konwersja z tej strony kodowej na Unicode przed przekazaniem jej do języka skryptowego.

Wątki

Klasy automatyzacji produktu WebSphere MQ dla elementu ActiveX implementują model wielowątkowości, w którym obiekty mogą być używane między wątkami.

Podczas gdy produkt MQAX zezwala na użycie obiektów MQQueue i MQQueueManager , produkt WebSphere MQ nie zezwala obecnie na współużytkowanie uchwytów między różnymi wątkami.

Próby użycia tych elementów w innym wątku powodują wystąpienie błędu, a produkt WebSphere MQ zwraca kod powrotu MQRC_HCONN_ERROR.

Uwaga: Dla każdego procesu istnieje tylko jeden obiekt MQSession. Użycie funkcji MQSession CompletionCode i ReasonCode nie jest zalecane w środowiskach wielowątkowych. Wartości błędów MQSession mogą zostać nadpisane przez drugi wątek między błędem podniesionym i sprawdzonym w pierwszym wątku. Wątki są serializowane przez czas trwania każdego wywołania metody lub dostępu do właściwości. Dlatego wydanie komendy Get z opcją oczekiwania powoduje, że inne wątki uzyskają dostęp do obiektów MQAX, które mają zostać zawieszone do czasu zakończenia operacji.

Obsługa błędów

Te informacje opisują właściwości obiektu MQAX, sposób obsługi błędów, reguły opisujące sposób obsługi wyjątków i uzyskiwanie właściwości.

Każdy obiekt MQAX zawiera właściwości służące do przechowywania informacji o błędach i metody ich resetowania lub czyszczenia. Dostępne są następujące właściwości:

- CompletionCode
- ReasonCode
- ReasonName

Metoda jest następująca:

- Kody ClearError

Jak działa błąd

Skrypt lub aplikacja MQAX wywołuje metodę obiektu MQAX lub uzyskuje dostęp do niej lub aktualizuje właściwość obiektu MQAX:

1. Wartości ReasonCode i CompletionCode w danym obiekcie są aktualizowane.
2. Atrybuty ReasonCode i CompletionCode w obiekcie MQSession są również aktualizowane przy użyciu tych samych informacji.

   Uwaga: W sekcji "Wątki" na stronie 1067 można znaleźć ograniczenia dotyczące używania kodów błędów MQSession w aplikacjach wielowątkowych.

   Jeśli właściwość CompletionCode jest większa lub równa właściwości ExceptionThreshold w sesji MQSession, MQAX zgłasza wyjątek (numer 32000). Użyj tego w skrypcie, używając instrukcji On Error (lub równoważnej), aby ją przetworzyć.
3. Użyj funkcji Error, aby pobrać powiązany łańcuch błędu, który ma postać:

   ```
   MQAX: CompletionCode=xxx, ReasonCode=xxx, ReasonName=xxx
   ```

   Więcej informacji na temat używania instrukcji On Error można znaleźć w dokumentacji języka skryptowego ActiveX .

   Korzystanie z kodu CompletionCode i ReasonCode w obiekcie MQSession jest wygodne w przypadku prostych procedur obsługi błędów.

   Właściwość ReasonName zwraca nazwę symboliczną produktu WebSphere MQ dla bieżącej wartości parametru ReasonCode.

Zgłaszanie wyjątków

W poniższych regułach opisano sposób obsługi wyjątków:

- Za każdym razem, gdy właściwość lub metoda ustawia kod zakończenia na wartość większą lub równą progowi wyjątku (zwykle jest ustawiona na wartość 2), zgłaszany jest wyjątek.
- Wszystkie wywołania metod i zestawy właściwości ustawiają kod zakończenia.

Pobieranie właściwości

Jest to przypadek szczególny, ponieważ wartości CompletionCode i ReasonCode nie są zawsze aktualizowane:

- Jeśli właściwość zostanie pomyślnie zakończona, obiekt i obiekt MQSession ReasonCode i CompletionCode pozostaną niezmienione.
- Jeśli właściwość nie powiedzie się i zostanie wyświetlony kod CompletionCode ostrzeżenia, wartości ReasonCode i CompletionCode pozostaną niezmienione.
- Jeśli właściwość nie powiedzie się i zostanie zgłoszony błąd CompletionCode , wartości ReasonCode i CompletionCode zostaną zaktualizowane tak, aby odzwierciedlały prawdziwe wartości, a przetwarzanie błędów jest kontynuowane zgodnie z opisem.

Klasa MQSession ma metodę *ReasonCodeName* , która może być używana do zastępowania kodu przyczyny produktu WebSphere MQ nazwą symboliczną. Jest to szczególnie przydatne podczas tworzenia programów, w których mogą wystąpić nieoczekiwane błędy. Nazwa nie jest jednak idealna do prezentacji dla użytkowników.

Każda klasa ma również właściwość *ReasonName*, która zwraca nazwę symboliczną bieżącego kodu przyczyny dla tej klasy.

WebSphere MQ Automation Classes for ActiveX reference

W tej sekcji opisano klasy klas automatyzacji produktu WebSphere MQ dla ActiveX (MQAX), opracowane dla ActiveX. Klasy umożliwiają pisanie aplikacji ActiveX , które mogą uzyskiwać dostęp do innych aplikacji działających w środowiskach innych niżActiveX , przy użyciu produktu WebSphere MQ.

Klasy automatyzacji WebSphere MQ dla interfejsu ActiveX

Klasy automatyzacji produktu WebSphere MQ dla elementu ActiveX udostępniają predefiniowane stałe liczbowe ActiveX (takie jak MQMT_REQUEST), które są potrzebne do korzystania z klas.

Klasy automatyzacji ActiveX składają się z następujących elementów:

- “Klasa MQSession” na stronie 1070
- “Klasa MQQueueManager” na stronie 1073
- “Klasa MQQueue” na stronie 1085
- “Klasa MQMessage” na stronie 1099
- “Klasa opcji MQPutMessage” na stronie 1121
- “Klasa opcji MQGetMessage” na stronie 1123
- “Klasa MQDistributionList” na stronie 1126
- “MQDistributionList-klasa elementu” na stronie 1130

Ponadto klasy automatyzacji WebSphere MQ dla elementu ActiveX udostępniają predefiniowane stałe liczbowe ActiveX (takie jak MQMT_REQUEST), które są potrzebne do korzystania z klas. Są one dostarczane w ramach typu wyliczeniowego MQ w bibliotece MQAX200. Stałe są podzbiorem tych zdefiniowanych w plikach nagłówkowych WebSphere MQ C (cmqc * .h) z dodatkowymi klasami automatyzacji WebSphere MQ dla kodów przyczyny ActiveX .

Informacje o klasach automatyzacji produktu WebSphere MQ dla klas ActiveX

Informacje zawarte w tej sekcji znajdują się obok tematów referencyjnych w sekcji Tworzenie odwołań do aplikacji.

Informacje na temat ważnych informacji można znaleźć w sekcji Funkcje, które mogą być używane tylko z instalacją podstawową w systemie Windows .

Klasa MQSession udostępnia obiekt główny, który zawiera status ostatniego działania wykonywanego dla dowolnego z obiektów MQAX. Więcej informacji zawiera sekcja “Obsługa błędów” na stronie 1067.

Klasy MQQueueManager i MQQueue zapewniają dostęp do bazowych obiektów produktu WebSphere MQ . Metody lub dostępy właściwości dla tych klas w ogólnym wyniku są wykonywane w ramach wywołań MQI produktu WebSphere MQ .

Klasy opcji MQMessage, MQPutMessagei MQGetMessageobudowują struktury danych MQMD, MQPMO i MQGMO i są używane w celu ułatwienia wysyłania komunikatów do kolejek i pobierania z nich komunikatów.

Klasa MQDistributionList hermetykuje kolekcję kolejek-lokalnych, zdalnych lub aliasów dla danych wyjściowych. Klasa elementu MQDistributionListhermetykuje struktury MQOR, MQRR i MQPMR i wiąże je z listą dystrybucyjną będącą właścicielem.

Przekazywanie parametrów

Parametry w wywołaniach metod są przekazywane przez wartość, z wyjątkiem sytuacji, w której parametr ten jest obiektem, w którym to przypadku jest to odniesienie, które jest przekazywane.

W definicjach klas podano listę typów danych dla każdego parametru lub właściwości. W przypadku wielu klientów ActiveX , takich jak Visual Basic, jeśli używana zmienna nie jest typu wymaganego, wartość jest

automatycznie przekształcana w wymagany typ lub z wymaganego typu, a taka konwersja jest możliwa. Jest to zgodne ze standardowymi regułami klienta; MQAX nie zapewnia takiej konwersji.

Wiele metod jest typu łańcuchowego o stałej długości lub zwraca łańcuch znaków o stałej długości. Reguły konwersji są następujące:

- Jeśli użytkownik dostarcza łańcuch o stałej długości w niepoprawnej długości, jako parametr wejściowy lub jako wartość zwracaną, wartość jest obcinana lub dopełniona spacjami kończącymi się zgodnie z wymaganiami.
- Jeśli użytkownik dostarcza łańcuch o zmiennej długości o niepoprawnej długości jako parametr wejściowy, wartość jest obcinana lub dopełniona spacjami kończącymi.
- Jeśli użytkownik dostarcza łańcuch o zmiennej długości o niepoprawnej długości jako wartość zwracaną, to łańcuch jest dopasowywany do wymaganej długości (ponieważ zwracanie wartości niszczy poprzednią wartość w łańcuchu).
- Łańcuchy podane jako parametry wejściowe mogą zawierać osadzone Nulls.

Klasy te można znaleźć w bibliotece MQAX200 .

Metody dostępu do obiektów

Metody te nie są bezpośrednio związane z żadnym pojedynczym wywołaniem produktu WebSphere MQ . Każda z tych metod tworzy obiekt, w którym są przechowywane informacje o odwołaniu, a następnie następuje połączenie z obiektem WebSphere MQ lub jego otwarcie:

Po nawiązaniu połączenia z menedżerem kolejek atrybut uchwytu połączenia jest generowany przez produkt WebSphere MQ.

Po otwarciu kolejki atrybut "uchwyt obiektu" jest generowany przez produkt WebSphere MQ.

Te atrybuty produktu WebSphere MQ nie są bezpośrednio dostępne dla programu MQAX.

Błędy

Błędy syntaktyczne podczas przekazywania parametrów mogą być wykrywane w czasie kompilacji i w czasie wykonywania przez klienta ActiveX . Błędy mogą zostać uwięzione przy użyciu opcji On Error w Visual Basic.

Wszystkie klasy ActiveX produktu WebSphere MQ zawierają dwie specjalne właściwości tylko do odczytu- ReasonCode i CompletionCode. Właściwości te można odczytać w dowolnym momencie.

Próba uzyskania dostępu do dowolnej innej właściwości lub wywołanie dowolnej metody wywołania metody może spowodować wygenerowanie błędu z produktu WebSphere MQ.

Jeśli zestaw właściwości lub wywołanie metody zakończy się powodzeniem, parametr ReasonCode obiektu będącego właścicielem jest ustawiony na wartość MQRC_NONE, a CompletionCode jest ustawiony na wartość MQCC_OK.

Jeśli dostęp do właściwości lub wywołanie metody nie powiedzie się, w tych polach są ustawiane kody przyczyny i zakończenia.

Klasa MQSession

Jest to klasa główna dla klas automatyzacji produktu WebSphere MQ dla elementu ActiveX.

Dla każdego procesu klienta ActiveX zawsze istnieje tylko jeden obiekt MQSession. Próba utworzenia drugiego obiektu powoduje utworzenie drugiego odwołania do oryginalnego obiektu.

Tworzenie

Nowy -tworzy nowy obiekt MQSession.

Składnia

Dim *mqsess* **As New MQSession Set** *mqsess* = **Nowa sesja MQSession**

Właściwości

- “Właściwość CompletionCode” na stronie 1071.
- “Właściwość ExceptionThreshold” na stronie 1071.
- “Właściwość ReasonCode” na stronie 1072.
- “Właściwość ReasonName” na stronie 1072.

Metoda

- “Metoda AccessGetMessageOptions” na stronie 1072.
- “Metoda AccessMessage” na stronie 1072.
- “Metoda AccessPutMessageOptions” na stronie 1072.
- “Metoda menedżera AccessQueue” na stronie 1073.
- “ClearError-metoda kodów” na stronie 1073.
- “Metoda ReasonCodeName” na stronie 1073.

Właściwość CompletionCode

Tylko do odczytu. Zwraca kod zakończenia WebSphere MQ ustawiony za pomocą najnowszej metody lub zestawu właściwości wydanego dla dowolnego obiektu WebSphere MQ .

Jest resetowany do wywołania MQCC_OK, gdy metoda lub zestaw właściwości są pomyślnie wywoływane dla dowolnego obiektu MQAX.

Procedura obsługi zdarzeń błędów może sprawdzić tę właściwość w celu zdiagnozowania błędu, bez konieczności sprawdzania, który obiekt był zaangażowany.

Korzystanie z kodu CompletionCode i ReasonCode w obiekcie MQSession jest bardzo wygodne w przypadku prostych procedur obsługi błędów.

Uwaga: Sekcja “Wątki” na stronie 1067 zawiera ograniczenia dotyczące używania kodów błędów MQSession w aplikacjach wielowątkowych.

Zdefiniowane w: Klasa MQSession

Typ danych: Long

Wartości:

- MQCC_OK
- MQCC_WARNING,
- MQCC_FAILED

Składnia:

Aby uzyskać informacje: *completioncode & = MQSession*.CompletionCode

Właściwość ExceptionThreshold

Odczyt-zapis. Definiuje poziom błędu programu WebSphere MQ , dla którego MQAX zgłosi wyjątek. Wartość domyślna to MQCC_FAILED. Wartość większa niż MQCC_FAILED skutecznie zapobiega przetwarzaniu wyjątków, pozostawiając programistę w celu wykonania sprawdzenia kodu CompletionCode i ReasonCode.

Zdefiniowane w: Klasa MQSession

Typ danych: Long

Wartości:

- Dowolna, ale należy wziąć pod uwagę wartość MQCC_WARNING, MQCC_FAILED lub większą.

Składnia:

Aby uzyskać następujące informacje: *ExceptionThreshold& = MQSession*.**ExceptionThreshold**

Aby ustawić wartość: *MQSession*.**ExceptionThreshold** = *ExceptionThreshold$*

Właściwość ReasonCode

Tylko do odczytu. Zwraca kod przyczyny ustawiony przez najnowszą metodę lub zestaw właściwości wydany dla dowolnego obiektu WebSphere MQ .

Procedura obsługi zdarzeń błędów może sprawdzić tę właściwość w celu zdiagnozowania błędu, bez konieczności sprawdzania, który obiekt był zaangażowany.

Korzystanie z kodu CompletionCode i ReasonCode w obiekcie MQSession jest bardzo wygodne w przypadku prostych procedur obsługi błędów.

Uwaga: Sekcja “Wątki” na stronie 1067 zawiera ograniczenia dotyczące używania kodów błędów MQSession w aplikacjach wielowątkowych.

Zdefiniowane w: Klasa MQSession

Typ danych: Long

Wartości:

- Patrz sekcja Przyczyna (MQLONG) oraz dodatkowe wartości MQAX wymienione w sekcji “Kody przyczyny” na stronie 1137.

Składnia: do pobrania: *kod_przyczyny & = Sesja MQSession*.**ReasonCode**

Właściwość ReasonName

Tylko do odczytu. Zwraca nazwę symboliczną najnowszego kodu przyczyny. Na przykład: "MQRC_QMGR_NOT_AVAILABLE".

Uwaga: Sekcja “Wątki” na stronie 1067 zawiera ograniczenia dotyczące używania kodów błędów MQSession w aplikacjach wielowątkowych.

Zdefiniowane w: Klasa MQSession

Typ danych: String

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: aby uzyskać następujące informacje: *reasonname $= MQSession*.**ReasonName**

Metoda AccessGetMessageOptions

Tworzy nowy obiekt opcji MQGetMessage.

Zdefiniowane w: Klasa MQSession

Składnia: *gmo = MQSession*.**AccessGetMessageOptions()**

Metoda AccessMessage

Tworzy nowy obiekt MQMessage.

Zdefiniowane w: Klasa MQSession

Składnia: *msg = Sesja MQSession*.**AccessMessage()**

Metoda AccessPutMessageOptions

Tworzy nowy obiekt opcji MQPutMessage.

Zdefiniowane w: Klasa MQSession

Składnia: *pmo = MQSession*.**AccessPutMessageOptions()**

Metoda menedżera AccessQueue

Tworzy nowy obiekt MQQueueManager i łączy go z rzeczywistym menedżerem kolejek za pomocą klienta MQI produktu WebSphere MQ lub serwera WebSphere MQ . Podobnie jak w przypadku nawiązywania połączenia, ta metoda również wykonuje otwarte dla obiektu menedżera kolejek.

Jeśli zarówno klient WebSphere MQ MQI Client, jak i serwer WebSphere MQ są zainstalowane w systemie, aplikacje MQAX będą domyślnie uruchamiane dla serwera. Aby uruchomić program MQAX dla klienta, biblioteka powiązań klienta musi być określona w zmiennej środowiskowej `GMQ_MQ_LIB` , na przykład w polu `GMQ_MQ_LIB=mqic.dll`.

W przypadku instalacji tylko klienta nie jest konieczne ustawianie zmiennej środowiskowej `GMQ_MQ_LIB` . Jeśli ta zmienna nie zostanie ustawiona, program WebSphere MQ podejmie próbę załadowania pliku amqzst.dll. Jeśli ta biblioteka DLL nie jest obecna (jak ma to miejsce w przypadku instalacji tylko klienta), produkt WebSphere MQ próbuje załadować plik mqic.dll.

If successful it sets the MQQueueManager's ConnectionStatus to TRUE.

Menedżer kolejek może być podłączony do co najwyżej jednego obiektu MQQueueManager na instancję ActiveX .

Jeśli nawiązanie połączenia z menedżerem kolejek nie powiedzie się, zostanie zgłoszone zdarzenie błędu, a obiekt MQSession ma ustawioną wartość ReasonCode i CompletionCode .

Zdefiniowane w: Klasa MQSession

Składnia: *set qm = MQSession*.**AccessQueueManager** (*Name$*)

Parametr:*Name$* Łańcuch. Nazwa menedżera kolejek, z którym ma zostać nawiązane połączenie.

ClearError-metoda kodów

Resetuje parametr CompletionCode do wartości MQCC_OK i ReasonCode na wartość MQRC_NONE.

Zdefiniowane w: Klasa MQSession

Składnia: Call *MQSession*.**ClearErrorCodes ()**

Metoda ReasonCodeName

Zwraca nazwę kodu przyczyny o podanej wartości liczbowej. Przydatne jest nadanie użytkownikom jaśniejszych wskazówek dotyczących warunków błędów. Nazwa jest nadal nieco kryptograficzna (na przykład ReasonCodeName (2059) to **MQRC_Q_MGR_NOT_AVAILABLE**), więc tam, gdzie to możliwe, należy wychwytywać błędy i zastąpić je tekstem opisowym odpowiednim dla aplikacji.

Zdefiniowane w: Klasa MQSession

Składnia: *errname $= MQSession*.**ReasonCodeNazwa**(*reasonCode&*)

Parametr:*kod_przyczyny &* Long. Kod przyczyny, dla którego wymagana jest nazwa symboliczna.

Klasa MQQueueManager

Ta klasa reprezentuje połączenie z menedżerem kolejek. Menedżer kolejek może być uruchomiony lokalnie (serwer WebSphere MQ Server) lub zdalnie z dostępem udostępnionym przez klienta WebSphere MQ . Aplikacja musi utworzyć obiekt tej klasy i połączyć ją z menedżerem kolejek. Gdy obiekt tej klasy zostanie zniszczony, zostanie automatycznie odłączony od jego menedżera kolejek.

Zawieranie

Obiekty klasy MQQueue są powiązane z tą klasą.

Nowy obiekt tworzy nowy obiekt MQQueueManager i ustawia wszystkie właściwości na wartości początkowe. Alternatywnie można użyć metody menedżera AccessQueueklasy MQSession.

Tworzenie

Nowy obiekt tworzy obiekt **nowy** MQQueueManager i ustawia wszystkie właściwości na wartości początkowe. Alternatywnie można użyć metody menedżera AccessQueueklasy MQSession.

Składnia

Dim *mgr* **As New MQQueueManager set** *mgr* = **New MQQueueManager**

Właściwości

- “Właściwość ReasonCode” na stronie 1081.
- “Właściwość ReasonName” na stronie 1082.
- “Właściwość RemoteEvent” na stronie 1082.
- “Właściwość zdarzenia StartStop” na stronie 1082.
- “Właściwość Dostępność SyncPoint” na stronie 1082.
- “Właściwość TriggerInterval” na stronie 1082.

Metody

- “Metoda AccessQueue” na stronie 1083.
- “Metoda listy AddDistribution” na stronie 1083.
- “Metoda wycofania” na stronie 1084.
- “Metoda rozpoczęcia” na stronie 1084.
- “ClearError-metoda kodów” na stronie 1084.
- “Metoda zatwierdzania” na stronie 1084.
- “Metoda połączenia” na stronie 1084.
- “Metoda rozłączania” na stronie 1084.

Dostęp do właściwości

W dowolnym momencie można uzyskać dostęp do następujących właściwości.

- “Właściwość identyfikatora AlternateUser” na stronie 1075.
- “Właściwość CompletionCode” na stronie 1077.
- “Właściwość ConnectionStatus” na stronie 1078.
- “Właściwość ReasonCode” na stronie 1081.

Dostęp do pozostałych właściwości można uzyskać tylko wtedy, gdy obiekt jest połączony z menedżerem kolejek, a ID użytkownika jest uprawniony do sprawdzania się względem tego menedżera kolejek. Jeśli zostanie ustawiony alternatywny identyfikator użytkownika, a bieżący identyfikator użytkownika ma uprawnienia do jego używania, to zamiast tego zostanie sprawdzony alternatywny ID użytkownika w celu uzyskania autoryzacji.

Jeśli te warunki nie mają zastosowania, program WebSphere MQ Automation Classes for ActiveX próbuje nawiązać połączenie z menedżerem kolejek i otworzyć go, aby automatycznie uzyskać dostęp do zapytania. Jeśli ta operacja nie powiedzie się, wywołanie ustawia CompletionCode o wartości MQCC_FAILED i jeden z następujących elementów ReasonCodes:

- MQRC_CONNECTION_BROKEN
- MQRC_NOT_AUTHORIZED
- Błąd MQRC_Q_MGR_NAME_ERROR
- MQRC_Q_MGR_NOT_AVAILABLE

Właściwość identyfikatora AlternateUser

Odczyt-zapis. Alternatywny identyfikator użytkownika, który ma być używany do sprawdzania poprawności dostępu do atrybutów menedżera kolejek.

Ta właściwość nie może być ustawiona, jeśli właściwość IsConnected ma wartość TRUE.

Nie można ustawić tej właściwości w czasie, gdy obiekt jest otwarty.

Klasa **Defined in:** MQQueueManager

Data Type: Łańcuch o długości 12 znaków

Syntax: Aby uzyskać: *altuser $= MQQueueManager*.**AlternateUserId** , aby ustawić wartość: *MQQueueManager*.**AlternateUserId** = *altuser $*

Właściwość AuthorityEvent

Tylko do odczytu. Atrybut MQI AuthorityEvent .

Zdefiniowane w:
Klasa MQQueueManager

Typ danych:
Długa liczba całkowita

Wartości:

- MQEVR_DISABLED
- MQEVR_ENABLED

Składnia: Aby uzyskać informacje: *authevent = MQQueueManager*.**AuthorityEvent**

Właściwość BeginOptions

Odczyt-zapis. Są to opcje, które mają zastosowanie do metody Begin. Początkowo MQBO_NONE.

Zdefiniowane w:
Klasa MQQueueManager

Typ danych:
Długa liczba całkowita

Wartości:

- MQBO_NONE

Składnia: Aby uzyskać informacje: *beginoptions & =MQQueueManager*.**BeginOptions**

Aby ustawić: *MQQueueManager*.**BeginOptions**=*beginoptions &*

Właściwość definicji ChannelAuto

Tylko do odczytu. Określa, czy dozwolona jest automatyczna definicja kanału.

Zdefiniowane w:
Klasa MQQueueManager

Typ danych:
Długa liczba całkowita

Wartości:

- MQCHAD_DISABLED
- MQCHAD_ENABLED

Składnia: Aby uzyskać informacje: *channelautodef & = MQQueueManager*.**ChannelAutoDefinicja**

Właściwość ChannelAutoDefinitionEvent

Tylko do odczytu. Określa, czy generowane są zdarzenia automatycznej definicji kanału.

Zdefiniowane w:
Klasa MQQueueManager

Typ danych:
Długa liczba całkowita

Wartości:

- MQEVR_DISABLED
- MQEVR_ENABLED

Składnia: Aby uzyskać: *channelautodefevent & =MQQueueManager*.**ChannelAutoDefinitionEvent**

Właściwość ChannelAutoDefinitionExit

Tylko do odczytu. Nazwa wyjścia użytkownika używana dla definicji kanału automatycznego.

Zdefiniowane w:
Klasa MQQueueManager

Typ danych:
Łańcuch

Składnia: Aby uzyskać informacje: *channelautodefexit$= MQQueueManager*.**ChannelAutoDefinitionExit**

Właściwość CharacterSet

Tylko do odczytu. Atrybut MQI CodedCharSetId .

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Składnia: do pobrania: *characterset & = MQQueueManager***.CharacterSet**

Właściwość CloseOptions

Odczyt-zapis. Opcje używane do sterowania tym, co się dzieje, gdy menedżer kolejek jest zamknięty. Wartością początkową jest MQCO_NONE.

Zdefiniowane w:
Klasa MQQueueManager

Typ danych:
Długa liczba całkowita

Wartości:

- MQCO_NONE

Składnia: Aby uzyskać następujące informacje: *closeopt & = MQQueueManager***.CloseOptions**

Aby ustawić następujące elementy: *MQQueueManager***.CloseOptions** =*closeopt &*

Właściwość CommandInputQueueName

Tylko do odczytu. Atrybut QName CommandInputMQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Łańcuch o długości 48 znaków

Składnia: aby uzyskać następujące informacje: *commandinputqname $= MQQueueManager***.CommandInputQueueName**

Właściwość CommandLevel

Tylko do odczytu. Zwraca wersję i poziom implementacji menedżera kolejek produktu WebSphere MQ (atrybut MQI CommandLevel).

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Składnia: Aby uzyskać: *level & = MQQueueManager***.CommandLevel**

Właściwość CompletionCode

Tylko do odczytu. Zwraca kod zakończenia ustawiony przez ostatnią metodę lub dostęp do właściwości wystawiony dla obiektu.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Wartości:

- MQCC_OK
- MQCC_WARNING,
- MQCC_FAILED

Składnia: Aby uzyskać: *completioncode & = MQQueueManager*.**CompletionCode**

Właściwość ConnectionHandle

Tylko do odczytu. Uchwyt połączenia dla obiektu menedżera kolejek produktu WebSphere MQ .

Zdefiniowane w:
Klasa MQQueueManager

Typ danych:
Długa liczba całkowita

Składnia: Aby uzyskać: *hconn & = MQQueueManager.***ConnectionHandle**

Właściwość ConnectionStatus

Tylko do odczytu. Wskazuje, czy obiekt jest połączony z menedżerem kolejek, czy nie.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Wartość boolowska

Wartości:

- PRAWDA (-1)
- FALSE (0)

Składnia: Aby uzyskać: *status = MQQueueManager*.**ConnectionStatus**

Właściwość ConnectOptions

Odczyt-Zapis. Te opcje mają zastosowanie do metody Connect. Początkowo MQCNO_NONE.

Zdefiniowane w:
Klasa MQQueueManager

Typ danych:
Długa liczba całkowita

Wartości:

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING
- MQCNO_NONE

Składnia: do pobrania: *connectoptions & =MQQueueManager*.**ConnectOptions**

Aby ustawić następujące elementy: *MQQueueManager*.**ConnectOptions**=*connectoptions &*

Właściwość DeadLetterQueueName

Tylko do odczytu. Atrybut nazwy QName DeadLetterMQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Łańcuch o długości 48 znaków

Składnia: do pobrania: *dlqname $= MQQueueManager*.**DeadLetterQueueName**

Właściwość DefaultTransmissionQueueName

Tylko do odczytu. Atrybut nazwy QName DefXmitMQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Łańcuch o długości 48 znaków

Składnia: Aby uzyskać: *defxmitqname $= MQQueueManager*.**DefaultTransmissionQueueName**

Opis, właściwość

Tylko do odczytu. Atrybut QMgrDesc MQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Łańcuch o długości 64 znaków

Składnia: Aby uzyskać: *description $= MQQueueManager*.**Opis**

Właściwość DistributionLists

Tylko do odczytu. Jest to zdolność menedżera kolejek do obsługi list dystrybucyjnych.

Zdefiniowane w:
Klasa MQQueueManager

Typ danych:
wartość boolowska

Wartości:

- PRAWDA (-1)
- FALSE (0)

Składnia: Aby uzyskać informacje: *distributionlists= MQQueueManager*.**DistributionLists**

Właściwość InhibitEvent

Tylko do odczytu. Atrybut MQI InhibitEvent .

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Wartości:

- MQEVR_DISABLED
- MQEVR_ENABLED

Składnia: Aby uzyskać informacje: *inhibevent & = MQQueueManager*.**InhibitEvent**

Właściwość IsConnected

Wartość wskazująca, czy menedżer kolejek jest aktualnie połączony.

Tylko do odczytu.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Wartość boolowska

Wartości:

- PRAWDA (-1)
- FALSE (0)

Składnia: Aby uzyskać: *isconnected = MQQueueManager*.**IsConnected**

Właściwość IsOpen

Wartość wskazująca, czy menedżer kolejek jest aktualnie otwarty na potrzeby zapytania.

Tylko do odczytu.

Zdefiniowane w:
Klasa MQQueueManager

Typ danych:
wartość boolowska

Wartości:

- PRAWDA (-1)
- FALSE (0)

Składnia: Aby uzyskać: *IsOpen = MQQueueManager*.**IsOpen**

Właściwość LocalEvent

Tylko do odczytu. Atrybut MQI LocalEvent .

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Wartości:

- MQEVR_DISABLED
- MQEVR_ENABLED

Składnia: Aby uzyskać informacje: *localevent & = MQQueueManager*.**LocalEvent**

Właściwość MaximumHandles

Tylko do odczytu. Atrybut MaxHandles MQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Składnia: Aby uzyskać: *maxUchwyty & = MQQueueManager*.**MaximumHandles**

Właściwość Length MaximumMessage

Tylko do odczytu. Atrybut menedżera kolejek długości MaxMsgMQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Składnia: Aby uzyskać informacje: *maxmessagelength & = MQQueueManager*.**MaximumMessageLength**

Właściwość MaximumPriority

Tylko do odczytu. Atrybut MaxPriority MQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Składnia: Aby uzyskać: *maxpriority & = MQQueueManager*.**MaximumPriority**

Właściwość Komunikaty MaximumUncommitted

Tylko do odczytu. Atrybut MaxUncommittedkomunikatów MQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Składnia: Aby uzyskać następujące informacje: *maxuncommitted & = MQQueueManager*.**MaximumUncommittedKomunikaty**

Właściwość Nazwa

Odczyt-zapis. Atrybut MQI QMgrName . Tej właściwości nie można zapisać po nawiązaniu połączenia z programem MQQueueManager .

Zdefiniowane w: klasa MQQueueManager

Typ danych: Łańcuch o długości 48 znaków

Składnia: aby uzyskać: *name $= MQQueueManager*.**name**

Aby ustawić: *MQQueueManager*.**name** = *nazwa $*

Uwaga: Visual Basic rezerwuje właściwość "Name" na potrzeby użycia w interfejsie wizualnym. Dlatego w przypadku używania w języku Visual Basic użyj dolnego przypadku, tj. "name".

Właściwość ObjectHandle

Tylko do odczytu. Uchwyt obiektu dla obiektu menedżera kolejek produktu WebSphere MQ .

Zdefiniowane w:
Klasa MQQueueManager

Typ danych
Długa liczba całkowita

Składnia: Aby uzyskać następujące informacje: *hobj & = MQQueueManager.***ObjectHandle**

Właściwość PerformanceEvent

Tylko do odczytu. Atrybut MQI PerformanceEvent .

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Wartości:

- MQEVR_DISABLED
- MQEVR_ENABLED

Składnia: Aby uzyskać następujące informacje: *perfevent & = MQQueueManager*.PerformanceEvent

Właściwość platformy

Tylko do odczytu. Atrybut platformy MQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Wartości:

- MQPL_WINDOWS_NT
- MQPL_WINDOWS

Składnia: Aby uzyskać: *platforma & = MQQueueManager*.**Platforma**

Właściwość ReasonCode

Tylko do odczytu. Zwraca kod przyczyny ustawiony przez ostatnią metodę lub dostęp do właściwości wystawiony dla obiektu.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: do pobrania: *kod_przyczyny & = MQQueueManager*.**ReasonCode**

Właściwość ReasonName

Tylko do odczytu. Zwraca nazwę symboliczną najnowszego kodu przyczyny. Na przykład: "MQRC_QMGR_NOT_AVAILABLE".

Zdefiniowane w: klasa MQQueueManager

Typ danych: String

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: aby uzyskać następujące informacje: *reasonname $= MQQueueManager*.**ReasonName**

Właściwość RemoteEvent

Tylko do odczytu. Atrybut MQI RemoteEvent .

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Wartości:

- MQEVR_DISABLED
- MQEVR_ENABLED

Składnia: Aby uzyskać następujące informacje: *remoteevent & = MQQueueManager*.**RemoteEvent**

Właściwość zdarzenia StartStop

Tylko do odczytu. Atrybut zdarzenia StartStopMQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Wartości:

- MQEVR_DISABLED
- MQEVR_ENABLED

Składnia: Aby uzyskać informacje: *strstpevent & = MQQueueManager*.**StartStopEvent**

Właściwość Dostępność SyncPoint

Tylko do odczytu. Atrybut SyncPoint MQI.

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Wartości:

- MQSP_AVAILABLE
- MQSP_NOT_AVAILABLE

Składnia: Aby uzyskać informacje: *syncpointavailability & = MQQueueManager*.**SyncPointDostępność**

Właściwość TriggerInterval

Tylko do odczytu. Atrybut MQI TriggerInterval .

Zdefiniowane w: klasa MQQueueManager

Typ danych: Long

Składnia: Aby uzyskać następujące informacje: *trigint & = MQQueueManager*.**TriggerInterval**

Metoda AccessQueue

Tworzy obiekt MQQueue i wiąże go z tym obiektem MQQueueManager , ustawiając właściwość odwołania połączenia dla kolejki. Ustawia on właściwości Nazwa, OpenOptions, DynamicQueueName i AlternateUserobiektu MQQueue na podane wartości i próbuje je otworzyć.

Jeśli otwarcie nie powiedzie się, wywołanie nie powiedzie się. Zdarzenie błędu jest zgłaszane dla obiektu. Ustawione są wartości ReasonCode i CompletionCodeoraz MQSession ReasonCode i CompletionCode obiektu.

Parametry DynamicQueueName, QueueManagerName i AlternateUserId są opcjonalne i domyślne dla wartości "".

Należy określić opcję OpenOption MQOO_INQUIRE oprócz innych opcji, jeśli właściwości kolejki mają zostać odczytane.

Nie należy ustawiać nazwy QueueManagerani ustawić jej na wartość "", jeśli kolejka, która ma zostać otwarta, jest lokalna. W przeciwnym razie należy ustawić nazwę menedżera kolejek zdalnych, który jest właścicielem kolejki, a następnie podjąć próbę otwarcia lokalnej definicji kolejki zdalnej. Więcej informacji na temat rozstrzygania nazw kolejek zdalnych i aliasów menedżera kolejek zawiera sekcja Jakie są aliasy? .

Jeśli właściwość Nazwa jest ustawiona na nazwę kolejki modelowej, należy określić nazwę kolejki dynamicznej, która ma zostać utworzona w parametrze Name$ DynamicQueue$. Jeśli wartość podana w parametrze Name$ DynamicQueuema wartość "", to wartość ustawiona w obiekcie kolejki i używana w wywołaniu open to "AMQ. *". Więcej informacji na temat nazw kolejek dynamicznych zawiera sekcja “Tworzenie kolejek dynamicznych” na stronie 229 .

Definicja

Zdefiniowane w: klasa MQQueueManager .

Składnia

Składnia: set queue = MQQueueManager.**AccessQueue**(Name$, OpenOptions&, QueueManagerName$, DynamicQueueName$, AlternateUserId$)

Parametry

Nazwa$ Łańcuch. Nazwa kolejki produktu WebSphere MQ .

OpenOptions: Long. Opcje, które mają być używane, gdy kolejka jest otwarta. Patrz OpenOptions (MQLONG).

QueueManagerNazwa $ Łańcuch. Nazwa menedżera kolejek, który jest właścicielem kolejki, która ma zostać otwarta. Wartość "" oznacza, że menedżer kolejek jest lokalny.

*DynamicQueueNazwa$* String. Nazwa przypisana do kolejki dynamicznej w momencie otwarcia kolejki, gdy parametr Name$ określa kolejkę modelową.

AlternateUserId$ Łańcuch. Alternatywny identyfikator użytkownika używany do sprawdzania poprawności dostępu podczas otwierania kolejki.

Metoda listy AddDistribution

Tworzy nowy obiekt MQDistributionList i ustawia jego odwołanie do połączenia z menedżerem kolejek będącym właścicielem.

Zdefiniowane w:
Klasa MQQueueManager

Składnia: *set distributionlist* = **MQQueueManager.***AddDistributionList*

Metoda wycofania

Tworzy kopię zapasową wszystkich niezatwierdzonych operacji umieszczania i pobierania komunikatów, które wystąpiły jako część jednostki pracy od ostatniego punktu synchronizacji.

Zdefiniowane w: klasa MQQueueManager

Składnia: wywoływanie funkcji *MQQueueManager.***Backout ()**

Metoda rozpoczęcia

Rozpoczyna jednostkę pracy, która jest koordynowana przez menedżer kolejek. Opcje początku mają wpływ na zachowanie tej metody.

Zdefiniowane w:
Klasa MQQueueManager

Składnia: wywoływanie funkcji *MQQueueManager*.**Begin ()**

ClearError-metoda kodów

Resetuje parametr CompletionCode do wartości MQCC_OK i ReasonCode na wartość MQRC_NONE zarówno dla klasy MQQueueManager , jak i klasy MQSession.

Zdefiniowane w: klasa MQQueueManager

Składnia: Wywołanie *MQQueueManager***.ClearErrorCodes ()**

Metoda zatwierdzania

Zatwierdza wszelkie operacje umieszczania i pobierania komunikatów, które wystąpiły jako część jednostki pracy od ostatniego punktu synchronizacji.

Zdefiniowane w: klasa MQQueueManager

Składnia: Wywołanie *MQQueueManager***.Commit ()**

Metoda połączenia

Łączy obiekt MQQueueManager z rzeczywistym menedżerem kolejek za pośrednictwem klienta lub serwera WebSphere MQ MQI. Podobnie jak w przypadku nawiązywania połączenia, ta metoda również otwiera obiekt menedżera kolejek, aby można było go odpytywać.

Ustawia wartość parametru IsConnected na wartość TRUE.

Maksymalnie jeden obiekt MQQueueManager dla instancji ActiveX może łączyć się z menedżerem kolejek.

Zdefiniowane w: klasa MQQueueManager

Składnia: Wywołanie *MQQueueManager***.Connect ()**

Metoda rozłączania

Odłącza obiekt MQQueueManager od menedżera kolejek.

Ustawia wartość IsConnected na FALSE.

Wszystkie obiekty kolejki powiązane z obiektem MQQueueManager nie mogą być używane i nie można ich ponownie otworzyć.

Wszystkie niezatwierdzone zmiany (operacje umieszczania i pobierania komunikatów) są zatwierdzane.

Zdefiniowane w: klasa MQQueueManager

Składnia: Wywołanie *MQQueueManager*.**Rozłącz ()**

Klasa MQQueue

Ta klasa reprezentuje dostęp do kolejki produktu WebSphere MQ . To połączenie jest udostępniane przez powiązany obiekt MQQueueManager . Gdy obiekt tej klasy zostanie zniszczony, zostanie automatycznie zamknięty.

Zawieranie

Klasa MQQueue jest zawarta w klasie MQQueueManager .

Tworzenie

Produkt New tworzy nowy obiekt MQQueue i ustawia wszystkie właściwości na wartości początkowe. Alternatywnie można użyć metody AccessQueue klasy MQQueueManager .

Składnia

```
Dim que As New MQQueue Set que  = New MQQueue
```

Właściwości

- “Właściwość identyfikatora AlternateUser” na stronie 1087.
- “Właściwość Nazwa BackoutRequeue” na stronie 1088.
- “Właściwość BackoutThreshold” na stronie 1088.
- “Właściwość Nazwa BaseQueue” na stronie 1088.
- “Właściwość CloseOptions” na stronie 1088.
- “Właściwość CompletionCode” na stronie 1088.
- “Właściwość ConnectionReference” na stronie 1089.
- “Właściwość czasu CreationDate” na stronie 1089.
- “Właściwość CurrentDepth” na stronie 1089.
- “Właściwość DefaultInputOpenOption” na stronie 1089.
- “Właściwość DefaultPersistence” na stronie 1089.
- “Właściwość DefaultPriority” na stronie 1089.
- “Właściwość DefinitionType” na stronie 1090.
- “Właściwość zdarzenia DepthHigh” na stronie 1090.
- “Właściwość DepthHighLimit” na stronie 1090.
- “Właściwość zdarzenia DepthLow” na stronie 1090.
- “DepthLow-właściwość Limit” na stronie 1090.
- “Właściwość zdarzenia DepthMaximum” na stronie 1091.
- “Właściwość zdarzenia DepthHigh” na stronie 1090.
- “Właściwość DepthHighLimit” na stronie 1090.
- “Właściwość zdarzenia DepthLow” na stronie 1090.
- “DepthLow-właściwość Limit” na stronie 1090.
- “Właściwość zdarzenia DepthMaximum” na stronie 1091.
- “Opis, właściwość” na stronie 1091.
- “Właściwość Nazwa DynamicQueue” na stronie 1091.
- “HardenGet, właściwość Backout” na stronie 1091.

- “Właściwość InhibitGet” na stronie 1091.
- “Właściwość InhibitPut” na stronie 1092.
- “Właściwość Nazwa InitiationQueue” na stronie 1092.
- “Właściwość IsOpen” na stronie 1092.
- “Właściwość MaximumDepth” na stronie 1092.
- “Właściwość Length MaximumMessage” na stronie 1092.
- “Właściwość sekwencji MessageDelivery” na stronie 1093.
- “Właściwość ObjectHandle” na stronie 1093.
- “Właściwość Liczba OpenInput” na stronie 1093.
- “Właściwość OpenOptions” na stronie 1093.
- “Właściwość Liczba OpenOutput” na stronie 1093.
- “Właściwość OpenStatus” na stronie 1094.
- “Właściwość ProcessName” na stronie 1094.
- “Właściwość Nazwa QueueManager” na stronie 1094.
- “Właściwość QueueType” na stronie 1094.
- “Właściwość ReasonCode” na stronie 1094.
- “Właściwość ReasonName” na stronie 1095.
- “Właściwość RemoteQueueManagerName” na stronie 1095.
- “Właściwość Nazwa RemoteQueueName” na stronie 1095.
- “Właściwość ResolvedQueueManagerName” na stronie 1095.
- “Właściwość Nazwa ResolvedQueue” na stronie 1095.
- “Właściwość RetentionInterval” na stronie 1095.
- “Właściwość zasięgu” na stronie 1096.
- “Właściwość ServiceInterval” na stronie 1096.
- “Właściwość zdarzenia ServiceIntervalEvent” na stronie 1096.
- “Właściwość współużytkowalności” na stronie 1096.
- “Właściwość Nazwa kolejki TransmissionQueue” na stronie 1096.
- “Właściwość TriggerControl” na stronie 1097.
- “Właściwość TriggerData” na stronie 1097.
- “Właściwość TriggerDepth” na stronie 1097.
- “Właściwość priorytetu TriggerMessage” na stronie 1097.
- “Właściwość TriggerType” na stronie 1097.
- “właściwość użycia” na stronie 1098.

Metody

- “ClearError-metoda kodów” na stronie 1098
- “Metoda zamknięcia” na stronie 1098
- “metoda GET” na stronie 1098
- “Metoda otwierania” na stronie 1099
- “metoda PUT” na stronie 1099

Dostęp do właściwości

Jeśli obiekt kolejki nie jest połączony z menedżerem kolejek, można zapoznać się z następującymi właściwościami:

- “Właściwość CompletionCode” na stronie 1088
- “Właściwość OpenStatus” na stronie 1094
- “Właściwość ReasonCode” na stronie 1094

i można odczytywać i zapisywać do:

- “Właściwość identyfikatora AlternateUser” na stronie 1087
- “Właściwość CloseOptions” na stronie 1088
- “Właściwość ConnectionReference” na stronie 1089
- “Właściwość Nazwa” na stronie 1093
- “Właściwość OpenOptions” na stronie 1093

Jeśli obiekt kolejki jest połączony z menedżerem kolejek, można odczytać wszystkie właściwości.

Właściwości atrybutu kolejki

Właściwości, które nie zostały wymienione w poprzedniej sekcji, to wszystkie atrybuty bazowej kolejki produktu WebSphere MQ . Dostęp do nich można uzyskać tylko wtedy, gdy obiekt jest połączony z menedżerem kolejek, a ID użytkownika jest autoryzowany do uzyskiwania informacji lub ustawiania dla tej kolejki. Jeśli zostanie ustawiony alternatywny identyfikator użytkownika, a bieżący identyfikator użytkownika ma uprawnienia do jego używania, to zamiast niego zostanie sprawdzony alternatywny ID użytkownika.

Właściwość musi być odpowiednią właściwością dla danego typu QueueType. Więcej informacji na ten temat zawiera sekcja Atrybuty kolejek .

Jeśli te warunki nie mają zastosowania, dostęp do właściwości ustawi obiekt CompletionCode o wartości MQCC_FAILED i jeden z następujących elementów ReasonCodes:

- MQRC_CONNECTION_BROKEN
- MQRC_NOT_AUTHORIZED
- Błąd MQRC_Q_MGR_NAME_ERROR
- MQRC_Q_MGR_NOT_CONNECTED
- MQRC_SELECTOR_NOT_FOR_TYPE (CompletionCode to MQCC_WARNING)

Otwieranie kolejki

Jedynym sposobem utworzenia obiektu MQQueue jest użycie metody MQQueueManager AccessQueue lub nowego obiektu. Otwarty obiekt MQQueue pozostaje otwarty (OpenStatus= TRUE), dopóki nie zostanie zamknięty lub usunięty lub dopóki obiekt menedżera kolejek nie zostanie usunięty lub połączenie zostanie utracone do menedżera kolejek. Wartość właściwości CloseOptions kolejki MQQueue steruje zachowaniem operacji zamykania, która ma miejsce, gdy obiekt MQQueue zostanie usunięty.

Metoda MQQueueManager AccessQueue otwiera kolejkę za pomocą parametru OpenOptions . Metoda MQQueue.Open otwiera kolejkę przy użyciu właściwości OpenOptions . Produkt WebSphere MQ sprawdza poprawność OpenOptions dla autoryzacji użytkownika w ramach procesu otwartego kolejki.

Właściwość identyfikatora AlternateUser

Odczyt-zapis. Alternatywny identyfikator użytkownika używany do sprawdzania poprawności dostępu do kolejki po jego otwarciu.

Nie można ustawić tej właściwości w czasie, gdy obiekt jest otwarty (to znaczy, gdy parametr IsOpen ma wartość TRUE).

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 12 znaków

Składnia: Aby uzyskać: *altuser $= MQQueue*.**AlternateUserId**

Aby ustawić wartość: *MQQueue.***AlternateUserId** = *altuser $*

Właściwość Nazwa BackoutRequeue

Tylko do odczytu. Atrybut BackOutRequeueQName MQI.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Składnia: Aby uzyskać: *backoutrequeuename $= MQQueue***.BackoutRequeueNazwa**

Właściwość BackoutThreshold

Tylko do odczytu. Atrybut BackoutThreshold MQI.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- Patrz BackoutThreshold (MQLONG).

Składnia: Aby uzyskać: *backoutthreshold & = MQQueue.***BackoutThreshold**

Właściwość Nazwa BaseQueue

Tylko do odczytu. Nazwa kolejki, do której alias jest tłumaczona.

Poprawna tylko dla kolejek aliasowych.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Składnia: aby uzyskać: *baseqname $= MQQueue***.BaseQueueName**

Właściwość CloseOptions

Odczyt-Zapis. Opcje używane do sterowania tym, co się dzieje, gdy kolejka jest zamknięta.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQCO_NONE
- MQCO_DELETE
- MQCO_DELETE_PURGE

Komendy MQCO_DELETE i MQCO_DELETE_PURGE są poprawne tylko dla kolejek dynamicznych.

Składnia: Aby uzyskać następujące informacje: *closeopt & = MQQueue***.CloseOptions**

Aby ustawić: *MQQueue***.CloseOptions** = *closeopt &*

Właściwość CompletionCode

Tylko do odczytu. Zwraca kod zakończenia ustawiony przez ostatnią metodę lub dostęp do właściwości wystawiony dla obiektu.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQCC_OK

- MQCC_WARNING,
- MQCC_FAILED

Składnia: Aby uzyskać: *completioncode & = MQQueue***.CompletionCode**

Właściwość ConnectionReference

Odczyt-zapis. Definiuje obiekt menedżera kolejek, do którego należy obiekt kolejki. Nie można zapisać odwołania do połączenia w czasie, gdy kolejka jest otwarta.

Zdefiniowane w: Klasa MQQueue

Typ danych: MQQueueManager

Wartości:

- Odwołanie do aktywnego obiektu menedżera kolejek produktu WebSphere MQ

Składnia: aby ustawić: *set MQQueue***.ConnectionReference** = *ConnectionReference*

Aby uzyskać następujące informacje: *set ConnectionReference = MQQueue***.ConnectionReference**

Właściwość czasu CreationDate

Tylko do odczytu. Data i godzina utworzenia tej kolejki.

Zdefiniowane w: Klasa MQQueue

Typ danych: Variant typu 7 (data/godzina) lub EMPTY

Składnia: Aby uzyskać: *datetime = MQQueue***.CreationDate**

Właściwość CurrentDepth

Tylko do odczytu. Liczba komunikatów znajdujących się obecnie w kolejce.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: aby uzyskać następujące informacje: *currentdepth & = MQQueue***.CurrentDepth**

Właściwość DefaultInputOpenOption

Tylko do odczytu. Określa sposób otwierania kolejki, jeśli opcja OpenOptions określa wartość MQOO_INPUT_AS_Q_DEF.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_SHARED

Składnia: Aby uzyskać następujące informacje: *defaultinop & = MQQueue***.DefaultInputOpenOption**

Właściwość DefaultPersistence

Tylko do odczytu. Domyślna trwałość komunikatów w kolejce.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać następujące informacje: *defpersistence & = MQQueue***.DefaultPersistence**

Właściwość DefaultPriority

Tylko do odczytu. Domyślny priorytet komunikatów w kolejce.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać następujący element: *defpriority & = MQQueue*.**DefaultPriority**

Właściwość DefinitionType

Tylko do odczytu. Typ definicji kolejki.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQQDT_PREDEFINIOWANE
- MQQDT_PERMANENT_DYNAMIC
- MQQDT_TEMPORARY_DYNAMIC

Składnia: do pobrania: *deftype & = MQQueue*.**DefinitionType**

Właściwość zdarzenia DepthHigh

Tylko do odczytu. Atrybut zdarzenia QDepthHighMQI.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQEVR_DISABLED
- MQEVR_ENABLED

Składnia: Aby uzyskać informacje: *depthhighevent & = MQQueue.*.**DepthHighEvent**

Właściwość DepthHighLimit

Tylko do odczytu. Atrybut Limit MQI QDepthHigh.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać informacje: *depthhighlimit & = MQQueue.*.**DepthHighLimit**

Właściwość zdarzenia DepthLow

Tylko do odczytu. Atrybut zdarzenia QDepthLowMQI.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQEVR_DISABLED
- MQEVR_ENABLED

Składnia: aby uzyskać informacje: *depthlowevent & = MQQueue.*.**DepthLowEvent**

DepthLow-właściwość Limit

Tylko do odczytu. Atrybut Limit MQI QDepthLow.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać informacje: *depthlowlimit & = MQQueue.***DepthLowLimit**

Właściwość zdarzenia DepthMaximum

Tylko do odczytu. Atrybut zdarzenia QDepthMaxMQI.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQEVR_DISABLED
- MQEVR_ENABLED

Składnia: Aby uzyskać informacje: *depthmaximumevent & = MQQueue.***DepthMaximumEvent**

Opis, właściwość

Tylko do odczytu. Opis kolejki.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 64 znaków

Składnia: Aby uzyskać następujący tekst: *description $= MQQueue***.Opis**

Właściwość Nazwa DynamicQueue

Odczyt-zapis, tylko do odczytu, gdy kolejka jest otwarta.

Ta opcja steruje nazwą kolejki dynamicznej używanej podczas otwierania kolejki modelowej. Można go ustawić za pomocą znaku wieloznacznego przez użytkownika jako zestaw właściwości (tylko wtedy, gdy kolejka jest zamknięta) lub jako parametr w parametrze MQQueueManager.AccessQueue().

Rzeczywista nazwa kolejki dynamicznej znajduje się w zapytaniu QueueName.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Wartości:

- Dowolna poprawna nazwa kolejki produktu WebSphere MQ .

Składnia: do ustawienia: *MQQueue***.DynamicQueueName** = *dynamicqueuename $*

Aby uzyskać następujące informacje: *dynamicqueuename $ = MQQueue***.DynamicQueueName**

HardenGet, właściwość Backout

Tylko do odczytu. Określa, czy zachować dokładną liczbę wycofanych danych.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQQA_BACKOUT_HARTOWANA
- MQQA_BACKOUT_NOT HARTOWANE

Składnia: Aby uzyskać informacje: *hardengetback & = MQQueue***.HardenGetBackout**

Właściwość InhibitGet

Odczyt-zapis. Atrybut InhibitGet MQI.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQQA_GET_INHIBITED
- MQQA_GET_ALLOWED

Składnia: Aby uzyskać: *getstatus & = MQQueue*.**InhibitGet**

Aby ustawić: *MQQueue*.**InhibitGet** = *getstatus &*

Właściwość InhibitPut

Odczyt-zapis. Atrybut MQI InhibitPut .

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQQA_PUT_INHIBITED
- MQQA_PUT_ALLOWED

Składnia: Aby uzyskać: *putstatus & = MQQueue*.**InhibitPut**

Aby ustawić: *MQQueue*.**InhibitPut** = *putstatus &*

Właściwość Nazwa InitiationQueue

Tylko do odczytu. Nazwa kolejki inicjuj.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Składnia: Aby uzyskać: *initqname $= MQQueue*.**InitiationQueueNazwa**

Właściwość IsOpen

Zwraca informację o tym, czy kolejka jest otwarta.

Tylko do odczytu.

Zdefiniowane w: Klasa MQQueue

Typ danych: Wartość boolowska

Wartości:

- PRAWDA (-1)
- FALSE (0)

Składnia: Aby uzyskać: *open = MQQueue*.**IsOpen**

Właściwość MaximumDepth

Tylko do odczytu. Maksymalna głębokość kolejki.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać następujące informacje: *maxdepth & = MQQueue*.**MaximumDepth**

Właściwość Length MaximumMessage

Tylko do odczytu. Maksymalna dozwolona długość komunikatu w bajtach dla tej kolejki.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać informacje: *maxmlength & = MQQueue***.MaximumMessageLength**

Właściwość sekwencji MessageDelivery

Tylko do odczytu. Kolejność dostarczania komunikatów.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQMDS_PRIORITY,
- MQMDS_FIFO

Składnia: Aby uzyskać: *messdelseq & = MQQueue***.MessageDelivery, kolejność**

Właściwość Nazwa

Odczyt-zapis. Atrybut kolejki MQI. Ta właściwość nie może zostać zapisana po otwarciu kolejki MQQueue.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Składnia: aby uzyskać: *name $= MQQueue***.name**

Aby ustawić: *MQQueue***.name** = *nazwa $*

Uwaga: Visual Basic rezerwuje właściwość "Name" na potrzeby użycia w interfejsie wizualnym. Dlatego w przypadku używania w języku Visual Basic użyj dolnego przypadku, tj. "name".

Właściwość ObjectHandle

Tylko do odczytu. Uchwyt obiektu dla obiektu kolejki WebSphere MQ .

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać: *hobj & = MQQueue.***ObjectHandle**

Właściwość Liczba OpenInput

Tylko do odczytu. Liczba operacji otwierania dla danych wejściowych.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać informacje: *openincount & = MQQueue***.OpenInputCount**

Właściwość OpenOptions

Odczyt-zapis. Opcje, które mają być używane do otwierania kolejki.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- Patrz OpenOptions (MQLONG).

Składnia: Aby uzyskać: *openopt & = MQQueue***.OpenOptions**

Aby ustawić wartość: *MQQueue.***OpenOptions** = *openopt &*

Właściwość Liczba OpenOutput

Tylko do odczytu. Liczba operacji otwierania dla danych wyjściowych.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać: *openoutcount & = MQQueue*.**OpenOutputCount**

Właściwość OpenStatus

Tylko do odczytu. Wskazuje, czy kolejka jest otwarta, czy nie. Wartością początkową jest TRUE po metodzie AccessQueue lub wartość FALSE po Nowym.

Zdefiniowane w: Klasa MQQueue

Typ danych: Wartość boolowska

Wartości:

- PRAWDA (-1)
- FALSE (0)

Składnia: Aby uzyskać: *status & = Kolejka MQQueue*.**OpenStatus**

Właściwość ProcessName

Tylko do odczytu. Atrybut ProcessName MQI.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Składnia: Do pobrania: *nazwa_proc_$ =Kolejka MQQueue*.**ProcessName**

Właściwość Nazwa QueueManager

Odczyt-zapis. Nazwa menedżera kolejek produktu WebSphere MQ .

Zdefiniowane w: Klasa MQQueue

Typ danych: String

Składnia: Aby uzyskać: *QueueManagerNazwa$ = MQQueue*.**QueueManagerNazwa**

Aby ustawić: *MQQueue*.**QueueManagerName** = *QueueManagerName$*

Właściwość QueueType

Tylko do odczytu. Atrybut QType MQI.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQQT_ALIAS
- MQQT_LOCAL
- MODEL MQQT_MODEL
- MQQT_REMOTE

Składnia: Aby uzyskać: *queuetype & = MQQueue*.**QueueType**

Właściwość ReasonCode

Tylko do odczytu. Zwraca kod przyczyny ustawiony przez ostatnią metodę lub dostęp do właściwości wystawiony dla obiektu.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: do pobrania: *kod_przyczyny & = Kolejka MQQueue*.**ReasonCode**

Właściwość ReasonName

Tylko do odczytu. Zwraca nazwę symboliczną najnowszego kodu przyczyny. Na przykład: "MQRC_QMGR_NOT_AVAILABLE".

Zdefiniowane w: Klasa MQQueue

Typ danych: String

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: aby uzyskać następujące informacje: *reasonname $= MQQueue*.**ReasonName**

Właściwość RemoteQueueManagerName

Tylko do odczytu. Nazwa zdalnego menedżera kolejek. Poprawna tylko w przypadku kolejek zdalnych.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Składnia: Aby uzyskać: *remqmanname $= MQQueue*.**RemoteQueueManagerName**

Właściwość Nazwa RemoteQueueName

Tylko do odczytu. Nazwa kolejki, o której wiadomo, że jest ona znana w zdalnym menedżerze kolejek. Poprawna tylko w przypadku kolejek zdalnych.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Składnia: Aby uzyskać: *remqname $= MQQueue*.**RemoteQueueName**

Właściwość ResolvedQueueManagerName

Tylko do odczytu. Nazwa docelowego menedżera kolejek, który jest znany menedżerowi kolejek lokalnych.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Składnia: aby uzyskać następujące informacje: *resqmanname $= MQQueue*.**ResolvedQueueManagerName**

Właściwość Nazwa ResolvedQueue

Tylko do odczytu. Nazwa docelowej kolejki docelowej, która jest znana menedżerowi kolejek lokalnych.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Składnia: Aby uzyskać: *resqname $= MQQueue*.**ResolvedQueueNazwa**

Właściwość RetentionInterval

Tylko do odczytu. Okres, w którym kolejka powinna zostać zachowana.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: do pobrania: *retinterval & = MQQueue*.**RetentionInterval**

Właściwość zasięgu

Tylko do odczytu. Określa, czy pozycja dla tej kolejki istnieje również w katalogu komórki.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQSCO_Q_MGR
- Komórka MQSCO_CELL

Składnia: do pobrania: *scope & = MQQueue*.**Scope**

Właściwość ServiceInterval

Tylko do odczytu. Atrybut MQI QServiceInterval .

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać: *serviceinterval & = MQQueue.***ServiceInterval**

Właściwość zdarzenia ServiceIntervalEvent

Tylko do odczytu. Atrybut zdarzenia MQI QServiceInterval.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQQSIE_WYSOKI
- MQQSIE_OK
- MQQSIE_NONE

Składnia: Aby uzyskać: *serviceintervalevent & = MQQueue.***ServiceIntervalZdarzenie**

Właściwość współużytkowalności

Tylko do odczytu. Współużytkowalność kolejki.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQQA_SHAREABLE
- MQQA_NOT_SHAREABLE

Składnia: Aby uzyskać następujące informacje: *shareability & = MQQueue*.**Shareability**

Właściwość Nazwa kolejki TransmissionQueue

Tylko do odczytu. Nazwa kolejki transmisji. Poprawna tylko w przypadku kolejek zdalnych.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 48 znaków

Składnia: aby uzyskać: *transqname $= MQQueue*.**TransmissionQueueName**

Właściwość TriggerControl

Odczyt-zapis. Sterowanie wyzwalaczem.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQTC_OFF
- MQTC_ON

Składnia: Aby uzyskać następujące informacje: *trigcontrol & = MQQueue***.TriggerControl**

Aby ustawić: *MQQueue***.TriggerControl** = *trigcontrol &*

Właściwość TriggerData

Odczyt-zapis. Dane wyzwalacza.

Zdefiniowane w: Klasa MQQueue

Typ danych: Łańcuch o długości 64 znaków

Składnia: Aby uzyskać: *trigdata $= MQQueue***.TriggerData**

Aby ustawić: *MQQueue***.TriggerData** = *trigdata $*

Właściwość TriggerDepth

Odczyt-zapis. Liczba komunikatów, które muszą znajdować się w kolejce, zanim zostanie zapisany komunikat wyzwalacza.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać: *trigdepth & = MQQueue***.TriggerDepth**

Aby ustawić: *MQQueue***.TriggerDepth** = *trigdepth &*

Właściwość priorytetu TriggerMessage

Odczyt-zapis. Priorytet komunikatu progowego dla wyzwalaczy.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Składnia: Aby uzyskać następujące informacje: *trigmesspriority & = MQQueue***.TriggerMessagePriority**

Aby ustawić wartość: *MQQueue***.TriggerMessagePriority** = *trigmesspriority &*

Właściwość TriggerType

Odczyt-zapis. Typ wyzwalacza.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQTT_NONE
- MQTT_FIRST
- MQTT_EVERY
- MQTT_DEPTH

Składnia: Aby uzyskać: *trigtype & = MQQueue***.TriggerType**

Aby ustawić wartość: *MQQueue*.**TriggerType** = *Trigtype &*

właściwość użycia

Tylko do odczytu. Wskazuje, dla której jest używana kolejka.

Zdefiniowane w: Klasa MQQueue

Typ danych: Long

Wartości:

- MQUS_NORMAL
- MQUS_TRANSMISSION

Składnia: Aby uzyskać: *usage & = MQQueue*.**Użycie**

ClearError-metoda kodów

Resetuje parametr CompletionCode do wartości MQCC_OK i ReasonCode na wartość MQRC_NONE zarówno dla klasy MQQueue, jak i klasy MQSession.

Zdefiniowane w: Klasa MQQueue

Składnia: Call *MQQueue*.**ClearErrorCodes ()**

Metoda zamknięcia

Zamyka kolejkę przy użyciu bieżących wartości CloseOptions.

Zdefiniowane w: Klasa MQQueue

Składnia: wywołaj komendę *MQQueue*.**Zamknij ()**

metoda GET

Pobiera komunikat z kolejki.

Ta metoda pobiera obiekt MQMessage jako parametr, korzystając z niektórych pól w strukturze MQMD obiektu jako parametry wejściowe. W szczególności używane są pola MessageId i CorrelId , dlatego ważne jest, aby upewnić się, że pola te są ustawione zgodnie z wymaganiami. Więcej informacji na temat tych pól można znaleźć w sekcji MsgId (MQBYTE24) i CorrelId (MQBYTE24) .

Jeśli metoda zakończy się niepowodzeniem, obiekt MQMessage nie zostanie zmieniony. Jeśli powiedzie się, porcje MQMD i Message Data w obiekcie MQMessage są zastępowane przez dane MQMD i Message Data z komunikatu przychodzącego. Właściwości elementu sterującego MQMessage są ustawiane w następujący sposób:

- Parametr **MessageLength** jest ustawiony na długość komunikatu WebSphere MQ .
- **DataLength** jest ustawiona na długość komunikatu WebSphere MQ .
- **DataOffset** jest ustawione na zero

Zdefiniowane w:
Klasa MQQueue

Składnia: wywołaj komendę *MQQueue*.**Get**(*Komunikat, Opcje GetMsg, Długość GetMsg*)

Parametry
Komunikat:

Obiekt MQMessage reprezentujący komunikat, który ma zostać pobrany.

Opcje GetMsg:

Opcjonalny obiekt opcji MQGetMessage, który umożliwia sterowanie operacją pobierania. Jeśli ten parametr nie zostanie określony, zostaną użyte domyślne opcje MQGetMessage.

GetMsgDługość:

Opcjonalna wartość 2-lub 4-bajtowa, która umożliwia sterowanie maksymalną długością komunikatu WebSphere MQ pobranego z kolejki.

Jeśli zostanie podana opcja MQGMO_ACCEPT_TRUNCATED_MSG, operacja GET zakończy się pomyślnie z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_TRUNCATED_MSG_ACCEPTED, jeśli wielkość komunikatu przekracza podaną długość.

Obiekt MessageData zawiera pierwsze bajty GetMsg(GetMsg).

Jeśli określono wartość MQGMO_ACCEPT_TRUNCATED_MSG ***nie*** , a wielkość komunikatu przekracza określoną długość, zwracany jest kod zakończenia MQCC_FAILED razem z kodem przyczyny MQRC_TRUNCATED_MESSAGE_FAILED.

Jeśli zawartość buforu komunikatów jest niezdefiniowana, całkowita długość komunikatu jest ustawiana na pełną długość komunikatu, który zostałby pobrany.

Jeśli parametr długości komunikatu nie zostanie określony, długość buforu komunikatów zostanie automatycznie dopasowana do co najmniej wielkości komunikatu przychodzącego.

Metoda otwierania

Otwiera kolejkę przy użyciu bieżących wartości:

1. QueueName
2. QueueManagerName
3. Identyfikator AlternateUser
4. Nazwa DynamicQueue

Zdefiniowane w:
Klasa MQQueue

Składnia: wywołaj komendę *MQQueue***.Otwórz ()**

metoda PUT

Umieszcza komunikat w kolejce.

Ta metoda przyjmuje obiekt MQMessage jako parametr. Właściwości deskryptora komunikatu (MQMD) tego obiektu mogą zostać zmienione w wyniku tej metody. Wartości, które mają natychmiast po uruchomieniu tej metody, to wartości, które zostały wprowadzone w produkcie WebSphere MQ.

Modyfikacje obiektu MQMessage po zakończeniu wprowadzania nie mają wpływu na rzeczywisty komunikat w kolejce produktu WebSphere MQ .

Zdefiniowane w:
Klasa MQQueue

Składnia: wywołaj komendę *MQQueue***.Put**(*Komunikat, Opcje PutMsg)*

Parametry
Komunikat

Obiekt MQMessage reprezentujący komunikat, który ma zostać umieszczony.

Opcje PutMsg

MQPutMessageObiekt opcji zawierający opcje służące do sterowania operacją put. Jeśli te wartości nie zostaną określone, zostaną użyte domyślne opcje PutMessage.

Klasa MQMessage

Ta klasa reprezentuje komunikat WebSphere MQ . Zawiera ona właściwości służące do hermetyzowania deskryptora komunikatu produktu WebSphere MQ (MQMD) i udostępnia bufor do przechowywania danych komunikatu zdefiniowanych przez aplikację.

Klasa zawiera metody zapisu służące do kopiowania danych z aplikacji ActiveX do obiektu MQMessage. Podobnie klasa zawiera metody odczytu służące do kopiowania danych z obiektu MQMessage do aplikacji ActiveX . Klasa zarządza przydzielaniem i rozdzielaniem pamięci dla buforu automatycznie. Aplikacja nie musi deklarować wielkości buforu, gdy tworzony jest obiekt MQMessage, ponieważ bufor rośnie, aby pomieścić zapisywane do niego dane.

Nie można umieścić komunikatu w kolejce produktu WebSphere MQ , jeśli wielkość buforu przekracza właściwość MaximumMessageLength tej kolejki.

Po zbudowaniu obiekt MQMessage może zostać umieszczony w kolejce produktu WebSphere MQ za pomocą metody MQQueue.Put . Ta metoda pobiera kopię części obiektu MQMD i danych komunikatu obiektu i umieszcza je w kolejce. Aplikacja może w związku z tym zmodyfikować lub usunąć obiekt MQMessage po umieszczonym w nim umieszczonym komunikacie, bez wpływu na komunikat w kolejce produktu WebSphere MQ . Menedżer kolejek może dopasować niektóre pola w strukturze MQMD, gdy kopiuje komunikat w kolejce WebSphere MQ .

Komunikat przychodzący może zostać odczytany w obiekcie MQMessage za pomocą metody MQQueue.Get . Spowoduje to zastąpienie wszystkich danych MQMD lub danych komunikatu, które mogły już być w obiekcie MQMessage, wartościami z komunikatu przychodzącego. Dopasowuje wielkość buforu danych obiektu MQMessage, tak aby była zgodna z wielkością danych komunikatu przychodzącego.

Zawieranie

Komunikaty są zawarte w klasie MQSession.

Tworzenie

Nowy -tworzy obiekt MQMessage. Jego właściwości deskryptora komunikatu są początkowo ustawiane na wartości domyślne, a bufor danych komunikatu jest pusty.

Składnia

```
Dim msg As New MQMessage or Set msg = New MQMessage
```

Właściwości

Właściwości elementu sterującego to:

- “Właściwość CompletionCode” na stronie 1102
- “Właściwość DataLength” na stronie 1103
- “Właściwość DataOffset” na stronie 1103
- “Właściwość MessageLength” na stronie 1103
- “Właściwość ReasonCode” na stronie 1104
- “Właściwość ReasonName” na stronie 1104

Właściwości deskryptora komunikatu to:

- “Właściwość AccountingToken” na stronie 1104
- “Właściwość Hex AccountingToken” na stronie 1104
- “Właściwość danych ApplicationId” na stronie 1104
- “Właściwość danych ApplicationOrigin” na stronie 1105
- “Właściwość BackoutCount” na stronie 1105
- “Właściwość CharacterSet” na stronie 1105
- “Właściwość CorrelationId” na stronie 1106
- “Właściwość CorrelationIdHex” na stronie 1106
- “Właściwość kodowania” na stronie 1106

Metody

- “Metoda ReadUTF” na stronie 1117
- “Metoda ResizeBuffer” na stronie 1117
- “Metoda zapisu” na stronie 1117
- “Metoda WriteBoolean” na stronie 1117
- “Metoda WriteByte” na stronie 1118
- “Metoda WriteDecimal2” na stronie 1118
- “Metoda WriteDecimal4” na stronie 1118
- “Metoda WriteDouble” na stronie 1118
- “Metoda WriteDouble4” na stronie 1119
- “Metoda WriteFloat” na stronie 1119
- “Metoda WriteInt2” na stronie 1119
- “Metoda WriteInt4” na stronie 1119
- “Metoda WriteLong” na stronie 1119
- “Metoda WriteNullTerminatedString” na stronie 1120
- “Metoda WriteShort” na stronie 1120
- “Metoda WriteString” na stronie 1120
- “Metoda WriteUInt2” na stronie 1120
- “WriteUnsigned-metoda typu Byte” na stronie 1121
- “Metoda WriteUTF” na stronie 1121

Dostęp do właściwości

Wszystkie właściwości mogą być odczytane w dowolnym momencie.

Właściwości elementu sterującego są tylko do odczytu, z wyjątkiem DataOffset , które jest odczytywanie_odczytu. Właściwości deskryptora komunikatu to wszystkie operacje odczytu i zapisu, z wyjątkiem BackoutCount i TotalMessage, które są tylko do odczytu.

Należy jednak pamiętać, że niektóre właściwości MQMD mogą być modyfikowane przez menedżer kolejek, gdy komunikat jest umieszczany w kolejce produktu WebSphere MQ . Szczegółowe informacje na temat sposobu ich modyfikowania znajdują się w polach MQMD .

Konwersja danych

Dane binarne można przekazać do komunikatu produktu WebSphere MQ , ustawiając właściwość CharacterSet na identyfikator kodowanego zestawu znaków menedżera kolejek (MQCCSI_Q_MGR) i przekazując go jako łańcuch. Za pomocą funkcji chr $można ustawić nieznakowe dane w łańcuchu.

Metody Read i Write wykonują konwersję danych. Są one przekształcane między formatami wewnętrznymi ActiveX , a formatami komunikatów produktu WebSphere MQ zgodnie z definicją właściwości Encoding i CharacterSet z deskryptora komunikatu. Podczas zapisywania komunikatu ustaw wartości w kodowaniu oraz CharacterSet , które są zgodne z charakterystyką odbiorcy komunikatu przed wywołaniem metody Write. Podczas odczytu komunikatu ten krok nie jest zwykle wymagany, ponieważ wartości te zostaną ustawione na podstawie tych wartości w przychodzącym MQMD.

Jest to dodatkowy krok konwersji danych, który jest wykonywany po konwersji wykonanej przez metodę MQQueue.Get .

Właściwość CompletionCode

Tylko do odczytu. Zwraca kod zakończenia WebSphere MQ ustawiony za pomocą najnowszej metody lub dostępu do właściwości wydanych dla tego obiektu.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Wartości:

- MQCC_OK
- MQCC_WARNING,
- MQCC_FAILED

Składnia: Aby uzyskać: *completioncode & = MQMessage*.**CompletionCode**

Właściwość DataLength

Tylko do odczytu. Ta właściwość zwraca wartość:

```
MQMessage.MessageLength - MQMessage.DataOffset
```

Może być używany przed metodą odczytu, aby sprawdzić, czy oczekiwana liczba znaków jest rzeczywiście obecna w buforze.

Wartością początkową jest zero.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Składnia: Aby uzyskać informacje: *bytesleft & = MQMessage*.**DataLength**

Właściwość DataOffset

Odczyt-zapis. Bieżąca pozycja w części Dane komunikatu obiektu komunikatu.

Wartość jest wyrażona jako przesunięcie bajtu od początku buforu danych komunikatu. Pierwszy znak w buforze odpowiada wartości DataOffset o wartości zero.

Metoda odczytu lub zapisu rozpoczyna swoją operację od znaku, do którego odwołuje się DataOffset. Te metody przetwarzają dane w buforze sekwencyjnie z tej pozycji i aktualizują wartość DataOffset tak, aby wskazywała na bajt (jeśli istnieje) bezpośrednio po ostatnim przetworzonym bajcie.

Parametr DataOffset może przyjmować tylko wartości z zakresu od zera do MessageLength włącznie. Gdy DataOffset = MessageLength wskazuje na koniec, to jest pierwszy niepoprawny znak buforu. W tej sytuacji dozwolone są metody zapisu-rozszerzają one dane w buforze i zwiększają wartość parametru MessageLength o liczbę dodanych bajtów. Odczyt poza koniec buforu jest niepoprawny.

Wartością początkową jest zero.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Składnia: Aby uzyskać: *currpos & = MQMessage*.**DataOffset**

Aby ustawić: *MQMessage*.**DataOffset** = *currpos &*

Właściwość MessageLength

Tylko do odczytu. Zwraca całkowitą długość części danych komunikatu w obiekcie komunikatu, niezależnie od wartości DataOffset.

Wartością początkową jest zero. Jest ona ustawiana na przychodzącą długość komunikatu po wywołaniu metody Get, która odwołuje się do tego obiektu komunikatu. Jest ona zwiększana, jeśli aplikacja korzysta z metody zapisu w celu dodania danych do obiektu. Metody odczytu nie mają wpływu na jego działanie.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Składnia: Aby uzyskać informacje: *msglength & = MQMessage*.**MessageLength**

Właściwość ReasonCode

Tylko do odczytu. Zwraca kod przyczyny ustawiony przez najnowszą metodę lub dostęp do właściwości wydany dla tego obiektu.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: do pobrania: *kod_przyczyny & = komunikat MQMessage*.**ReasonCode**

Właściwość ReasonName

Tylko do odczytu. Zwraca nazwę symboliczną najnowszego kodu przyczyny. Na przykład: "MQRC_QMGR_NOT_AVAILABLE". **Zdefiniowane w:** Klasa MQMessage

Typ danych: String

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: aby uzyskać: *reasonname $= MQMessage*.**ReasonName**

Właściwość AccountingToken

Odczyt-zapis. Element MQMD AccountingToken -część kontekstu tożsamości komunikatu.

Jej wartością początkową jest wszystkie wartości NULL.

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 32 znaków

Składnia: Aby uzyskać: *actoken $= MQMessage*.**AccountingToken**

Aby ustawić: *MQMessage.***AccountingToken** = *actoken $*

Więcej informacji na temat tego, kiedy należy używać właściwości AccountingTokenHex w miejsce właściwości AccountingToken , można znaleźć w sekcji “Właściwości deskryptora komunikatu” na stronie 1064 .

Właściwość Hex AccountingToken

Odczyt-zapis. Element MQMD AccountingToken -część kontekstu tożsamości komunikatu.

Każda z dwóch znaków reprezentuje szesnastkowy odpowiednik pojedynczego znaku ASCII. Na przykład para znaków "6" i "1" oznacza pojedynczy znak "A", a para znaków "6" i "2" reprezentuje pojedynczy znak "B" itd.

Należy podać 64 poprawne znaki szesnastkowe.

Jego wartością początkową jest "0 ... 0"

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch 64 znaków szesnastkowych reprezentujących 32 znaki ASCII

Składnia: Aby uzyskać: *actokenh $= MQMessage*.**AccountingTokenHex**

Aby ustawić: *MQMessage.***AccountingTokenHex** = *actokenh $*

Więcej informacji na temat tego, kiedy należy używać właściwości AccountingTokenHex w miejsce właściwości AccountingToken , można znaleźć w sekcji “Właściwości deskryptora komunikatu” na stronie 1064 .

Właściwość danych ApplicationId

Odczyt-zapis. Element MQMD ApplIdentityData-część kontekstu tożsamości komunikatu.

Jego początkowa wartość to wszystkie odstępy.

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 32 znaków

Składnia: Aby uzyskać: *applid $= MQMessage*.**ApplicationIdData**

Aby ustawić wartość: *MQMessage*.**ApplicationIdData** = *applid $*

Właściwość danych ApplicationOrigin

Odczyt-zapis. Element danych MQMD ApplOrigin-część kontekstu pochodzenia komunikatu.

Jego początkowa wartość to wszystkie odstępy.

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 4 znaków

Składnia: aby uzyskać następujące informacje: *applor $= MQMessage*.**ApplicationOriginData**

Aby ustawić wartość: *MQMessage*.**ApplicationOriginData** = *applor $*

Właściwość BackoutCount

Tylko do odczytu. Element MQMD BackoutCount.

Jego wartością początkową jest 0.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Składnia: Aby uzyskać następujące informacje: *backoutct & = MQMessage*.**BackoutCount**

Właściwość CharacterSet

Odczyt-zapis. Element MQMD CodedCharSetId.

Jego wartością początkową jest wartość specjalna **MQCCSI_Q_MGR** .

Jeśli parametr CharacterSet jest ustawiony na wartość **MQCCSI_Q_MGR**, strona kodowa dla bieżących ustawień narodowych jest używana do konwersji znaków w metodzie WriteString . W przypadku aplikacji serwera używana strona kodowa jest stroną kodową menedżera kolejek. W przypadku aplikacji klienckich jest to domyślna bieżąca strona kodowa ustawień narodowych.

Na przykład:

```
msg.CharacterSet = MQCCSI_Q_MGR
msg.WriteString(chr$(n))
```

gdzie 'n' jest większe lub równe zeru i mniejsze lub równe 255, powoduje, że jeden bajt wartości' n' jest zapisywany w buforze.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Składnia: Aby uzyskać: *:30ccid& = MQMessage* **.CharacterSet**

Aby ustawić: *MQMessage*.**CharacterSet**= *ccid &*

Przykład

Jeśli chcesz, aby łańcuch zapisany na stronie kodowej 437, wywołaj:

```
Message.CharacterSet = 437
Message.WriteString ("string to be written")
```

Przed wywołaniem dowolnego wywołania WriteString ustaw wartość, która ma być ustawiona w CharacterSet .

Właściwość CorrelationId

Odczyt-zapis. Element CorrelationId , który ma zostać włączony do deskryptora MQMD komunikatu podczas umieszczania w kolejce. Identyfikator, który ma być dopasowywany podczas pobierania komunikatu z kolejki.

Jej początkowa wartość jest równa null.

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 24 znaków

Składnia: Aby uzyskać: *correlid $= MQMessage*.**CorrelationId** , aby ustawić: *MQMessage*.**CorrelationId** = *correlid $*

Więcej informacji na temat tego, kiedy należy użyć wartości CorrelationIdHex w miejsce właściwości CorrelationId , można znaleźć w sekcji “Właściwości deskryptora komunikatu” na stronie 1064 .

Właściwość CorrelationIdHex

Odczyt-zapis. Element CorrelationId , który ma zostać włączony do deskryptora MQMD komunikatu podczas umieszczania w kolejce. Ponadto CorrelationId ma być porównywany podczas pobierania komunikatu z kolejki.

Co dwa znaki w łańcuchu reprezentują szesnastkowy odpowiednik pojedynczego znaku ASCII. Na przykład para znaków "6" i "1" oznacza pojedynczy znak "A", a para znaków "6" i "2" reprezentuje pojedynczy znak "B" itd.

Należy podać 48 poprawnych znaków szesnastkowych.

Jego początkowa wartość to "0 ... 0".

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 48 znaków szesnastkowych reprezentujących 24 znaki ASCII

Składnia: Aby uzyskać: *correlidh $= MQMessage*.**CorrelationIdHex**

Aby ustawić: *MQMessage*.**CorrelationIdHex** = *correlidh $*

Sekcja “Właściwości deskryptora komunikatu” na stronie 1064 zawiera omówienie sytuacji, w której należy użyć wartości CorrelationIdHex w miejscu właściwości CorrelationId .

Właściwość kodowania

Odczyt-zapis. Pole MQMD, które identyfikuje reprezentację używaną dla wartości liczbowych w danych komunikatu aplikacji.

Jego wartością początkową jest wartość specjalna MQENC_NATIVE, która różni się w zależności od platformy.

Ta właściwość jest używana przez następujące metody:

- Metoda ReadDecimal2
- Metoda ReadDecimal4
- Metoda ReadDouble
- Metoda ReadDouble4
- Metoda ReadFloat
- Metoda ReadInt2

- Metoda ReadInt4
- Metoda ReadLong
- Metoda ReadShort
- Metoda ReadUInt2
- Metoda WriteDecimal2
- Metoda WriteDecimal4
- Metoda WriteDouble
- Metoda WriteDouble4
- Metoda WriteFloat
- Metoda WriteInt2
- Metoda WriteInt4
- Metoda WriteLong
- Metoda WriteShort
- Metoda WriteUInt2

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Składnia: Aby uzyskać: *encoding & = MQMessage***.Kodowanie** Do ustawienia: *MQMessage***.Kodowanie** = *kodowanie &*

W przypadku przygotowywania do zapisu danych w buforze komunikatów należy ustawić to pole w taki sposób, aby były zgodne z charakterystyką platformy odbierającej menedżera kolejek, jeśli odbierający menedżer kolejek nie jest w stanie wykonać własnej konwersji danych.

Właściwość utraty ważności

Odczyt-zapis. Pole czasu utraty ważności MQMD, oczekiwane w dziesiątych częściach sekundy.

Jej wartością początkową jest wartość specjalna MQEI_UNLIMITED

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Składnia: Aby uzyskać: *wygaśnięcie & = komunikat MQMessage***.Utrata ważności**

Aby ustawić: *MQMessage***.Utrata ważności** = *utrata ważności &*

Właściwość sprzężenia zwrotnego

Odczyt-zapis. Pole informacji zwrotnej MQMD.

Jego początkowa wartość jest wartością specjalną MQFB_NONE.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Wartości:

- Patrz Feedback(Opinia).

Składnia: Aby uzyskać: *feedback & = MQMessage***.Opinia**

Aby ustawić: *MQMessage***.Opinia** = *informacja zwrotna &*

Właściwość Format

Odczyt-zapis. Pole formatu MQMD. Podaje nazwę wbudowanego lub zdefiniowanego przez użytkownika formatu opisowego, który opisuje rodzaj danych komunikatu.

Jego wartością początkową jest wartość specjalna MQFMT_NONE.

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 8 znaków

Składnia: Aby uzyskać: *format $= MQMessage*.**Format**

Aby ustawić: *MQMessage*.**Format** = *format $*

Właściwość GroupId

Odczyt-zapis. Element GroupId , który ma zostać włączony do rekordu MQPMR komunikatu podczas umieszczania w kolejce. Identyfikator, który ma być dopasowywany podczas pobierania komunikatu z kolejki. Jej wartością początkową jest wszystkie wartości NULL.

Zdefiniowane w:
Klasa MQMessage

Typ danych:
Łańcuch 24 znaków

Składnia: Do pobrania: *groupid $= MQMessage*.**GroupId**

Aby ustawić: *MQMessage*.**GroupId** = *groupid $*

Więcej informacji na temat sytuacji, w której należy użyć wartości GroupIdHex w miejsce właściwości GroupId , można znaleźć w sekcji “Właściwości deskryptora komunikatu” na stronie 1064 .

Właściwość GroupIdHex

Odczyt-zapis. Element GroupId , który ma zostać włączony do rekordu MQPMR komunikatu podczas umieszczania w kolejce. Identyfikator, który ma być dopasowywany podczas pobierania komunikatu z kolejki.

Co dwa znaki w łańcuchu reprezentują szesnastkowy odpowiednik pojedynczego znaku ASCII. Na przykład para znaków "6" i "1" oznacza pojedynczy znak "A", a para znaków "6" i "2" reprezentuje pojedynczy znak "B" itd.

Należy podać 48 poprawnych znaków szesnastkowych.

Jego początkowa wartość to "0 ... 0".

Zdefiniowane w:
Klasa MQMessage

Typ danych:
Łańcuch 48 znaków szesnastkowych, reprezentujący 24 znaki ASCII.

Składnia: Aby uzyskać: *groupidh $= MQMessage*.**GroupIdHex**

Aby ustawić wartość: *MQMessage*.**GroupIdHex** = *groupidh $*

Więcej informacji na temat sytuacji, w której należy użyć wartości GroupIdHex w miejsce właściwości GroupId , można znaleźć w sekcji “Właściwości deskryptora komunikatu” na stronie 1064 .

Właściwość MessageData

Odczyt-zapis. Pobiera lub ustawia całą zawartość komunikatu jako łańcuch znaków.

Zdefiniowane w: Klasa MQMessage

Typ danych: Variant

Uwaga: Typ danych używany przez tę właściwość jest typu Variant, ale MQAX oczekuje, że będzie to typ wariantu typu String. Jeśli zostanie przekazany wariant innego typu niż ten typ, zostanie zwrócony błąd MQRC_OBJECT_TYPE_ERROR.

Składnia: aby uzyskać: *String$* = *MQMessage*.**MessageData**

Aby ustawić: *MQMessage*.**MessageData** = *String$*

Właściwość MessageFlags

Odczyt-Zapis. Flagi komunikatów określające informacje sterujące Segmentation. Wartością początkową jest 0.

Zdefiniowane w:
Klasa MQMessage

Typ danych:
Długa liczba całkowita

Wartości:
Patrz MsgFlags (MQLONG).

Składnia: Aby uzyskać następujący komunikat: *messageflags & = MQMessage*.**MessageFlags**

Aby ustawić wartość: *MQMessage*.**MessageFlags** = *messageflags &*

Właściwość MessageId

Odczyt-zapis. Element MessageId , który ma zostać włączony do deskryptora MQMD komunikatu podczas umieszczania w kolejce. Identyfikator, który ma być dopasowywany podczas pobierania komunikatu z kolejki.

Jej wartością początkową jest wszystkie wartości NULL.

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 24 znaków

Składnia: Aby uzyskać następujący komunikat: *messageid $= MQMessage*.**MessageId**

Aby ustawić: *MQMessage*.**MessageId** = *messageid $*

Aby uzyskać więcej informacji na temat sytuacji, w której należy użyć wartości MessageIdHex w miejsce właściwości MessageId , należy zapoznać się z informacjami w sekcji “Właściwości deskryptora komunikatu” na stronie 1064 .

Właściwość Hex MessageId

Odczyt-zapis. Element MessageId , który ma zostać włączony do deskryptora MQMD komunikatu podczas umieszczania w kolejce. Ponadto MessageId ma być porównywany podczas pobierania komunikatu z kolejki.

Co dwa znaki w łańcuchu reprezentują szesnastkowy odpowiednik pojedynczego znaku ASCII. Na przykład para znaków "6" i "1" oznacza pojedynczy znak "A", a para znaków "6" i "2" reprezentuje pojedynczy znak "B" itd.

Należy podać 48 poprawnych znaków szesnastkowych.

Jego początkowa wartość to "0 ... 0".

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 48 znaków szesnastkowych reprezentujących 24 znaki ASCII

Składnia: Aby uzyskać: *messageidh $= MQMessage*.**MessageIdHex**

Aby ustawić: *MQMessage*.**MessageIdHex** = *messageidh $*

Więcej informacji na temat sytuacji, w której należy użyć wartości MessageIdHex w miejsce właściwości MessageId , zawiera sekcja “Właściwości deskryptora komunikatu” na stronie 1064 .

Właściwość Liczba MessageSequence

Odczyt-Zapis. Informacje o sekwencji identyfikujące komunikat w grupie. Wartością początkową jest 1.

Zdefiniowane w:
Klasa MQMessage

Typ danych:
Długa liczba całkowita

Wartości:
Patrz MsgSeqNumber (MQLONG).

Składnia: Aby uzyskać: *sequencenumber & = MQMessage*.**SequenceNumber**

Aby ustawić: *MQMessage*.**SequenceNumber** = *sequencenumber &*

Właściwość MessageType

Odczyt-zapis. Pole MsgType deskryptora MQMD.

Jego początkowa wartość to MQMT_DATAGRAM.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Wartości:

- Patrz MsgType (MQLONG).

Składnia: do pobrania: *msgtype & = MQMessage*.**MessageType**

Aby ustawić: *MQMessage*.**MessageType** = *typ_komunikatu &*

Właściwość przesunięcia

Odczyt-Zapis. Przesunięcie w posegmentowanym komunikacie. Wartością początkową jest 0.

Zdefiniowane w:
Klasa MQMessage

Typ danych:
Długa liczba całkowita

Wartości:
Patrz Przesunięcie (MQLONG).

Składnia: Aby uzyskać: *offset & = MQMessage*.**Przesunięcie**

Aby ustawić: *MQMessage*.**Przesunięcie** = *przesunięcie &*

Właściwość OriginalLength

Odczyt-Zapis. Oryginalna długość segmentowanego komunikatu. Wartością początkową jest MQOL_UNDEFINED.

Zdefiniowane w:
Klasa MQMessage

Typ danych:
Długa liczba całkowita

Wartości:
Patrz OriginalLength (MQLONG).

Składnia: Do pobrania: *originallength & = MQMessage*.**OriginalLength**

Aby ustawić: *MQMessage*.**OriginalLength** = *originallength &*

Właściwość trwałości

Odczyt-zapis. Ustawienie trwałości komunikatu.

Jej wartością początkową jest MQPER_PERSISTENCE_AS_Q_DEF.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Składnia: Aby uzyskać: *utrwal & = komunikat_MQMessage*.**Trwałość**

Aby ustawić: *MQMessage*.**Trwałość** = *utrwal &*

Właściwość priorytetu

Odczyt-zapis. Priorytet komunikatu.

Jej wartością początkową jest wartość specjalna MQPRI_PRIORITY_AS_Q_DEF

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Składnia: Aby uzyskać: *priority & = MQMessage*.**Priorytet**

Aby ustawić: *MQMessage*.**Priorytet** = *priorytet &*

PutApplication, właściwość Nazwa

Odczyt-zapis. Nazwa wywołania MQMD PutAppl-część kontekstu źródłowego komunikatu.

Jego początkowa wartość to wszystkie odstępy.

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 28 znaków

Składnia: Aby uzyskać: *putapplnm $= MQMessage*.**PutApplicationNazwa**

Aby ustawić wartość: *MQMessage*.**PutApplicationName** = *putapplnm $*

PutApplication, właściwość typu

Odczyt-zapis. Typ wywołania MQMD PutAppl-część kontekstu źródłowego komunikatu.

Jego początkowa wartość to MQAT_NO_CONTEXT

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Wartości:

- Patrz PutApplType (MQLONG).

Składnia: Aby uzyskać: *putappltp & = MQMessage*.**PutApplicationTyp**

Aby ustawić wartość: *MQMessage*.**PutApplicationType** = *putappltp &*

Właściwość Czas PutDate

Odczyt/zapis. Ta właściwość łączy pola PutDate (PutDate) i PutTime (PutTime) z tabeli MQMD. Są to części kontekstu źródłowego komunikatu, które wskazują, kiedy komunikat został umieszczony.

Rozszerzenie ActiveX jest przekształcane między formatem daty/godziny ActiveX a formatami daty i godziny używani w produkcie WebSphere MQ MQMD. Jeśli zostanie odebrany komunikat, który ma niepoprawną wartość PutDate lub PutTime, właściwość Czas PutDateTime po metodzie get zostanie ustawiona na wartość EMPTY.

Jego wartością początkową jest EMPTY.

Zdefiniowane w: Klasa MQMessage

Typ danych: Variant typu 7 (data/godzina) lub EMPTY.

Składnia: Aby uzyskać: *datetime = MQMessage*.**PutDate**

Aby ustawić: *MQMessage*.**PutDate, godzina** = *datetime*

Właściwość Nazwa ReplyToQueueManager

Odczyt-zapis. Pole menedżera kolejek MQMD ReplyTo.

Wartość początkowa to wszystkie odstępy.

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 48 znaków

Składnia: Aby uzyskać: *replytoqmgr $= MQMessage*.**ReplyToQueueManagerName**

Aby ustawić wartość: *MQMessage*.**ReplyToQueueManagerName** = *replytoqmgr $*

Właściwość ReplyToQueueName

Odczyt-zapis. Pole Q MQMD ReplyTo(ReplyTo).

Wartość początkowa to wszystkie odstępy.

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 48 znaków

Składnia: Do pobrania: *replytoq $= MQMessage*.**ReplyToQueueName**

Aby ustawić: *MQMessage*.**ReplyToQueueName** = *replytoq $*

Właściwość raportu

Odczyt-zapis. Opcje raportu.

Jego początkowa wartość to MQRO_NONE.

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Wartości:

- Patrz Raport.

Składnia: Aby uzyskać: *report & = MQMessage*.**Raport**

Aby ustawić: *MQMessage*.**Raport** = *raport &*

Właściwość długości TotalMessageLength

Tylko do odczytu. Pobiera długość ostatniego komunikatu odebranego przez komendę MQGET. Jeśli komunikat nie został obcięty, ta wartość jest równa wartości właściwości MessageLength .

Zdefiniowane w: Klasa MQMessage

Typ danych: Long

Składnia: Aby uzyskać informacje: *totalmessagelength & = MQMessage*.**TotalMessageLength**

Właściwość UserId

Odczyt-zapis. Element MQMD UserIdentifier -część kontekstu tożsamości komunikatu.

Jego początkowa wartość to wszystkie odstępy.

Zdefiniowane w: Klasa MQMessage

Typ danych: Łańcuch o długości 12 znaków

Składnia: Aby uzyskać: *userid $= MQMessage*.**UserId**

Aby ustawić: *MQMessage*.**UserId** = *id_użytkownika $*

ClearError-metoda kodów

Resetuje parametr CompletionCode do wartości MQCC_OK i ReasonCode na wartość MQRC_NONE zarówno dla klasy MQMessage, jak i klasy MQSession.

Zdefiniowane w: Klasa MQMessage

Składnia: Call *MQMessage*.**ClearErrorCodes ()**

Metoda ClearMessage

Ta metoda kasuje część buforu danych obiektu MQMessage. Wszystkie dane komunikatu w buforze danych zostaną utracone, ponieważ wartości MessageLength, DataLengthi DataOffset są ustawione na zero.

Część deskryptora komunikatu (MQMD) nie może zostać zmieniona; przed ponownym użyciem obiektu MQMessage może być konieczna modyfikacja niektórych pól MQMD. Aby ponownie ustawić pola MQMD, użyj opcji Nowa, aby zastąpić obiekt nową instancją.

Zdefiniowane w: Klasa MQMessage

Składnia: Call *MQMessage*.**ClearMessage()**

Metoda odczytu

Odczytuje sekwencję bajtów z buforu komunikatów do tablicy bajtów. Wartość DataOffset jest zwiększana, a długość danych jest zmniejszana o liczbę odczytanych bajtów.

Zdefiniowane w:
Klasa MQMessage

Składnia: Data = MQMessage.**Odczyt**(len &)

Parametry:
len &: Long. Długość danych w bajtach do odczytania.

Metoda ReadBoolean

Odczytuje jednobajtową wartość boolowskową z bieżącej pozycji w buforze komunikatów i zwraca 2-bajtową wartość boolowskim TRUE (-1) /FALSE (0). Wartość DataOffset jest zwiększana o jeden, a długość danych jest zmniejszana o jeden.

Zdefiniowane w:
Klasa MQMessage

Składnia: *value = MQMessage*.**ReadBoolean**

Metoda ReadByte

Ta metoda odczytuje 1 bajt z buforu danych komunikatu, począwszy od znaku, do którego odwołuje się DataOffset , i zwraca go jako liczbę całkowitą (2-bajtową ze znakiem) w zakresie od -128 do 127.

Metoda kończy się niepowodzeniem, jeśli właściwość MQMessage.DataLength jest mniejsza niż 1, gdy jest ona wydawana.

Wartość DataOffset jest zwiększana o 1, a DataLength jest zmniejszana o 1, jeśli metoda zakończy się powodzeniem.

Przyjmuje się, że bajt danych komunikatu jest binarną liczbą całkowitą ze znakiem.

Zdefiniowane w: Klasa MQMessage

Składnia: *integerv% = MQMessage*.**ReadByte**

Metoda ReadDecimal2

Odczytuje 2-bajtową upakowaną liczbę dziesiętną i zwraca ją jako dwubajtową liczbę całkowitą ze znakiem. Wartość DataOffset jest zwiększana o dwie, a długość danych jest zmniejszana o dwa.

Zdefiniowane w:
Klasa MQMessage

Składnia: *value% = MQMessage*.**ReadDecimal2**

Metoda ReadDecimal4

Odczytuje 4-bajtowy upakowany numer dziesiętny i zwraca go jako 4-bajtową liczbę całkowitą ze znakiem. Wartość DataOffset jest zwiększana o cztery, a długość danych jest zmniejszana o cztery.

Zdefiniowane w:
Klasa MQMessage

Składnia: Call *value & = MQMessage*.**ReadDecimal4**

Metoda ReadDouble

Ta metoda odczytuje 8 bajtów z buforu danych komunikatów, począwszy od bajtu, do którego odwołuje się funkcja DataOffset , i zwraca go jako wartość zmiennopozycyjną Double (signed 8-byte).

Metoda kończy się niepowodzeniem, jeśli właściwość MQMessage.DataLength jest mniejsza niż 8, gdy jest ona wydawana.

Wartość DataOffset jest zwiększana o 8, a DataLength jest zmniejszana o 8, jeśli metoda zakończy się powodzeniem.

Przyjmuje się, że 8 znaków danych komunikatu jest binarną liczbą zmiennopozycyjną. Kodowanie jest określane przez właściwość MQMessage.Encoding . Należy pamiętać, że konwersja z formatu System/360 nie jest obsługiwana.

Zdefiniowane w: Klasa MQMessage

Składnia: *wątpliwość# = MQMessage*.**ReadDouble**

Metoda ReadDouble4

Metody ReadDouble4 i WriteDouble4 są alternatywami dla opcji ReadFloat i WriteFloat. Jest to spowodowane tym, że obsługują one 4-bajtowe wartości komunikatów zmiennopozycyjnych System/390 , które są zbyt duże, aby można było przekształcić je w 4-bajtowy format zmiennopozycyjny IEEE.

Ta metoda odczytuje 4 bajty z buforu danych komunikatów, począwszy od bajtu, do którego odwołuje się funkcja DataOffset , i zwraca go jako wartość zmiennopozycyjną Double (signed 8-byte).

Metoda kończy się niepowodzeniem, jeśli właściwość MQMessage.DataLength jest mniejsza niż 4, gdy jest ona wydawana.

Wartość DataOffset jest zwiększana o 4, a DataLength jest zmniejszana o 4, jeśli metoda zakończy się powodzeniem.

Przyjmuje się, że 4 znaki danych komunikatu są binarną liczbą zmiennopozycyjną. Kodowanie jest określane przez właściwość MQMessage.Encoding . Należy pamiętać, że konwersja z formatu System/360 nie jest obsługiwana.

Zdefiniowane w: Klasa MQMessage

Składnia: *wątpliwość# = komunikat_MQMessage.***ReadDouble4**

Metoda ReadFloat

Ta metoda odczytuje 4 bajty z buforu danych komunikatów, począwszy od bajtu, do którego odwołuje się funkcja DataOffset , i zwraca go jako wartość zmiennopozycyjną o pojedynczej (podpisanej 4-bajtowej) wartości zmiennopozycyjnej.

Metoda kończy się niepowodzeniem, jeśli właściwość MQMessage.DataLength jest mniejsza niż 4, gdy jest ona wydawana.

Wartość DataOffset jest zwiększana o 4, a DataLength jest zmniejszana o 4, jeśli metoda zakończy się powodzeniem.

Przyjmuje się, że 4 znaki danych komunikatu są liczbą zmiennopozycyjną. Kodowanie jest określane przez właściwość MQMessage.Encoding . Należy pamiętać, że konwersja z formatu System/360 nie jest obsługiwana.

Zdefiniowane w: Klasa MQMessage

Składnia: *singlev! = MQMessage*.**ReadFloat**

Metoda ReadInt2

Metoda jest identyczna z metodą ReadShort .

Składnia: *integerv% = MQMessage*.**ReadInt2**

Metoda ReadInt4

Ta metoda jest identyczna z metodą ReadLong .

Składnia: *bigint & = MQMessage*.**ReadInt4**

Metoda ReadLong

Ta metoda odczytuje 4 bajty z buforu danych komunikatów, począwszy od bajtu, do którego odwołuje się funkcja DataOffset , i zwraca go jako wartość typu Long (podpisaną 4-bajtową).

Metoda kończy się niepowodzeniem, jeśli właściwość MQMessage.DataLength jest mniejsza niż 4, gdy jest ona wydawana.

Wartość DataOffset jest zwiększana o 4, a DataLength jest zmniejszana o 4, jeśli metoda zakończy się powodzeniem.

Przyjmuje się, że 4 znaki danych komunikatu są binarną liczbą całkowitą. Kodowanie jest określane przez właściwość MQMessage.Encoding .

Zdefiniowane w: Klasa MQMessage

Składnia: *bigint & = MQMessage*.**ReadLong**

Metoda ReadNullTerminatedString

Ta metoda jest używana w miejsce ReadString , jeśli łańcuch może zawierać znaki o kodzie zero osadzonym.

Ta metoda odczytuje określoną liczbę bajtów z buforu danych komunikatów, począwszy od bajtu, do którego odwołuje się funkcja DataOffset , i zwraca go jako łańcuch ActiveX . Jeśli łańcuch zawiera osadzoną wartość NULL przed końcem, długość zwróconego łańcucha zostanie zmniejszona tak, aby odzwierciedlała tylko te znaki przed wartością NULL.

Wartość DataOffset jest zwiększana, a wartość DataLength jest zmniejszana o podaną wartość, niezależnie od tego, czy łańcuch zawiera osadzone znaki o wartości NULL.

Zakłada się, że znaki w danych komunikatu są łańcuchami na stronie kodowej, która jest określona przez właściwość MQMessage.CharacterSet . Konwersja na reprezentację ActiveX jest wykonywana dla aplikacji.

Zdefiniowane w:
Klasa MQMessage

Składnia: *łańcuch $ = komunikat MQMessage*.**ReadNullTerminatedString(długość &)**

Parametry:
length & *Long.* Długość pola łańcucha w bajtach.

Metoda ReadShort

Ta metoda odczytuje 2 bajty z buforu danych komunikatów, począwszy od bajtu, do którego odwołuje się DataOffset , i zwraca go jako wartość typu Integer (podpisaną 2-bajtową).

Metoda kończy się niepowodzeniem, jeśli właściwość MQMessage.DataLength jest mniejsza niż wartość 2, gdy jest ona wydawana.

Wartość DataOffset jest zwiększana o 2, a wartość DataLength jest zmniejszana o 2, jeśli metoda powiedzie się.

Przyjmuje się, że 2 znaki danych komunikatu są binarną liczbą całkowitą. Kodowanie jest określane przez właściwość MQMessage.Encoding .

Zdefiniowane w: Klasa MQMessage

Składnia: *integerv% = MQMessage*.**ReadShort**

Metoda ReadString

Ta metoda odczytuje n bajtów z buforu danych komunikatu, począwszy od bajtu, do którego odwołuje się element DataOffset , i zwraca go jako łańcuch ActiveX .

Metoda kończy się niepowodzeniem, jeśli właściwość MQMessage.DataLength jest mniejsza niż n, gdy jest ona wydawana.

Wartość DataOffset jest zwiększana o n, a DataLength jest zmniejszana o n, jeśli metoda zakończy się powodzeniem.

Przyjmuje się, że n znaków danych komunikatu jest łańcuchem na stronie kodowej, który jest określony przez właściwość MQMessage.CharacterSet . Konwersja na reprezentację ActiveX jest wykonywana dla aplikacji.

Zdefiniowane w: Klasa MQMessage

Składnia: *stringv $= MQMessage*.**ReadString**(*długość &*)

Parametr
długość & Długa. Długość pola łańcucha w bajtach.

Metoda ReadUInt2

Ta metoda odczytuje 2 bajty z buforu danych komunikatów, począwszy od bajtu, do którego odwołuje się funkcja DataOffset , i zwraca go jako wartość typu Long (podpisaną 4-bajtową).

Metoda kończy się niepowodzeniem, jeśli właściwość MQMessage.DataLength jest mniejsza niż wartość 2, gdy jest ona wydawana.

Wartość DataOffset jest zwiększana o 2, a wartość DataLength jest zmniejszana o 2, jeśli metoda powiedzie się.

Przyjmuje się, że 2 bajty danych komunikatu są binarną liczbą całkowitą bez znaku. Kodowanie jest określane przez właściwość MQMessage.Encoding .

Zdefiniowane w: Klasa MQMessage

Składnia: *bigint & = MQMessage*.**ReadUInt2**

ReadUnsigned-metoda typu Byte

Ta metoda odczytuje 1 bajt z buforu danych komunikatów, zaczynając od bajtu, do którego odwołuje się funkcja DataOffset , i zwraca go jako liczbę całkowitą (2-bajtową ze znakiem) w zakresie od 0 do 255.

Metoda kończy się niepowodzeniem, jeśli właściwość MQMessage.DataLength jest mniejsza niż 1, gdy jest ona wydawana.

Wartość DataOffset jest zwiększana o 1, a DataLength jest zmniejszana o 1, jeśli metoda zakończy się powodzeniem.

Przyjmuje się, że 1 znak danych komunikatu jest binarną liczbą całkowitą bez znaku.

Zdefiniowane w: Klasa MQMessage

Składnia: *integerv% = MQMessage*.**ReadUnsignedBajt**

Metoda ReadUTF

Ta metoda odczytuje łańcuch formatu UTF z komunikatu rozpoczynający się od bajtu przywołanego przez element DataOffset i zwraca go jako łańcuch ActiveX . Łańcuch w komunikacie składa się z dwubajtowej długości, po której następują dane znakowe.

Metoda kończy się niepowodzeniem, jeśli właściwość MQMessage.DataLength jest mniejsza niż długość łańcucha, gdy jest ona wydawana.

Wartość DataOffset jest zwiększana o długość łańcucha, a wartość DataLength jest zmniejszana o długość łańcucha, jeśli metoda powiedzie się.

Zdefiniowane w:
Klasa MQMessage

Składnia: *wartość $= MQMessage*.**ReadUTF**

Metoda ResizeBuffer

Ta metoda zmienia ilość pamięci masowej obecnie przydzielonej wewnętrznie w celu przechowywania buforu danych komunikatu. Daje on aplikacji pewną kontrolę nad automatycznym zarządzaniem buforami, w tym, że jeśli aplikacja wie, że ma zajmować się dużym komunikatem, może zapewnić, że przydzielany jest wystarczająco duży bufor. Aplikacja nie musi korzystać z tego wywołania-jeśli tak się nie stanie, automatyczny kod zarządzania buforami powiększy rozmiar bufora, aby zmieścić się.

Jeśli wielkość buforu zostanie ponownie zmniejszona, a bieżąca wartość parametru MessageLengthzostanie zmniejszona, ryzyko utraty danych będzie ryzykować. Jeśli dane zostaną utracone, metoda zwraca kod CompletionCode o wartości MQCC_WARNING i ReasonCode o wartości MQRC_DATA_OBCIĘTE.

W przypadku zmiany wielkości buforu na wartość mniejszą niż wartość właściwości **DataOffset** , należy:

- Właściwość **DataOffset** jest zmieniana w taki sposób, aby wskazywała na koniec nowego buforu.
- Właściwość **DataLength** jest ustawiona na zero.
- Właściwość **MessageLength** została zmieniona na nową wielkość buforu.

Zdefiniowane w:
Klasa MQMessage

Składnia: *MQMessage*.**ResizeBuffer**(*Length &*)

parametr:
Długość & długa. Wielkość wymagana w znakach.

Metoda zapisu

Zapisuje sekwencję bajtów w buforze komunikatów z tablicy bajtów na pozycji, do której odnosi się przesunięcie danych. Jeśli jest to konieczne, długość buforu (MQMessage.MQMessageLength) jest rozszerzana w taki sposób, aby pomieścić pełną długość tablicy bajtów. DataOffset jest zwiększane o liczbę bajtów zapisanych, jeśli metoda zakończy się powodzeniem.

Zdefiniowane w:
Klasa MQMessage

Składnia: Wywołanie *MQMessage*.**Zapis**(wartość)

Parametry:
data: tablica bajtów lub odwołanie wariantu do tablicy bajtów

Metoda WriteBoolean

Zapisuje jednobajtową wartość boolowką w bieżącym położeniu w buforze komunikatów z dwubajtowej wartości boolowskiej. Wartość DataOffset jest zwiększana o jeden.

Zdefiniowane w:
Klasa MQMessage

Składnia: wywołaj metodę *MQMessage*.**WriteBoolean**(wartość)

parametr:
wartość: Boolean (2-bajty). Wartość, która ma zostać zapisana.

Metoda WriteByte

Ta metoda przyjmuje podpisaną 2-bajtową liczbę całkowitą i zapisuje ją w buforze danych komunikatu jako jednobajtowy numer binarny w położeniu określonym przez wartość DataOffset. Zastępuje on wszystkie dane już znajdujące się w pozycji w buforze i rozszerza długość buforu (MQMessage.MessageLength), jeśli jest to konieczne.

Wartość DataOffset jest zwiększana o jeden, jeśli metoda zakończy się powodzeniem.

Podana wartość powinna miewać się w zakresie od -128 do 127. Jeśli tak nie jest, metoda zwraca wartość CompletionCode MQCC_FAILED i ReasonCode MQRC_WRITE_VALUE_ERROR.

Zdefiniowane w: Klasa MQMessage

Składnia: Call *MQMessage*.**WriteByte**(*wartość%*)

Parametr: *wartość%* Liczba całkowita. Wartość, która ma zostać zapisana.

Metoda WriteDecimal2

Zapisuje podpisaną 2-bajtową liczbę całkowitą w postaci dwubajtowej upakowanej liczby dziesiętnej. Wartość DataOffset jest zwiększana o dwa.

Zdefiniowane w:
Klasa MQMessage

Składnia: Wywołaj *MQMessage*.**WriteDecimal2**(wartość%)

parametr:
wartość% Integer. Wartość, która ma zostać zapisana.

Metoda WriteDecimal4

Zapisuje podpisaną 4-bajtową liczbę całkowitą jako 4-bajtową upakowaną liczbę dziesiętną. Wartość DataOffset jest zwiększana o cztery.

Zdefiniowane w:
Klasa MQMessage

Składnia: Wywołanie *MQMessage*.**WritedDecimal4**(wartość &)

parametr:
value & Long. Wartość, która ma zostać zapisana.

Metoda WriteDouble

Ta metoda przyjmuje 8-bajtową wartość zmiennopozycyjną i zapisuje ją w buforze danych komunikatów jako 8-bajtowa liczba zmiennopozycyjna, począwszy od pozycji, do której odwołuje się DataOffset. Zastępuje on wszystkie dane znajdujące się już na tych pozycjach w buforze i rozszerza długość buforu (MQMessage.MessageLength), jeśli jest to konieczne.

Wartość DataOffset jest zwiększana o 8, jeśli metoda zakończy się powodzeniem.

Metoda przekształca się w reprezentację zmiennopozycyjną określoną przez właściwość MQMessage.Encoding . *Konwersja do formatu System/360 nie jest obsługiwana.*

Zdefiniowane w: Klasa MQMessage

Składnia: Call *MQMessage*.**WriteDouble**(*wartość#*)

parametr:
value# Double. Wartość, która ma zostać zapisana.

Metoda WriteDouble4

Opis sytuacji, w których należy użyć funkcji ReadDouble4 i WriteDouble4 w miejsce ReadFloat i WriteFloat, znajduje się w sekcji “Metoda ReadDouble4” na stronie 1114 .

Ta metoda przyjmuje 8-bajtową wartość zmiennopozycyjną i zapisuje ją w buforze danych komunikatów jako 4-bajtową liczbę zmiennopozycyjną, rozpoczynając od pozycji, do której odwołuje się DataOffset.

Wartość DataOffset jest zwiększana o 4, jeśli metoda zakończy się powodzeniem.

Zastępuje on wszystkie dane znajdujące się już na tych pozycjach w buforze i rozszerza długość buforu (MQMessage.MessageLength), jeśli jest to konieczne.

Metoda przekształca się w reprezentację zmiennopozycyjną określoną przez właściwość MQMessage.Encoding . *Konwersja do formatu System/360 nie jest obsługiwana.*

Zdefiniowane w: Klasa MQMessage

Składnia: wywołaj metodę *MQMessage*.**WriteDouble4 (value#)**

Parametr: *value#* Double. Wartość, która ma zostać zapisana.

Metoda WriteFloat

Ta metoda przyjmuje 4-bajtową wartość zmiennopozycyjną i zapisuje ją w buforze danych komunikatu jako 4-bajtowy numer zmiennopozycyjny rozpoczynający się od znaku, do którego odwołuje się element DataOffset. Zastępuje on wszystkie dane znajdujące się już na tych pozycjach w buforze i rozszerza długość buforu (MQMessage.MessageLength), jeśli jest to konieczne.

Wartość DataOffset jest zwiększana o 4, jeśli metoda zakończy się powodzeniem.

Metoda przekształca się w reprezentację binarną określoną przez właściwość MQMessage.Encoding . *Konwersja do formatu System/360 nie jest obsługiwana.*

Zdefiniowane w: Klasa MQMessage

Składnia: wywołaj metodę *MQMessage*.**WriteFloat**(*wartość!*)

Parametr *wartość!* Float. Wartość, która ma zostać zapisana.

Metoda WriteInt2

Ta metoda jest identyczna z metodą WriteShort .

Składnia: Wywołanie *MQMessage*.**WriteInt2**(*wartość%*)

Parametr *wartość%* Liczba całkowita. Wartość, która ma zostać zapisana.

Metoda WriteInt4

Ta metoda jest identyczna z metodą WriteLong .

Składnia: Wywołanie *MQMessage*.**WriteInt4**(*wartość &*)

Parametr *wartość &* Long. Wartość, która ma zostać zapisana.

Metoda WriteLong

Ta metoda przyjmuje podpisaną 4-bajtową liczbę całkowitą i zapisuje ją w buforze danych komunikatów jako 4-bajtowy numer binarny rozpoczynający się od bajtu przywołanego przez DataOffset. Zastępuje on wszystkie dane znajdujące się już na tych pozycjach w buforze i rozszerza długość buforu (MQMessage.MessageLength), jeśli jest to konieczne.

Wartość DataOffset jest zwiększana o 4, jeśli metoda zakończy się powodzeniem.

Metoda przekształca się w reprezentację binarną określoną przez właściwość MQMessage.Encoding .

Zdefiniowane w: Klasa MQMessage

Składnia: wywołaj *MQMessage*.**WriteLong**(*wartość &*)

Parametr *wartość &* Long. Wartość, która ma zostać zapisana.

Metoda WriteNullTerminatedString

Ta metoda wykonuje normalny łańcuch WriteString i dopełnia wszystkie pozostałe bajty aż do określonej długości z wartością NULL. Jeśli liczba bajtów zapisanych przez początkowy łańcuch zapisu jest równa określonej długości, wówczas nie są zapisywane żadne znaki puste. Jeśli liczba bajtów przekracza określoną długość, to jest ustawiony błąd (kod przyczyny MQRC_WRITE_VALUE_ERROR).

Wartość DataOffset jest zwiększana o określoną długość, jeśli metoda zakończy się powodzeniem.

Zdefiniowane w: Klasa MQMessage

Składnia: Call *MQMessage*.**WriteNullTerminatedString***(value$, length &)*

Parametry:
wartość $String. Wartość, która ma zostać zapisana.

długość i długość. Długość pola łańcucha w bajtach.

Metoda WriteShort

Ta metoda przyjmuje 2-bajtową liczbę całkowitą i zapisuje ją w buforze danych komunikatu jako dwubajtowy numer binarny rozpoczynający się w bajcie przywołanym przez DataOffset. Zastępuje on wszystkie dane znajdujące się na tych pozycjach w buforze, a w razie potrzeby rozszerzy długość buforu (MQMessage.MessageLength).

Wartość DataOffset jest zwiększana o 2, jeśli metoda powiedzie się.

Metoda przekształca się w reprezentację binarną określoną przez właściwość MQMessage.Encoding .

Zdefiniowane w: Klasa MQMessage

Składnia: Call *MQMessage***.WriteShort**(*wartość%*)

Parametr *wartość%* Liczba całkowita. Wartość, która ma zostać zapisana.

Metoda WriteString

Ta metoda pobiera łańcuch ActiveX i zapisuje go w buforze danych komunikatu, rozpoczynając od bajtu przywołanego przez element DataOffset. Zastępuje on wszystkie dane znajdujące się na tych pozycjach w buforze, a w razie potrzeby rozszerzy długość buforu (MQMessage.MessageLength).

DataOffset jest zwiększane o długość łańcucha w bajtach, jeśli metoda zakończy się powodzeniem.

Metoda przekształca znaki w stronę kodową określoną za pomocą właściwości MQMessage.CharacterSet .

Zdefiniowane w: Klasa MQMessage

Składnia: Wywołanie *MQMessage*.**WriteString**(*wartość $*)

Parametr *wartość $* Łańcuch. Wartość, która ma zostać zapisana.

Metoda WriteUInt2

Ta metoda przyjmuje podpisaną 4-bajtową liczbę całkowitą i zapisuje ją w buforze danych komunikatu jako dwubajtowy, niepodpisany numer binarny, rozpoczynający się od bajtu, do którego odwołuje się DataOffset. Zastępuje on wszystkie dane znajdujące się już na tych pozycjach w buforze i rozszerza długość buforu (MQMessage.MessageLength), jeśli jest to konieczne.

Wartość DataOffset jest zwiększana o 2, jeśli metoda powiedzie się.

Metoda przekształca się w reprezentację binarną określoną przez właściwość MQMessage.Encoding . Podana wartość powinna miewać się w zakresie od 0 do 2 * *16-1. Jeśli nie jest to metoda, zwraca wartość CompletionCode MQCC_FAILED i ReasonCode MQRC_WRITE_VALUE_ERROR.

Zdefiniowane w: Klasa MQMessage

Składnia: Wywołanie *MQMessage***.WriteUInt2**(*wartość &*)

Parametr *wartość &* Long. Wartość, która ma zostać zapisana.

WriteUnsigned-metoda typu Byte

Ta metoda przyjmuje podpisaną 2-bajtową liczbę całkowitą i zapisuje ją w buforze danych komunikatów jako jednobajtowy niepodpisany numer binarny, rozpoczynający się od znaku, do którego odwołuje się DataOffset. Zastępuje on wszystkie dane znajdujące się już na tych pozycjach w buforze i rozszerza długość buforu (MQMessage.MessageLength), jeśli jest to konieczne.

Wartość DataOffset jest zwiększana o 1, jeśli metoda zakończy się powodzeniem.

Podana wartość powinna miewać się w zakresie od 0 do 255. Jeśli nie jest to metoda, zwraca wartość CompletionCode MQCC_FAILED i ReasonCode MQRC_WRITE_VALUE_ERROR.

Zdefiniowane w:
Klasa MQMessage

Składnia: Call *MQMessage*.**WriteUnsignedByte**(*value%*)

Parametr *wartość%* Liczba całkowita. Wartość, która ma zostać zapisana.

Metoda WriteUTF

Ta metoda pobiera łańcuch ActiveX i zapisuje go w buforze danych komunikatów w bieżącej pozycji w formacie UTF. Zapis danych składa się z dwubajtowej długości, po której następują dane znakowe. DataOffset jest zwiększane o długość łańcucha, jeśli metoda zakończy się powodzeniem.

Zdefiniowane w:
Klasa MQMessage

Składnia: Wywołanie *MQMessage*.**WriteUTF**(wartość$)

parametr:
wartość $String. Wartość, która ma zostać zapisana.

Klasa opcji MQPutMessage

Ta klasa obudowuje różne opcje, które sterują działaniem umieszczania komunikatu w kolejce WebSphere MQ .

Zawieranie

Klasa opcji MQPutMessagejest zawarta w klasie MQSession.

Tworzenie

Opcja **Nowy** powoduje utworzenie nowego obiektu opcji MQPutMessagei ustawia wszystkie jego właściwości na wartości początkowe.

Alternatywnie można użyć metody AccessPutMessageOptions klasy MQSession.

Składnia

Dim *pmo* **Jako Nowy Opcje programu MQPutMessage** or

Ustaw *pmo* = **Nowy Opcje programu MQPutMessage**

Właściwości

- "Właściwość CompletionCode" na stronie 1122.
- "Właściwość opcji" na stronie 1122.
- "Właściwość ReasonCode" na stronie 1122.
- "Właściwość ReasonName" na stronie 1122.

- “Właściwość RecordFields” na stronie 1123.
- “Właściwość ResolvedQueueManagerName” na stronie 1123.
- “Właściwość Nazwa ResolvedQueue” na stronie 1123.

Metody

- “ClearError-metoda kodów” na stronie 1123.

Właściwość CompletionCode

Tylko do odczytu. Zwraca kod zakończenia ustawiony przez ostatnią metodę lub dostęp do właściwości wystawiony dla obiektu.

Zdefiniowane w: klasa opcji MQPutMessage

Typ danych: Long

Wartości:

- MQCC_OK
- MQCC_WARNING,
- MQCC_FAILED

Składnia: Do pobrania: *completioncode & = PutOpts*.**CompletionCode**

Właściwość opcji

Odczyt-zapis. Pole Opcje MQPMO. Wartością początkową tego pola jest MQPMO_NONE. Więcej informacji na ten temat zawiera sekcja Opcje MQPMO.

Zdefiniowana w: MQPutMessage-Klasa opcji.

Typ danych: Long

Składnia: Aby uzyskać: *options & = PutOpts*.**Opcje**

Aby ustawić: *PutOpts*.**Opcje** = *opcje &*

Opcje MQPMO_PASS_IDENTITY_CONTEXT i MQPMO_PASS_ALL_CONTEXT nie są obsługiwane.

Właściwość ReasonCode

Tylko do odczytu. Zwraca kod przyczyny ustawiony przez ostatnią metodę lub dostęp do właściwości wystawiony dla obiektu.

Zdefiniowane w: klasa opcji MQPutMessage

Typ danych: Long

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: Do pobrania: *kod_przyczyny & = PutOpts*.**ReasonCode**

Właściwość ReasonName

Tylko do odczytu. Zwraca nazwę symboliczną najnowszego kodu przyczyny. Na przykład: "MQRC_QMGR_NOT_AVAILABLE".

Zdefiniowane w: klasa opcji MQPutMessage

Typ danych: String

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: aby uzyskać: *reasonname $= PutOpts*.**ReasonName**

Właściwość RecordFields

Odczyt-zapis. Flagi wskazujące, które pola mają być dostosowywane w zależności od kolejki podczas umieszczania komunikatu na liście dystrybucyjnej. Wartością początkową jest zero.

Ta właściwość odpowiada flagom PutMsgRecFields w strukturze MQPMO MQI. W interfejsie MQI te flagi kontrolują, które pola (w strukturze MQPMR) są obecne i używane przez komendę MQPUT. W obiekcie opcji MQPutMessagete pola są zawsze obecne, a więc flagi mają wpływ tylko na to, które pola są używane przez komendę Put. Więcej informacji na ten temat zawiera publikacja *Skorowidz programistyczny aplikacji WebSphere MQ* .

Zdefiniowane w:
Klasa opcji MQPutMessage

Typ danych:
Długa liczba całkowita

Składnia: Aby uzyskać: *recordfields & = PutOpts*.**RecordFields**

Aby ustawić: *PutOpts*.**RecordFields** = *recordfields &*

Właściwość ResolvedQueueManagerName

Tylko do odczytu. Pole nazwy ResolvedQMgrproduktu MQPMO. Szczegółowe informacje można znaleźć w sekcji ResolvedQMgrNazwa (MQCHAR48) . Wartością początkową jest wszystkie odstępy.

Zdefiniowane w: klasa opcji MQPutMessage

Typ danych: Łańcuch o długości 48 znaków

Składnia: aby uzyskać informacje: *qmgr $= PutOpts*.**ResolvedQueueManagerName**

Właściwość Nazwa ResolvedQueue

Tylko do odczytu. Pole MQPMO ResolvedQName . Szczegółowe informacje na ten temat zawiera sekcja ResolvedQName (MQCHAR48) . Wartością początkową jest wszystkie odstępy.

Zdefiniowane w: klasa opcji MQPutMessage

Typ danych: Łańcuch o długości 48 znaków

Składnia: aby uzyskać: *qname $= PutOpts*.**ResolvedQueueNazwa**

ClearError-metoda kodów

Resetuje parametr CompletionCode do wartości MQCC_OK i ReasonCode na wartość MQRC_NONE zarówno dla klasy opcji MQPutMessage, jak i klasy MQSession.

Zdefiniowane w: klasa opcji MQPutMessage

Składnia: Wywołanie *PutOpts*.**ClearErrorCodes ()**

Klasa opcji MQGetMessage

Ta klasa obudowuje różne opcje, które sterują działaniem pobierania komunikatu z kolejki produktu WebSphere MQ .

Zawieranie

Klasa opcji MQGetMessagejest zawarta w klasie MQSession.

Tworzenie

Opcja **Nowy** powoduje utworzenie nowego obiektu opcji MQGetMessagei ustawia wszystkie jego właściwości na wartości początkowe.

Alternatywnie można użyć metody AccessGetMessageOptions klasy MQSession.

Właściwości

- "Właściwość CompletionCode" na stronie 1124
- "Właściwość MatchOptions" na stronie 1124
- "Właściwość opcji" na stronie 1125
- "Właściwość ReasonCode" na stronie 1125
- "Właściwość ReasonName" na stronie 1125
- "Właściwość Nazwa ResolvedQueue" na stronie 1125
- "Właściwość WaitInterval" na stronie 1125

Metody

- "ClearError-metoda kodów" na stronie 1125

Składnia

Dim *gmo* **Jako nowe opcje MQGetMessage** lub

Ustaw *gmo* = **Nowe opcje MQGetMessage**

Właściwość CompletionCode

Tylko do odczytu. Zwraca kod zakończenia ustawiony przez ostatnią metodę lub dostęp do właściwości wystawiony dla obiektu.

Zdefiniowana w: MQGetMessage-Klasa opcji.

Typ danych: Long

Wartości:

- MQCC_OK
- MQCC_WARNING,
- MQCC_FAILED

Składnia: Aby uzyskać: *completioncode & = GetOpts*.**CompletionCode**

Właściwość MatchOptions

Odczyt-zapis. Opcje kontrolujące kryteria wyboru używane dla komendy MQGET. Wartością początkową jest MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID.

Zdefiniowane w:
Klasa opcji MQGetMessage

Typ danych:
Długa liczba całkowita

Wartości:
Patrz MatchOptions (MQLONG).

Składnia: Aby uzyskać informacje: *matchoptions & = GetOpts*.**MatchOptions**

Aby ustawić: *GetOpts*.**MatchOptions** =*matchoptions &*

Właściwość opcji

Odczyt-zapis. Pole Opcje MQGMO. Szczegółowe informacje na ten temat zawiera sekcja Opcje . Wartością początkową jest MQGMO_NO_WAIT.

Zdefiniowana w: MQGetMessage-Klasa opcji.

Typ danych: Long

Składnia: Do pobrania: *opcje & = GetOpts*.**Opcje** Do ustawienia: *GetOpts*.**Opcje** = *opcje &*

Właściwość ReasonCode

Tylko do odczytu. Zwraca kod przyczyny ustawiony przez ostatnią metodę lub dostęp do właściwości wystawiony dla obiektu.

Zdefiniowane w: klasa opcji MQGetMessage

Typ danych: Long

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: do pobrania: *kod_przyczyny & = GetOpts*.**ReasonCode**

Właściwość ReasonName

Tylko do odczytu. Zwraca nazwę symboliczną najnowszego kodu przyczyny. Na przykład: "MQRC_QMGR_NOT_AVAILABLE". **Zdefiniowane w:** klasa opcji MQGetMessage

Typ danych: String

Wartości:

- Patrz: kody przyczyny interfejsu API.

Składnia: aby uzyskać następujące informacje: *reasonname $= MQGetMessageOptions*.**ReasonName**

Właściwość Nazwa ResolvedQueue

Tylko do odczytu. Pole MQGMO ResolvedQName . Szczegółowe informacje na ten temat zawiera sekcja ResolvedQName (MQCHAR48) . Wartością początkową jest wszystkie odstępy.

Zdefiniowane w: klasa opcji MQGetMessage

Typ danych: Łańcuch o długości 48 znaków

Składnia: aby uzyskać: *qname $= GetOpts*.**ResolvedQueueNazwa**

Właściwość WaitInterval

Odczyt/zapis. Pole MQGMO WaitInterval . Maksymalny czas (w milisekundach) oczekiwania na nadejście odpowiedniego komunikatu. W przypadku, gdy zażądano działania wait, właściwość Options. To pole ma wartość początkową równą 0. Szczegółowe informacje na temat opcji MQGMO można znaleźć w sekcji MQGMO.

Zdefiniowane w: klasa opcji MQGetMessage

Typ danych: Long

Składnia: Aby uzyskać: *wait & = GetOpts*.**WaitInterval**

Aby ustawić: *GetOpts*.**WaitInterval** = wait &

ClearError-metoda kodów

Resetuje parametr CompletionCode do wartości MQCC_OK i ReasonCode na wartość MQRC_NONE zarówno dla klasy opcji MQGetMessage, jak i klasy MQSession.

Zdefiniowane w: klasa opcji MQGetMessage

Składnia: Wywołanie *GetOpts*.**ClearErrorCodes ()**

Klasa MQDistributionList

Ta klasa hermetyzuje kolekcję kolejek-lokalnych, zdalnych lub aliasów dla danych wyjściowych.

Tworzenie

new -tworzy nowy obiekt MQDistributionList .

Alternatywnie można użyć metody AddDistributionList klasy MQQueueManager .

Właściwości

Metoda

Składnia

Dim *distlist.***As New MQDistributionList** lub **Set** *distlist* = **New MQDistributionList**

Właściwość identyfikatora AlternateUser

Odczyt-zapis. Alternatywny identyfikator użytkownika używany do sprawdzania poprawności dostępu do listy kolejek po ich otwarciu.

Zdefiniowane w:
Klasa MQDistributionList

Typ danych:
Łańcuch o długości 12 znaków

Składnia: Aby uzyskać: *altuser $= MQDistributionList*.**AlternateUserId**

Aby ustawić: *MQDistributionList*.**AlternateUserId** = *altuser $*

Właściwość CloseOptions

Odczyt-zapis. Opcje używane do sterowania tym, co się dzieje, gdy lista dystrybucyjna jest zamknięta. Wartością początkową jest MQCO_NONE.

Zdefiniowane w:
Klasa MQDistributionList

Typ danych:
Długa liczba całkowita

Wartości:

- MQCO_NONE
- MQCO_DELETE
- MQCO_DELETE_PURGE

Składnia: Do pobrania: *closeopt & = MQDistributionList*.**CloseOptions**

Aby ustawić: *MQDistributionList*.**CloseOptions** = *closeopt &*

Właściwość CompletionCode

Tylko do odczytu. Kod zakończenia ustawiony przez ostatnią metodę lub dostęp do właściwości wystawiony dla obiektu.

Zdefiniowane w:
Klasa MQDistributionList

Typ danych:
Długa liczba całkowita

Wartości:

- MQCC_OK
- MQCC_WARNING,
- MQCC_FAILED

Składnia: do pobrania: *completioncode & = MQDistributionList*.**CompletionCode**

Właściwość ConnectionReference

Odczyt-zapis. Menedżer kolejek, do którego należy lista dystrybucyjna.

Zdefiniowane w:
Klasa MQDistributionList

Typ danych:
MQQueueManager

Składnia: Aby uzyskać: *set queuemanager = MQDistributionList*.**ConnectionReference**

Aby ustawić: *set MQDistributionList*. **ConnectionReference** = *menedżer_kolejek*

Właściwość FirstDistributionListItem

Tylko do odczytu. Pierwszy obiekt pozycji listy dystrybucyjnej powiązany z listą dystrybucyjną.

Zdefiniowane w:
Klasa MQDistributionList

Typ danych:
Element MQDistributionList

Wartości:

Składnia: Aby uzyskać: *set distributionlistitem = MQDistributionList*.**FirstDistributionListItem**

Właściwość IsOpen

Tylko do odczytu.

Zdefiniowane w:
Klasa MQDistributionList

Typ danych:
wartość boolowska

Wartości:

- PRAWDA (-1)
- FALSE (0)

Składnia: aby uzyskać: *IsOpen = MQDistributionList*.**IsOpen**

Właściwość OpenOptions

Odczyt-zapis. Opcje, które mają być używane, gdy lista dystrybucyjna jest otwarta.

Zdefiniowane w:
Klasa MQDistributionList

Typ danych:
Długa liczba całkowita

Wartości:
Więcej informacji zawiera sekcja Opcje MQPMO.

Składnia: Do pobrania: *openopt & = MQDistributionList*.**OpenOptions**

Aby ustawić: *MQDistributionList.***OpenOptions** = openopt &

Właściwość ReasonCode

Tylko do odczytu. Kod przyczyny ustawiony przez ostatnią metodę lub dostęp do właściwości wystawiony dla obiektu.

Zdefiniowane w:
Klasa MQDistributionList

Typ danych:
Długa liczba całkowita

Wartości:
Patrz: kody przyczyny interfejsu API.

Składnia: do pobrania: *kod_przyczyny & = MQDistributionList*.**ReasonCode**

Właściwość ReasonName

Tylko do odczytu. Nazwa symboliczna dla parametru ReasonCode. Na przykład "MQRC_QMGR_NOT_AVAILABLE".

Zdefiniowane w:
Klasa MQDistributionList

Typ danych:
Łańcuch

Wartości:
Patrz: kody przyczyny interfejsu API.

Składnia: aby uzyskać: *reasonname $= MQDistributionList*.**ReasonName**

Metoda AddDistributionListItem

Tworzy nowy obiekt elementu MQDistributionListi wiąże go z obiektem listy dystrybucyjnej. Parametr nazwy kolejki jest obowiązkowy.

Właściwość DistributionList pozycji listy dystrybucyjnej jest ustawiana na listę dystrybucyjną, a właściwość FirstDistributionListItem listy dystrybucyjnej jest ustawiona tak, aby odwoła się do tej nowej pozycji listy dystrybucyjnej.

Dla nowej pozycji listy dystrybucyjnej właściwość PreviousDistributionListItem jest ustawiona na nic, a właściwość NextDistributionListItem jest ustawiona tak, aby odwoła się do pozycji listy dystrybucyjnej,

która była wcześniej pierwsza, lub nic, jeśli wcześniej nie było (to znaczy, że nowy jest wstawiany przed tymi, które istnieją już).

Spowoduje to zwrócenie błędu, jeśli lista dystrybucyjna jest otwarta.

Zdefiniowane w:
Klasa MQDistributionList

Składnia: set distributionlistitem = *MQDistributionList*.**AddDistributionListItem** (QName$, QMgrName$)

Parametry:

QName$ Łańcuch. Nazwa kolejki produktu WebSphere MQ .

QMgrName$ Łańcuch. Nazwa menedżera kolejek produktu WebSphere MQ .

ClearError-metoda kodów

Resetuje parametr CompletionCode do wartości MQCC_OK i ReasonCode na wartość MQRC_NONE zarówno dla klasy MQDistributionList , jak i klasy MQSession.

Zdefiniowane w:
Klasa MQDistributionList

Składnia: wywołaj *MQDistributionList*.**ClearErrorCodes**()

Metoda zamknięcia

Zamyka listę dystrybucyjną przy użyciu bieżącej wartości opcji Zamknij.

Zdefiniowane w:
Klasa MQDistributionList

Składnia: Wywołaj *MQDistributionList*.**Zamknij**()

Metoda otwierania

Otwiera każdą z kolejek określonych za pomocą właściwości QueueName i (tam gdzie jest to odpowiednie) QueueManagernazw pozycji listy dystrybucyjnej powiązanych z bieżącym obiektem przy użyciu bieżącej wartości identyfikatora AlternateUser.

Zdefiniowane w:
Klasa MQDistributionList

Składnia: wywołaj *MQDistributionList*.**Otwórz**()

metoda PUT

Umieszcza komunikat w każdej z kolejek identyfikowanych przez pozycje listy dystrybucyjnej powiązane z listą dystrybucyjną.

Zdefiniowane w:

Klasa MQDistributionList

Składnia

Wywołaj komendę MQDistributionList.**Put**(Message, PutMsgOptions &)

Parametry

Komunikat Obiekt MQMessage reprezentujący komunikat, który ma zostać umieszczony.

PutMsgOpcje MQPutMessageObiekt opcji zawierający opcje służące do sterowania operacją put. Jeśli nie zostanie podana, zostaną użyte domyślne opcje PutMessage.

Ta metoda przyjmuje obiekt MQMessage jako parametr. Następujące właściwości elementu listy dystrybucyjnej mogą zostać zmienione w wyniku użycia tej metody:

- CompletionCode
- ReasonCode
- ReasonName
- MessageId
- MessageIdHex
- CorrelationId
- CorrelationIdHex
- GroupId
- GroupIdHex
- Opinie
- AccountingToken
- AccountingTokenHex

MQDistributionList-klasa elementu

Ta klasa hermetykuje struktury MQOR, MQRR i MQPMR i wiąże je z listą dystrybucyjną będącą właścicielem.

Tworzenie

Należy użyć metody AddDistributionListItem klasy MQDistributionList .

Właściwości

Metody

- "Właściwość AccountingToken" na stronie 1131.
- "Właściwość Hex AccountingToken" na stronie 1131.
- "Właściwość CompletionCode" na stronie 1132.
- "Właściwość CorrelationId" na stronie 1132.
- "Właściwość CorrelationIdHex" na stronie 1132.
- "Właściwość DistributionList" na stronie 1132.
- "Właściwość sprzężenia zwrotnego" na stronie 1133.
- "Właściwość GroupId" na stronie 1133.
- "Właściwość GroupIdHex" na stronie 1133.
- "Właściwość MessageId" na stronie 1133.
- "Właściwość Hex MessageId" na stronie 1134.
- "Właściwość NextDistributionListItem" na stronie 1134.
- "Właściwość PreviousDistributionListItem" na stronie 1134.
- "Właściwość Nazwa QueueManager" na stronie 1134.
- "Właściwość QueueName" na stronie 1135.
- "Właściwość ReasonCode" na stronie 1135.
- "Właściwość ReasonName" na stronie 1135.
- "ClearError-metoda kodów" na stronie 1135.

Właściwości:

- Właściwość AccountingToken
- Właściwość Hex AccountingToken
- Właściwość CompletionCode
- Właściwość CorrelationId
- Właściwość CorrelationIdHex
- Właściwość DistributionList
- Właściwość sprzężenia zwrotnego
- Właściwość GroupId
- Właściwość GroupIdHex
- Właściwość MessageId
- Właściwość Hex MessageId
- Właściwość NextDistributionListItem
- Właściwość PreviousDistributionListItem
- Właściwość Nazwa QueueManager
- Właściwość QueueName
- Właściwość ReasonCode
- Właściwość ReasonName

Metody:

- ClearError-metoda kodów

Tworzenie:

Należy użyć metody AddDistributionListItem klasy MQDistributionList .

Właściwość AccountingToken

Odczyt-zapis. Element AccountingToken , który ma zostać umieszczony w kolejce komunikatów MQPMR komunikatu podczas umieszczania w kolejce. Jej wartością początkową jest wszystkie wartości NULL.

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch zawierający 32 znaki

Składnia: Aby uzyskać: *accountingtoken $= MQDistributionListItem*.**AccountingToken**

Aby ustawić: *MQDistributionListItem*.**AccountingToken** = *accountingtoken $*

Właściwość Hex AccountingToken

Odczyt-zapis. Element AccountingToken , który ma zostać umieszczony w kolejce komunikatów MQPMR komunikatu podczas umieszczania w kolejce.

Co dwa znaki w łańcuchu reprezentują szesnastkowy odpowiednik pojedynczego znaku ASCII. Na przykład para znaków "6" i "1" oznacza pojedynczy znak "A", a para znaków "6" i "2" reprezentuje pojedynczy znak "B" itd.

Należy podać 64 poprawne znaki szesnastkowe.

Jego początkowa wartość to "0 ... 0".

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch o długości 64 znaków szesnastkowych, który zawiera 32 znaki ASCII.

Składnia: Aby uzyskać: *accountingtokenh $= MQDistributionListItem*.**AccountingTokenHex**

Aby ustawić wartość: *MQDistributionListItem*.**AccountingTokenHex** = *accountingtokenh $*

Właściwość CompletionCode

Tylko do odczytu. Kod zakończenia ustawiony przez ostatnie otwarcie lub żądanie umieszczenia wydane dla obiektu listy dystrybucyjnej będącego właścicielem.

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Długa liczba całkowita

Wartości:

- MQCC_OK
- MQCC_WARNING,
- MQCC_FAILED

Składnia: Do pobrania: *completioncode $= MQDistributionListItem*.**CompletionCode**

Właściwość CorrelationId

Odczyt-zapis. Identyfikator CorrelId , który ma zostać włączony do rekordu MQPMR komunikatu podczas umieszczania w kolejce. Jej wartością początkową jest wszystkie wartości NULL.

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch 24 znaków

Składnia: Do pobrania: *correlid $= MQDistributionList*.**CorrelationId**

Aby ustawić wartość: *MQDistributionListItem*.**CorrelationId** = *correlid $*

Właściwość CorrelationIdHex

Odczyt-zapis. Identyfikator CorrelId , który ma zostać włączony do rekordu MQPMR komunikatu podczas umieszczania w kolejce.

Co dwa znaki w łańcuchu reprezentują szesnastkowy odpowiednik pojedynczego znaku ASCII. Na przykład para znaków "6" i "1" oznacza pojedynczy znak "A", a para znaków "6" i "2" reprezentuje pojedynczy znak "B" itd.

Należy podać 48 poprawnych znaków szesnastkowych.

Jego początkowa wartość to "0 .. 0".

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch 48 znaków szesnastkowych, reprezentujący 24 znaki ASCII.

Składnia: do pobrania: *correlidh = $= MQDistributionList*.**CorrelationIdHex**

Aby ustawić wartość: *MQDistributionListItem*.**CorrelationIdHex** = *correlidh $*

Właściwość DistributionList

Tylko do odczytu. Lista dystrybucyjna, z którą powiązany jest ten element listy dystrybucyjnej.

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
MQDistributionList

Składnia: Do pobrania: *set distributionlist = MQDistributionList*.**DistributionList**

Właściwość sprzężenia zwrotnego

Odczyt-zapis. Wartość sprzężenia zwrotnego, która ma być zawarta w komunikacie MQPMR komunikatu umieszczanego w kolejce.

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Długa liczba całkowita

Wartości:
Patrz Feedback (MQLONG).

Składnia: aby uzyskać informacje: *feedback & = MQDistributionListItem*.**Feedback**

Aby ustawić: *MQDistributionListItem*.**Opinia** = *feedback &*

Właściwość GroupId

Odczyt-zapis. Element GroupId , który ma zostać włączony do rekordu MQPMR komunikatu podczas umieszczania w kolejce. Jej wartością początkową jest wszystkie wartości NULL.

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch 24 znaków

Składnia: Do pobrania: *groupid $= MQDistributionListItem*.**GroupId**

Aby ustawić wartość: *MQDistributionListItem*.**GroupId** = *groupid $*

Właściwość GroupIdHex

Odczyt-zapis. Element GroupId , który ma zostać włączony do rekordu MQPMR komunikatu podczas umieszczania w kolejce.

Co dwa znaki w łańcuchu reprezentują szesnastkowy odpowiednik pojedynczego znaku ASCII. Na przykład para znaków "6" i "1" oznacza pojedynczy znak "A", a para znaków "6" i "2" reprezentuje pojedynczy znak "B" itd.

Należy podać 48 poprawnych znaków szesnastkowych.

Jego początkowa wartość to "0 .. 0".

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch o długości 48 znaków szesnastkowych, który zawiera 24 znaki ASCII.

Składnia: do pobrania: *groupidh $= MQDistributionList*.**GroupIdHex**

Aby ustawić wartość: *MQDistributionListItem*.**GroupIdHex** = *groupidh $*

Właściwość MessageId

Odczyt-zapis. Element MessageId , który ma zostać włączony do rekordu MQPMR komunikatu podczas umieszczania w kolejce. Jej wartością początkową jest wszystkie wartości NULL.

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch 24 znaków

Składnia: Do pobrania: *messageid $= MQDistributionList*.**MessageId**

Aby ustawić wartość: *MQDistributionListItem*.**MessageId** = *messageid $*

Właściwość Hex MessageId

Odczyt-zapis. Element MessageId , który ma zostać włączony do rekordu MQPMR komunikatu podczas umieszczania w kolejce.

Co dwa znaki w łańcuchu reprezentują szesnastkowy odpowiednik pojedynczego znaku ASCII. Na przykład para znaków "6" i "1" oznacza pojedynczy znak "A", a para znaków "6" i "2" reprezentuje pojedynczy znak "B" itd.

Należy podać 48 poprawnych znaków szesnastkowych.

Jego początkowa wartość to "0 .. 0".

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch 48 znaków szesnastkowych, reprezentujący 24 znaki ASCII.

Składnia: Do pobrania: *messageidh $= MQDistributionList*.**MessageIdHex**

Aby ustawić wartość: *MQDistributionListItem*.**MessageIdHex** = *messageidh $*

Właściwość NextDistributionListItem

Tylko do odczytu. Następny obiekt pozycji listy dystrybucyjnej powiązany z tą samą listą dystrybucyjną.

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Element MQDistributionList

Składnia: Do pobrania: *set distributionlistitem = MQDistributionList.***NextDistributionListItem**

Właściwość PreviousDistributionListItem

Tylko do odczytu. Poprzedni obiekt pozycji listy dystrybucyjnej powiązany z tą samą listą dystrybucyjną.

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Element MQDistributionList

Składnia: Aby uzyskać: *set distributionlistitem = MQDistributionListItem*.**PreviousDistributionListItem**

Właściwość Nazwa QueueManager

Odczyt-zapis. Nazwa menedżera kolejek produktu WebSphere MQ .

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch o długości 48 znaków.

Składnia: Do pobrania: *qmname $= MQDistributionListItem*.**QueueManagerNazwa**

Aby ustawić wartość: *MQDistributionListItem*.**QueueManagerName** = *qmname $*

Właściwość QueueName

Odczyt-zapis. Nazwa kolejki produktu WebSphere MQ .

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch o długości 48 znaków.

Składnia: Do pobrania: *qname $= MQDistributionListItem*.**QueueName**

Aby ustawić: *MQDistributionListItem*.**QueueName** = *qname $*

Właściwość ReasonCode

Tylko do odczytu. Kod zakończenia ustawiony przez ostatnią otwartą lub wystawiona dla obiektu listy dystrybucyjnej.

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Długa liczba całkowita

Wartości:
Patrz: kody przyczyny interfejsu API.

- MQCC_OK
- MQCC_WARNING,
- MQCC_FAILED

Składnia: do pobrania: *kod_przyczyny & = MQDistributionList*.**ReasonCode**

Właściwość ReasonName

Tylko do odczytu. Nazwa symboliczna dla parametru ReasonCode. Na przykład "MQRC_QMGR_NOT_AVAILABLE".

Zdefiniowane w:
MQDistributionList-klasa elementu

Typ danych:
Łańcuch

Wartości:
Patrz: kody przyczyny interfejsu API.

Składnia: aby uzyskać: *reasonname $= MQDistributionListItem*.**ReasonName**

ClearError-metoda kodów

Resetuje parametr CompletionCode do wartości MQCC_OK i ReasonCode na wartość MQRC_NONE zarówno dla klasy elementu MQDistributionList, jak i klasy MQSession.

Zdefiniowane w:
MQDistributionList-klasa elementu

Składnia: wywołaj *MQDistributionList*.**ClearErrorKody**

Rozwiązywanie problemów

Informacje o udostępnionym narzędziu do śledzenia, wspólne pułapy i pomoc w sposobie ich unikania.

W tej sekcji opisano dostępne narzędzia śledzenia oraz szczegółowe informacje na temat pułapek, które ułatwiają ich uniknięcie:

- “Korzystanie ze śledzenia” na stronie 1136

- “Jeśli działanie skryptu WebSphere MQ Automation Classes for ActiveX nie powiedzie się” na stronie 1137
- “Kody przyczyny” na stronie 1137
- “Narzędzie na poziomie kodu” na stronie 1140

Korzystanie ze śledzenia

Produkt MQAX zawiera narzędzie do śledzenia, które ułatwia organizacji serwisu identyfikowanie tego, co się dzieje, gdy występuje problem. Zawiera on ścieżki, które zostały podjęte podczas uruchamiania skryptu MQAX. Jeśli nie masz problemu, uruchom śledzenie, aby uniknąć niepotrzebnego korzystania z zasobów systemowych.

Dostępne są trzy zmienne środowiskowe, które można ustawić w celu sterowania procesem śledzenia:

- ŚLEDZENIE OMQ_TRACE
- OMQ_TRACE_PATH
- OMQ_TRACE_LEVEL

Należy zwrócić uwagę, że podanie wartości *any* dla parametru OMQ_TRACE powoduje przełączenie tej samej wydajności śledzenia. Nawet jeśli parametr OMQ_TRACE zostanie ustawiony na OFF, śledzenie będzie nadal aktywne.

Aby wyłączyć śledzenie, nie należy określać wartości parametru OMQ_TRACE.

1. Kliknij opcję **Uruchom** .
2. Kliknij opcję **Panel sterowania** .
3. Dwukrotnie kliknij **System**
4. Kliknij opcję **Zaawansowane** .
5. Kliknij opcję **Środowisko** .
6. W sekcji zatytułowanej "Zmienne użytkownika dla (nazwa użytkownika)" kliknij przycisk **Nowy** .
7. Wprowadź nazwę zmiennej i poprawną wartość w odpowiednich polach, a następnie kliknij przycisk **OK** .
8. Kliknij przycisk **OK** , aby zamknąć okno Zmienne środowiskowe.
9. Kliknij przycisk **OK** , aby zamknąć okno Właściwości systemu.
10. Zamknij okno Panel sterowania

Podejmując decyzję o miejscu, w którym mają być zapisywane pliki śledzenia, należy upewnić się, że masz wystarczające uprawnienia do zapisu, a nie tylko do odczytu, dysku.

Po włączeniu śledzenia następuje spowolnienie działania komendy MQAX, ale nie wpływa to na wydajność środowisk ActiveX ani WebSphere MQ . Jeśli plik śledzenia nie jest już potrzebny, można go usunąć.

Aby zmienić status zmiennej OMQ_TRACE, należy zatrzymać działanie MQAX.

Nazwa i katalog pliku śledzenia

Nazwa pliku śledzenia ma postać OMQnnnnn.trc, gdzie nnnnn jest identyfikatorem procesu ActiveX uruchomionego w danym momencie.

Tabela 157. Komendy i ich efekty

Komenda	Efekt
SET OMQ_TRACE_PATH = drive: \katalog	Określa katalog, w którym ma być zapisywany plik śledzenia.
USTAW ZMIENNĄ OMQ_TRACE_PATH =	Usuwa zmienną środowiskową OMQ_PATH w bieżącym katalogu roboczym (jeśli uruchomiono element ActiveX).
ECHO %OMQ_TRACE_PATH%	Wyświetla bieżące ustawienie katalogu śledzenia w systemie Windows.

Tabela 157. Komendy i ich efekty (kontynuacja)

Komenda	Efekt
SET OMQ_TRACE = xxxxxxxx	Powoduje to włączenie śledzenia. Śledzenie można przełączać, umieszczając jeden lub więcej znaków po znaku '='. Na przykład: SET OMQ_TRACE=yes SET OMQ_TRACE = no. W obu tych przykładach śledzenie zostanie włączone. Jest to efektywne tylko dla jednego okna/sesji
SET OMQ_TRACE=	Wyłącz śledzenie
ECHO %OMQ_TRACE%	Wyświetla zawartość zmiennej środowiskowej w systemie Windows.
SET	Wyświetla zawartość wszystkich zmiennych środowiskowych w systemie Windows.
USTAW WARTOŚĆ OMQ_TRACE_LEVEL = 9	Ustawia poziom śledzenia na 9. Wartości większe niż 9 nie generują żadnych dodatkowych informacji w pliku śledzenia.

Jeśli działanie skryptu WebSphere MQ Automation Classes for ActiveX nie powiedzie się

Jeśli działanie skryptu WebSphere MQ Automation Classes for ActiveX nie powiedzie się, istnieje pewna liczba źródeł informacji.

Raport objawów pierwszego niepowodzenia

Niezależnie od narzędzia śledzenia, w przypadku nieoczekiwanych i wewnętrznych błędów, może zostać wygenerowany raport objawów pierwszego niepowodzenia.

Ten raport znajduje się w pliku o nazwie OMQnnnnn.fdc, gdzie nnnnn jest numerem procesu ActiveX , który jest uruchomiony w danym momencie. Plik ten znajduje się w katalogu roboczym, z którego został uruchomiony element ActiveX lub w ścieżce określonej w zmiennej środowiskowej OMQ_PATH.

Inne źródła informacji

Produkt WebSphere MQ udostępnia różne dzienniki błędów i informacje o śledzeniu, w zależności od używanej platformy. Zapoznaj się z dziennikiem zdarzeń aplikacji Windows NT.

Kody przyczyny

Oprócz dokumentów opisanych w produkcie WebSphere MQ MQI mogą występować następujące kody przyczyny. Inne kody można znaleźć w dzienniku zdarzeń aplikacji produktu WebSphere MQ .

Tabela 158. Kody przyczyny i to, co oznaczają

Kod przyczyny	Wyjaśnienie
MQRC_LIBRARY_LOAD_ERROR (6000)	Nie można załadować jednej lub większej liczby bibliotek produktu WebSphere MQ . Sprawdź, czy wszystkie biblioteki produktu WebSphere MQ znajdują się w poprawnej ścieżce wyszukiwania w systemie, z którego korzysta użytkownik. Na przykład należy upewnić się, że katalogi zawierające biblioteki produktu WebSphere MQ znajdują się w zmiennej PATH.
MQRC_CLASS_LIBRARY_ERROR (6001)	Jeden z wywołań biblioteki klas produktu WebSphere MQ zwrócił nieoczekiwany kod ReasonCode/CompletionCode. Szczegółowe informacje można znaleźć w raporcie objawów pierwszego niepowodzenia. Zanotuj ostatnio używaną metodę/właściwość i klasę, a następnie poinformuj IBM o problemie.
MQRC_STRING_LENGTH_TOO_BIG (6002)	Podjęto próbę zapisu łańcucha formatu UTF o długości większej niż 65,535 bajtów do buforu komunikatów.
MQRC_WRITE_VALUE_ERROR (6003)	Używana jest wartość spoza zakresu, na przykład msg.WriteByte (240).

Tabela 158. Kody przyczyny i to, co oznaczają (kontynuacja)

Kod przyczyny	Wyjaśnienie
MQRC_PACKED_DECIMAL_ERROR (6004)	Podjęto próbę odczytu upakowanej liczby dziesiętnej z buforu komunikatów, ale dane w wskaźniku danych nie są w poprawnym formacie spakowanego danych.
MQRC_FLOAT_CONVERSION_ERROR (6005)	Próbowano odczytać pojedynczy lub podwójny numer zmiennopozycyjny z buforu komunikatów, ale dane w wskaźniku danych nie są w odpowiednim formacie zmiennopozycyjnym.
MQRC_REOPEN_EXCL_INPUT_ERROR (6100)	Otwarty obiekt nie ma poprawnego **OpenOptions** i wymaga co najmniej jednej dodatkowej opcji. Niejawne ponowne otwarcie jest wymagane, ale uniemożliwiono zamknięcie. Ustaw opcję **OpenOptions**w sposób jawny, tak aby obejmował wszystkie ewentualności, tak aby niejawne ponowne otwarcie nie było wymagane. Zamknięcie zostało zablokowane, ponieważ kolejka jest otwarta do wyłącznego wejścia, a zamknięcie może spowodować, że inne osoby będą miały możliwość uzyskania dostępu do kolejki.
MQRC_REOPEN_INQUIRE_ERROR (6101)	Otwarty obiekt nie ma poprawnego OpenOptions i wymaga co najmniej jednej dodatkowej opcji. Niejawne ponowne otwarcie jest wymagane, ale uniemożliwiono zamknięcie. Ustaw jawnie parametr OpenOptions , tak aby zawierał MQOO_INQUIRE. Zamknięcie zostało uniemożliwione, ponieważ co najmniej jedna właściwość obiektu musi zostać sprawdzona dynamicznie przed zamknięciem, a OpenOptions nie zawiera już komendy MQOO_INQUIRE.
MQRC_REOPEN_SAVED_CONTEXT_ERR (6102)	Otwarty obiekt nie ma poprawnego OpenOptions i wymaga co najmniej jednej dodatkowej opcji. Niejawne ponowne otwarcie jest wymagane, ale uniemożliwiono zamknięcie. Ustaw OpenOptions w sposób jawny, tak aby obejmował wszystkie ewentualności, tak aby niejawne ponowne otwarcie nie było wymagane. Zamknięcie zostało zablokowane, ponieważ kolejka jest otwarta za pomocą komendy MQOO_SAVE_ALL_CONTEXT, a destruktywna operacja Get została wcześniej wykonana. Spowodowało to, że zachowana informacja o stanie została powiązana z otwartą kolejką, a informacje te zostaną zniszczone przez zamknięcie.
MQRC_REOPEN_TEMPORARY_Q_ERROR (6103)	Otwarty obiekt nie ma poprawnego **OpenOptions** i wymaga co najmniej jednej dodatkowej opcji. Wymagane jest niejawne ponowne otwarcie, ale uniemożliwiono zamknięcie. Ustaw opcję **OpenOptions**w sposób jawny, tak aby obejmował wszystkie ewentualności, tak aby niejawne ponowne otwarcie nie było wymagane. Zamknięcie zostało zablokowane, ponieważ kolejka jest kolejką lokalną o typie definicji MQQDT_TEMPORARY_DYNAMIC, która zostanie zniszczona przez zamknięcie.
MQRC_ATTRIBUTE_LOCKED (6104)	Podjęto próbę zmiany wartości lub atrybutu obiektu, gdy obiekt jest otwarty. Niektóre atrybuty, takie jak **AlternateUserId**, nie mogą być zmieniane, gdy obiekt jest otwarty.
MQRC_CURSOR_NOT_VALID (6105)	Kursor przeglądania dla otwartej kolejki został unieważniony, ponieważ był ostatnio używany przez niejawne ponowne otwarcie. Ustaw OpenOptions w sposób jawny, tak aby obejmował wszystkie ewentualności, tak aby niejawne ponowne otwarcie nie było wymagane.
MQRC_ENCODING_ERROR (6106)	Kodowanie następnego elementu komunikatu musi mieć wartość MQENC_NATIVE dla odczytu.
BŁĄD MQRC_STRUCID_ERROR (6107)	Struktura identyfikatora następnego elementu komunikatu, który pochodzi z 4 znaków zaczynając od wskaźnika danych, jest albo brakująca, albo jest niespójna z typem zmiennej, do której element jest odczytywający.
MQRC_NULL_POINTER (6108)	Podano pusty wskaźnik, w którym wymagany lub domniemany wskaźnik niezerowy jest wymagany. Może to być spowodowane używaniem jawnych deklaracji dla obiektów WebSphere MQ używanych z VBA jako parametrów do wywołań (na przykład komunikat dim jako Obiekt to ok, komunikat dim jako MqMessage może powodować problemy). Na przykład w programie Excel z określonym q i ustawionym komunikatem dim jako MqMessageq.put msg podaje reasonCode MQRC_NULL_POINTER. Działa on poprawnie z poziomu VisualBasic.

Tabela 158. Kody przyczyny i to, co oznaczają (kontynuacja)

Kod przyczyny	Wyjaśnienie
MQRC_NO_CONNECTION_REFERENCE (6109)	Obiekt **MQQueue** utracił połączenie z obiektem **MQQueueManager**. Taka sytuacja ma miejsce, jeśli program **MQQueueManager**zostanie rozłączony. Usuń obiekt **MQQueue** .
MQRC_NO_BUFFER (6110)	Bufor nie jest dostępny. W przypadku obiektu **MQMessage** nie można przydzielić tego obiektu, ponieważ nie ma wewnętrznej niespójności w stanie obiektu, który nie powinien wystąpić.
MQRC_BINARY_DATA_LENGTH_ERROR (6111)	Długość danych binarnych jest niespójna z długością atrybutu docelowego. Wartość zero jest poprawną długością dla wszystkich atrybutów. 24 jest poprawną długością dla **CorrelationId** , a dla elementu **MessageId** 32 jest poprawną długością dla elementu **AccountingToken** .
MQRC_BUFFER_NOT_AUTOMATIC (6112)	Nie można zmienić wielkości buforu zdefiniowanego przez użytkownika i zarządzanego przez użytkownika. Ponieważ bufory komunikatów są zarządzane przez system, wskazuje to na wewnętrzną niespójność.
MQRC_INSUFFICIENT_BUFFER (6113)	Brak wystarczającej ilości miejsca w buforze po umieszczonym wskaźniku danych w celu dostosowania go do żądania. Może to być spowodowane tym, że bufor nie może być rezygnowany.
MQRC_INSUFFICIENT_DATA (6114)	Po umieszczonym wskaźniku danych nie ma wystarczających danych, aby zmieścić się w żądaniu odczytu. Zmniejsz bufor do odpowiedniej wielkości i ponownie odczytaj dane.
MQRC_DATA_OBCIĘTY (6115)	Dane zostały obcięte podczas kopiowania z jednego buforu do innego. Może to być spowodowane tym, że nie można zmienić wielkości buforu docelowego, lub ponieważ wystąpił problem z adresowaniem jednego lub innego buforu, lub dlatego, że bufor jest malejący z mniejszym wymianą.
MQRC_ZERO_LENGTH (6116)	Podano zerową długość, w przypadku której wymagana lub domniemana długość dodatnia jest wymagana.
MQRC_NEGATIVE_LENGTH (6117)	Podano ujemną długość, w której wymagana jest długość zerowa lub dodatnia.
MQRC_NEGATIVE_OFFSET (6118)	Podano ujemne przesunięcie w przypadku, gdy wymagane jest przesunięcie zerowe lub dodatnie.
MQRC_INCONSISTENT_FORMAT (6119)	Format następnego elementu komunikatu jest niespójny z typem zmiennej, do której element jest odczytyowany.
MQRC_INCONSISTENT_OBJECT_STATE (6120)	Między tym obiektem jest niespójność, która jest otwarta, a przywoływany obiekt MQQueueManager , który nie jest połączony.
MQRC_CONTEXT_OBJECT_NOT_VALID (6121)	Odwołanie do kontekstu opcji MQPutMessagenie odwołuje się do poprawnego obiektu MQQueue. Obiekt został wcześniejszy zniszczony.
MQRC_CONTEXT_OPEN_ERROR (6122)	Odwołanie do kontekstu opcji MQPutMessageodwołuje się do obiektu MQQueue, który nie może zostać otwarty w celu ustanowienia kontekstu. Może to być spowodowane tym, że obiekt MQQueue ma nieodpowiednie opcje otwarcia. Sprawdź przywoływany kod przyczyny obiektu, aby ustalić przyczynę.
Błąd MQRC_STRUC_LENGTH_ERROR (6123)	Długość wewnętrznej struktury danych jest niespójna z jej treścią. W przypadku wartości MQRMH długość nie jest wystarczająca, aby pomieścić pola stałe i wszystkie dane przesunięcia.
MQRC_NOT_CONNECTED (6124)	Metoda nie powiodła się, ponieważ wymagane połączenie z menedżerem kolejek nie było dostępne, a połączenie nie może zostać nawiązane niejawnie.
MQRC_NOT_OPEN (6125)	Metoda nie powiodła się, ponieważ obiekt WebSphere MQ nie był otwarty, a otwarcie nie może być wykonane niejawnie.
MQRC_DISTRIBUTION_LIST_EMPTY (6126)	Otwarcie obiektu MQDistributionList nie powiodło się, ponieważ na liście dystrybucyjnej nie ma obiektów MQDistributionList Item. Czynność naprawczy: dodaj co najmniej jeden obiekt elementu MQDistributionListdo listy dystrybucyjnej.

Tabela 158. Kody przyczyny i to, co oznaczają (kontynuacja)	
Kod przyczyny	**Wyjaśnienie**
MQRC_INCONSISTENT_OPEN_OPTIONS (6127)	Metoda nie powiodła się, ponieważ obiekt jest otwarty, a opcje otwarcia są niespójne z wymaganą operacją. Czynność naprawczy: otwórz obiekt z odpowiednimi opcjami otwarcia i spróbuj ponownie.
MQRC_WRONG_VERSION (6128)	Metoda nie powiodła się, ponieważ podany lub napotkany numer wersji jest niepoprawny lub nie jest obsługiwany.

Narzędzie na poziomie kodu

Zespół serwisowy IBM może zapytać o poziom zainstalowanego kodu.

Aby to sprawdzić, uruchom program narzędziowy MQAXLEV.

W wierszu komend przejdź do katalogu zawierającego plik MQAX200.dll lub dodaj pełną długość ścieżki i wpisz:

```
MQAXLev MQAX200.dll > MQAXLEV.OUT
```

gdzie MQAXLEV.OUT jest nazwą pliku wyjściowego.

Jeśli zbiór wyjściowy nie zostanie określony, na ekranie zostaną wyświetlone szczegóły.

Przykładowy plik wyjściowy z narzędzia na poziomie kodu został szczegółowo opisany w poniższym przykładzie:

Przykładowy plik wyjściowy z narzędzia na poziomie kodu

```
5639-B43 (C) Copyright IBM Corp. 1996, 2024. ALL RIGHTS RESERVED.
     ***** Code Level is 5.1 ***** lib/mqole/mqole.cpp, mqole, p000, p000 L981119 1.8 98/08/21
lib/mqlsx/gmqdyn0a.c, mqlsx, p000, p000 L990212  1.6 99/02/11 16:40:24
lib/mqlsx/pc/gmqdyn1p.c, mqlsx, p000, p000 L990212  1.6 99/02/11 16:44:14
lib/mqlsx/xmqcsa.c, mqole, p000, p000 L990216             1.3 99/02/15 13:24:34
lib/mqlsx/xmqfdca.c, mqlsx, p000, p000 L990212               1.3 99/02/11 16:40:35
lib/mqlsx/xmqtrca.c, mqlsx, p000, p000 L990212            1.5 99/02/11 16:12:02
lib/mqlsx/xmqutila.c, mqlsx, p000, p000 L990212           1.3 99/02/11 16:40:40
lib/mqlsx/xmqutl1a.c, mqlsx, p000, p000 L990212           1.4 99/02/11 16:40:30
lib/mqlsx/xmqcnv1a.c, mqlsx, p000, p000 L990212           1.9 99/02/11 16:40:56
lib/mqlsx/xmqmsg.c, mqole, p000, p000 L990219             1.11 99/02/18 12:12:59
```

Interfejs ActiveX do interfejsu MQAI

Krótki przegląd interfejsów COM i ich zastosowanie w interfejsie MQAI zawiera sekcja “Korzystanie z interfejsu modelu obiektu komponentu (klasy automatyzacji produktu WebSphere MQ dla elementu ActiveX)” na stronie 1062.

Interfejs MQAI umożliwia aplikacjom budowanie i wysyłanie komend PCF (Programmable Command Format) bez bezpośredniego uzyskiwania i formatowania buforów o zmiennej długości wymaganych dla PCF. Więcej informacji na temat interfejsu MQAI zawiera sekcja Wprowadzenie do interfejsu administracyjnego WebSphere MQ (MQAI). Klasa MQAI ActiveX MQBag obudowuje worki danych obsługiwane przez interfejs MQAI w sposób, który może być używany w dowolnym języku, który obsługuje tworzenie obiektów COM, na przykład Visual Basic, C + +, Java i innych klientów skryptowych ActiveX .

Interfejs MQAI ActiveX jest przeznaczony do użytku z klasami MQAX udostępniających interfejs COM do interfejsu MQI. Więcej informacji na temat klas MQAX zawiera sekcja “Projektowanie aplikacji MQAX, które uzyskują dostęp do aplikacji innych niżActiveX” na stronie 1063.

Interfejs ActiveX udostępnia pojedynczą klasę o nazwie MQBag. Ta klasa jest używana do tworzenia worków danych MQAI, a jej właściwości i metody są używane do tworzenia elementów danych i pracy

z elementami danych w każdej z nich. Metoda Execute MQBag Execute wysyła dane o torbie do menedżera kolejek produktu WebSphere MQ jako komunikat PCF i zbiera odpowiedzi.

Aby uzyskać więcej informacji na temat klasy MQBag, jej właściwości i metod, patrz “Klasa MQBag” na stronie 1141.

Komunikat PCF jest wysyłany do określonego obiektu menedżera kolejek, opcjonalnie przy użyciu określonych kolejek żądań i odpowiedzi. Odpowiedzi są zwracane w nowym obiekcie MQBag. Pełny zestaw komend i odpowiedzi jest opisany w sekcji Definicje formatów komend programowalnych. Komendy mogą być wysyłane do dowolnego menedżera kolejek w sieci WebSphere MQ , wybierając odpowiednie kolejki żądań i odpowiedzi.

Klasa MQBag

Klasa, MQBag, jest używana do tworzenia obiektów MQBag zgodnie z wymaganiami. Po utworzeniu instancji klasa MQBag zwraca nowe odwołanie do obiektu MQBag.

Utwórz obiekt MQBag w języku Visual Basic w następujący sposób:

```
Dim mqbag As MQBag
Set mqbag = New MQBag
```

Właściwość MQBag

Właściwości obiektów MQBag zostały wyjaśnione na następującej liście:

- “Właściwość elementu” na stronie 1142.
- “Właściwość liczebności” na stronie 1143.
- “Właściwość opcji” na stronie 1143.

Metody MQBag

Metody obiektów MQBag są wyjaśnione na następującej liście:

- “Dodaj metodę” na stronie 1144.
- “Metoda AddInquiry” na stronie 1144.
- “Wyczyść metodę” na stronie 1145.
- “Metoda execute” na stronie 1145.
- “Metoda FromMessage” na stronie 1146.
- “Metoda ItemType” na stronie 1146.
- “metoda usuwania” na stronie 1147.
- “Metoda selektora” na stronie 1147.
- “Metoda ToMessage” na stronie 1148.
- “Metoda obcinania” na stronie 1148.

Obsługa błędów

Jeśli podczas operacji na obiekcie MQBag wykryty zostanie błąd, w tym te, które zostały zwrócone do tego obiektu przez bazowy obiekt MQAX lub MQAI, zgłaszany jest wyjątek błędu. Klasa MQBag obsługuje interfejs COM ISupportErrorInfo, więc następujące informacje są dostępne dla procedury obsługi błędów:

- Numer błędu: składający się z kodu przyczyny WebSphere MQ dla wykrytego błędu i kodu narzędzia COM. Pole obiektu, zgodnie ze standardem COM, wskazuje obszar odpowiedzialności za błąd. W przypadku błędów wykrytych przez produkt WebSphere MQ zawsze jest to FACILITY_ITF.
- Źródło błędu: identyfikuje typ i wersję obiektu, który wykrył błąd. W przypadku błędów wykrytych podczas operacji MQBag źródłem błędu jest zawsze MQBag.MQBag1.

- Opis błędu: łańcuch zawierający nazwę symboliczną dla kodu przyczyny produktu WebSphere MQ .

Sposób uzyskiwania dostępu do informacji o błędzie zależy od języka skryptowego. Na przykład w języku Visual Basic informacje są zwracane w obiekcie Err, a kod przyczyny WebSphere MQ jest uzyskiwany przez odjęcie stałego błędu vbObjectz poziomu Err.Number.

ReasonCode = Err.Number -Błąd vbObject

Jeśli komunikat wykonania operacji MQBag Execute wysyła komunikat PCF i odpowiedź zostanie odebrana, operacja zostanie uznana za zakończonej powodzeniem, mimo że wysłana komenda mogła się nie powiodła. W takim przypadku sam worek odpowiedzi zawiera kody przyczyny zakończenia i przyczyny błędu, zgodnie z opisem w sekcji Definicje formatów komend programowalnych .

Właściwość elementu

Przeznaczenie

Właściwość Item reprezentuje element w torbie. Jest on używany do ustawiania lub sprawdzania wartości elementu. Użycie tej właściwości odpowiada następującym wywoławczym MQAI:

- "ŁańcuchmqSet"
- "mqSetLiczba całkowita"
- "mqInquireLiczba całkowita"
- "ŁańcuchmqInquire"
- "TorbamqInquire"

w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

Element (Selector, ItemIndex, Value)

Parametry

Selector **(VARIANT)-dane wejściowe**
Selektor elementu, który ma zostać ustawiony lub wypytany.

Po zapytaniu o element jest to wartość domyślna MQSEL_ANY_USER_SELECTOR. Podczas ustawiania elementu domyślnie jest używana lista MQIA_LIST lub MQCA_LIST.

Jeśli wartość `Selector` nie jest typu long, wyniki MQRC_SELECTOR_TYPE_ERROR.

Ten parametr jest opcjonalny.

ItemIndex **(LONG)-dane wejściowe**
Ta wartość identyfikuje wystąpienie elementu określonego selektora, który ma zostać ustawiony lub zrzekłany. Wartość MQIND_NONE jest wartością domyślną.

Ten parametr jest opcjonalny.

Value **(VARIANT)-wejście/wyjście**
Zwracana wartość lub wartość, która ma zostać ustawiona. Po zapytaniu o element wartość zwracana może być typu long, string lub MQBag. Jednak podczas ustawiania elementu wartość musi być typu long lub string (jeśli nie), MQRC_ITEM_VALUE_ERROR powoduje błąd.

Wizualne wywołanie języka podstawowego

Podczas uzyskiwania informacji o wartości elementu w obrębie torby:

```
Value = mqbag[.Item]([Selector],
[ItemIndex])
```

Dla odwołań do MQBag:

```
Set abag = mqbag[.Item]([Selector].
[ItemIndex])
```

Aby ustawić wartość elementu w worku:

```
mqbag[.Item]([Selector],
[ItemIndex]) = Value
```

Właściwość liczebności

Przeznaczenie

Właściwość Liczba reprezentuje liczbę elementów danych w torbie. Ta właściwość odpowiada wywołaniu MQAI, "mqCountElementów" w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

Liczba (*Selector, Value*)

Parametry

***Selector* (VARIANT)-dane wejściowe**
Selektor elementów danych, które mają być uwzględnione w liczbie.

Wartość MQSEL_ALL_USER_SELECTORS jest wartością domyślną.

Jeśli wartość `Selector` nie jest typu long, zwracana jest wartość MQRC_SELECTOR_TYPE_ERROR.

***Value* (LONG)-dane wyjściowe**
Liczba elementów w torbie uwzględnionych przez *Selector*.

Wizualne wywołanie języka podstawowego

Aby zwrócić liczbę elementów w torbie:

```
ItemCount = mqbag.Count([Selector])
```

Właściwość opcji

Przeznaczenie

Właściwość Opcje ustawia opcje dla użycia torby. Ta właściwość odpowiada parametrowi `Options` wywołania MQAI, "mqCreateBag," w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

Opcje (*Options*)

Parametry

***Options* (LONG)-wejście/wyjście**
Opcje torby.

Uwaga: Opcje torby muszą być ustawione jako ***przed*** elementy danych są dodawane do lub ustawiane w worku. Jeśli opcje są zmieniane, gdy worek nie jest pusty, wyniki MQRC_OPTIONS_ERROR są wyświetlane. Ma to zastosowanie nawet wtedy, gdy worek jest następnie wyczyszczony.

Wizualne wywołanie języka podstawowego

Po pytaniu o opcje elementu w obrębie torby:

```
Options = mqbag.Options
```

Aby ustawić opcję elementu w worku:

```
mqbag.Options = Options
```

Metody MQBag

Metody obiektów MQBag są wyjaśnione na następujących stronach.

Dodaj metodę

Przeznaczenie

Metoda Add dodaje element danych do torby. Ta metoda odpowiada wywołaniach MQAI, "mqAddInteger" i "mqAddString" w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

Dodaj (*`Value, Selector`*)

Parametry

`Value` **(VARIANT)-dane wejściowe**
Liczba całkowita lub łańcuchowa elementu danych.

`Selector` **(VARIANT)-dane wejściowe**
Selektor identyfikujący element, który ma zostać dodany.

W zależności od typu produktu `Value`wartością domyślną jest MQIA_LIST lub MQCA_LIST. Jeśli parametr `Selector` nie jest typu long, wyniki MQRC_SELECTOR_TYPE_ERROR.

Wizualne wywołanie języka podstawowego

Aby dodać element do torby:

```
mqbag.Add(Value,[Selector])
```

Metoda AddInquiry

Przeznaczenie

Metoda AddInquiry dodaje selektor określający atrybut, który ma zostać zwrócony po wysłaniu worka administracyjnego w celu wykonania komendy INQUIRE. Ta metoda odpowiada wywołaniu MQAI, "mqAddInquiry", w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

AddInquiry *(Inquiry)*

Parametry

***Inquiry* (LONG)-dane wejściowe**
Selektor atrybutu WebSphere MQ , który ma zostać zwrócony przez komendę administracyjną INQUIRE.

Wizualne wywołanie języka podstawowego

Aby skorzystać z metody AddInquiry :

```
mqbag.AddInquiry(Inquiry)
```

Wyczyść metodę

Przeznaczenie

Metoda Clear usuwa wszystkie elementy danych z torby. Ta metoda odpowiada wywołaniu MQAI, "mqClearBag,", w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

Wyczyść

Wizualne wywołanie języka podstawowego

Aby usunąć wszystkie dane itmes z torby:

```
mqbag.Clear
```

Metoda execute

Przeznaczenie

Metoda Execute wysyła komunikat komendy administracyjnej do serwera komend i oczekuje na wszystkie komunikaty odpowiedzi. Ta metoda odpowiada wywołaniu MQAI, "mqExecute,", w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

Wykonaj *(QueueManager, Command, OptionsBag, RequestQ, ReplyQ, ReplyBag)*

Parametry

***QueueManager* (MQQueueManager)-dane wejściowe**
Menedżer kolejek, z którym połączona jest aplikacja.

***Command* (LONG)-dane wejściowe**
Komenda do wykonania.

***OptionsBag* (MQBag)-dane wejściowe**
Torba zawierająca opcje, które wpływają na przetwarzanie wywołania.

***RequestQ* (MQQueue)-dane wejściowe**
Kolejka, w której zostanie umieszczony komunikat komendy administracyjnej.

***ReplyQ* (MQQueue)-dane wejściowe**
Kolejka, w której odbierane są wszystkie komunikaty odpowiedzi.

***ReplyBag* (MQBag)-dane wyjściowe**
Odwołanie do torby zawierające dane z komunikatów odpowiedzi.

Wizualne wywołanie języka podstawowego

Aby wysłać komunikat komendy administracyjnej i poczekać na wszystkie komunikaty odpowiedzi:

```
Set ReplyBag = mqbag.Execute(QueueManager, Command,
[OptionsBag],[RequestQ],[ReplyQ])
```

Metoda FromMessage

Przeznaczenie

Metoda FromMessage ładuje dane z wiadomości do torby. Ta metoda odpowiada wywołaniu MQAI, "mqBufferToBag," w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

FromMessage *(Message, OptionsBag*)

Parametry

***Message* (MQMessage)-dane wejściowe**
Komunikat zawierający dane, które mają zostać przekształcone.

***OptionsBag* (MQBag)-dane wejściowe**
Opcje służące do sterowania przetwarzaniem wywołania.

Wizualne wywołanie języka podstawowego

Aby załadować dane z wiadomości do torby:

```
mqbag.FromMessage(Message,[OptionsBag])
```

Metoda ItemType

Przeznaczenie

Metoda ItemType zwraca typ wartości w określonej pozycji w worku. Ta metoda odpowiada wywołaniu MQAI, "mqInquireItemInfo," w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

ItemType *(Selector, ItemIndex, ItemType*)

Parametry

***Selector* (VARIANT)-dane wejściowe**
Selektor identyfikujący element, który ma zostać zapytany.

Wartość MQSEL_ANY_USER_SELECTOR jest wartością domyślną. Jeśli parametr `Selector` nie jest typu long, wyniki MQRC_SELECTOR_TYPE_ERROR.

***ItemIndex* (LONG)-dane wejściowe**
Indeks pozycji do zapytania.

Wartość MQIND_NONE jest wartością domyślną.

***ItemType* (LONG)-dane wyjściowe**
Typ danych określonego elementu.

Uwaga: Należy podać parametr `Selector` , parametr `ItemIndex` lub oba te parametry. Jeśli żaden z tych parametrów nie jest obecny, wyniki komendy MQRC_PARAMETER_MISSING.

Wizualne wywołanie języka podstawowego

Aby zwrócić typ wartości:

```
ItemType = mqbag.ItemType([Selector],
[ItemIndex])
```

metoda usuwania

Przeznaczenie

Metoda usuwania usuwa element z torby. Ta metoda odpowiada wywołaniu MQAI, "mqDeleteItem," w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

Usuń *(Selector, ItemIndex*)

Parametry

***Selector* (VARIANT)-dane wejściowe**
Selektor identyfikujący element, który ma zostać usunięty.

Wartość MQSEL_ANY_USER_SELECTOR jest wartością domyślną. Jeśli parametr `Selector` nie jest typu long, wyniki MQRC_SELECTOR_TYPE_ERROR.

***ItemIndex* (LONG)-dane wejściowe**
Indeks elementu, który ma zostać usunięty.

Wartość MQIND_NONE jest wartością domyślną.

Uwaga: Należy podać parametr `Selector` , parametr `ItemIndex` lub oba te parametry. Jeśli żaden z tych parametrów nie jest obecny, wyniki komendy MQRC_PARAMETER_MISSING.

Wizualne wywołanie języka podstawowego

Aby usunąć pozycję z torby:

```
mqbag.Remove([Selector],[ItemIndex])
```

Metoda selektora

Przeznaczenie

Metoda Selector zwraca selektor określonego elementu w obrębie torby. Ta metoda odpowiada wywołaniu MQAI, "mqInquireItemInfo" w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

Selektor *(Selector, ItemIndex, OutSelector*)

Parametry

***Selector* (VARIANT)-dane wejściowe**
Selektor identyfikujący element, który ma zostać zapytany.

Wartość MQSEL_ANY_USER_SELECTOR jest wartością domyślną. Jeśli parametr `Selector` nie jest typu long, wyniki MQRC_SELECTOR_TYPE_ERROR.

***ItemIndex* (LONG)-dane wejściowe**
Indeks elementu, który ma zostać sprawdzony.

Wartość MQIND_NONE jest wartością domyślną.

***OutSelector* (VARIANT)-dane wyjściowe**
Selektor określonego elementu.

Uwaga: Należy podać parametr `Selector` , parametr `ItemIndex` lub oba te parametry. Jeśli żaden z tych parametrów nie jest obecny, wyniki komendy MQRC_PARAMETER_MISSING.

Wizualne wywołanie języka podstawowego

Aby zwrócić selektor elementu:

```
OutSelector = mqbag.Selector([Selector],
[ItemIndex])
```

Metoda ToMessage

Przeznaczenie

Metoda ToMessage zwraca odwołanie do obiektu MQMessage. Odwołanie zawiera dane z torby. Ta metoda odpowiada wywołaniu MQAI, "mqBagToBuffer" w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

ToMessage *(OptionsBag, Message)*

Parametry

***OptionsBag* (MQBag)-dane wejściowe**
Torba zawierająca opcje, które sterują przetwarzaniem metody.

***Message* (MQMessage)-dane wyjściowe**
Odwołanie do obiektu MQMessage zawierające dane z worka.

Wizualne wywołanie języka podstawowego

Aby użyć metody ToMessage , wykonaj następujące czynności:

```
Set Message = mqbag.ToMessage([OptionsBag])
```

Metoda obcinania

Przeznaczenie

Metoda obcinania zmniejsza liczbę elementów użytkownika w torbie. Ta metoda odpowiada wywołaniu MQAI, "mqTruncateBag," w Informacje dodatkowe dotyczące formatów komend programowalnych.

Format

Obetnij *(ItemCount)*

Parametry

***ItemCount* (LONG)-dane wejściowe**
Liczba elementów użytkownika, które mają pozostać w torbie po obcięciu.

Wizualne wywołanie języka podstawowego

Aby zmniejszyć liczbę elementów użytkownika w torbie:

```
mqbag.Truncate(ItemCount)
```

Informacje o przykładowych klasach automatyzacji WebSphere MQ dla ActiveX Starter

W niniejszym dodatku opisano klasy automatyzacji WebSphere MQ Automation dla ActiveX Starter, a także objaśnienia, w jaki sposób można ich używać.

Produkt WebSphere MQ for Windows udostępnia następujące przykładowe programy Visual Basic:

- MQAXTRIV.VBP
- MQAXBSRV.VBP
- MQAXDLST.VBP
- MQAXCLSS.VBP

Te przykłady są uruchamiane w Visual Basic 4 lub Visual Basic 5. Znajdą je Państwo w katalogu ... \tools\mqax\samples\vb.

W tym samym katalogu znajdą się również przykłady dla programów Microsoft Excel i html. Są to:

- MQAX.XLS
- MQAXTRIV.XLS
- MQAXTRIV.HTM

Uwaga: Jeśli używany jest produkt Visual Basic 5, **należy** wybrać i zainstalować komponent Visual Basic grid32.ocx.

Co wykazano w próbkach

W przykładach przedstawiono sposób użycia klas automatyzacji WebSphere MQ dla ActiveX do:

- Połącz z menedżerem kolejek
- Dostęp do kolejki
- Umieszczanie komunikatu w kolejce
- Pobieranie komunikatu z kolejki

Centralna część przykładu Visual Basic jest pokazana na kolejnych stronach.

“Przygotowanie do uruchomienia przykładów” na stronie 1150 i

“Obsługa błędów w próbkach” na stronie 1150

Uruchamianie przykładów ActiveX Starter

Przed uruchomieniem klas automatyzacji WebSphere MQ dla ActiveX Starter należy sprawdzić, czy jest uruchomiony domyślny menedżer kolejek i czy zostały utworzone wymagane definicje

kolejek. Szczegółowe informacje na temat tworzenia i uruchamiania menedżera kolejek oraz tworzenia kolejki można znaleźć w sekcji Administrowanie. W tym przykładzie używana jest kolejka SYSTEM.DEFAULT.LOCAL.QUEUE , która powinna być zdefiniowana na dowolnym normalnie ustawionym serwerze WebSphere MQ .

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

- Połącz z menedżerem kolejek
- Dostęp do kolejki
- Umieszczanie komunikatu w kolejce
- Pobieranie komunikatu z kolejki

Informacje na temat przykładów startowych MQAX dla produktu Microsoft Basic w wersji 4 lub nowszej zawiera sekcja “Uruchamianie przykładu MQAXTRIV” na stronie 1150

Informacje na temat przykładu, które umożliwiają przeglądanie właściwości i metod menedżerów kolejek i obiektów kolejki, zawiera sekcja “Uruchamianie przykładu MQAXCLSS” na stronie 1152 .

For information on the MQAXDLST sample, “Przykład MQAXDLST” na stronie 1152

Informacje na temat uruchamiania przykładu startowego MQAX dla programu Microsoft Excel 95 lub nowszego, MQAXTRIV.XLS, patrz “Uruchamianie komendy MQAXTRIV.XLS , przykład” na stronie 1152.

Informacje na temat uruchamiania demonstracji Banku za pomocą programu MQAX.XLS, patrz sekcja “Uruchamianie demonstracji w Banku za pomocą programu MQAX.XLS” na stronie 1153 .

Informacje na temat przykładowego programu startowego za pomocą przeglądarki WWW kompatybilnej z ActiveX można znaleźć w sekcji “Przykład startowy za pomocą przeglądarki WWW kompatybilnej z ActiveX” na stronie 1153 .

Przygotowanie do uruchomienia przykładów

Aby uruchomić dowolną z przykładów, należy wykonać jedną z poniższych czynności w zależności od tego, które z przykładów mają być uruchamiane.

- Microsoft Visual Basic, wersja 4 (lub nowsza)
- Microsoft Excel 95 (lub nowszy)
- Przeglądarka WWW

Potrzebne są również:

- Menedżer kolejek produktu WebSphere MQ jest uruchomiony.
- Kolejka produktu WebSphere MQ jest już zdefiniowana.

Obsługa błędów w próbkach

Większość przykładów znajdujących się w pakiecie WebSphere MQ Automation Classes for ActiveX nie może być w niewielkim zakresie lub nie obsługuje błędów. Więcej informacji na temat obsługi błędów zawiera sekcja “Obsługa błędów” na stronie 1067.

Uruchamianie przykładu MQAXTRIV

1. Uruchom menedżer kolejek.
2. W Eksploratorze Windows lub w File Managerwybierz ikonę dla przykładu MQAXTRIV.VBP (plik projektu Visual Basic) i otwórz plik.

   Program Visual Basic uruchamia i otwiera plik MQAXTRIV.VBP.
3. W Visual Basic naciśnij klawisz funkcyjny 5 (F5), aby uruchomić próbkę.
4. Kliknij w dowolnym miejscu w formularzu okna, "MQAX trivial tester".

Jeśli wszystko działa poprawnie, tło okna powinno zmienić się na zielony. Jeśli wystąpił problem z konfiguracją, tło okna powinno zostać zmienione na czerwony, a informacje o błędach zostaną wyświetlone.

Na poniższym rysunku przedstawiono centralną część przykładu Visual Basic.

```
Option Explicit

Private Sub Form_Click()

'*****************************************************************************
'* This simple example illustrates how to put and get a WebSphere MQ message to
'* and from a WebSphere MQ message queue. The data from the message returned by the
'*get is read and compared with that from the original message.
'*****************************************************************************
Dim MQSess As MQSession                 '* session object
Dim QMgr As MQQueueManager              '* queue manager object
Dim Queue As MQQueue                    '* queue object
Dim PutMsg As MQMessage                 '* message object for put
Dim GetMsg As MQMessage                 '* message object for get
Dim PutOptions As MQPutMessageOptions   '* get message option

Dim GetOptions As MQGetMessageOptions   '* put message options
Dim PutMsgStr As String                 '* put message data string
Dim GetMsgStr As String                 '* get message data string
'*****************************************************************************
'* Handle errors
'*****************************************************************************
On Error GoTo HandleError

'*****************************************************************************
'* Initialize the current position for the form
'*****************************************************************************
CurrentX = 0
CurrentY = 0

'*****************************************************************************
'* Create the MQSession object and access the MQQueueManager and (local) MQQueue
'*****************************************************************************
Set MQSess = New MQSession
Set QMgr = MQSess.AccessQueueManager("")
Set Queue = QMgr.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE", _
                      MQOO_OUTPUT Or MQOO_INPUT_AS_Q_DEF)

'*****************************************************************************
'* Create a new MQMessage object for use with put, add some data then create an
'* MQPutMessageOptions object and put the message
'*****************************************************************************
Set PutMsg = MQSess.AccessMessage()
PutMsgStr = "12345678 " & Time
PutMsg.MessageData = PutMsgStr
Set PutOptions = MQSess.AccessPutMessageOptions()
Queue.Put PutMsg, PutOptions

'*****************************************************************************
'* Create a new MQMessage object for use with get, set the MessageId (to that of
'* the message that was put), create an MQGetMessageOptions object and get the
'* message.
'*
'* Note: Setting the MessageId ensures that the get returns the MQMessage
'* that was put earlier.
'*****************************************************************************

Set GetMsg = MQSess.AccessMessage()
GetMsg.MessageId = PutMsg.MessageId
Set GetOptions = MQSess.AccessGetMessageOptions()
Queue.Get GetMsg, GetOptions
'*****************************************************************************
'* Read the data from the message returned by the get, compare it with
'* that from the original message and output a suitable message.
'*****************************************************************************
GetMsgStr = GetMsg.MessageData
Cls
If GetMsgStr = PutMsgStr Then
    BackColor = RGB(127, 255, 127)  '* set to green for ok
    Print
    Print "Message data comparison was successful."
    Print "Message data: """ & GetMsgStr & """"
```

```
Else
    BackColor = RGB(255, 255, 127)  '* set to amber for compare error
    Print "Compare error: "
    Print "The message data returned by the get did not match the " &_
    "input data from the original message that was put."
    Print
    Print "Input message data:        """ & PutMsgStr & """"
    Print "Returned message data: """ & GetMsgStr & """"
End If

Exit Sub
'*****************************************************************************
'* Handle errors
'*****************************************************************************
HandleError:
Dim ErrMsg As String
Dim StrPos As Integer

Cls
BackColor = RGB(255, 0, 0)  '* set to red for error
Print "An error occurred as follows:"
Print ""
If MQSess.CompletionCode <> MQCC_OK Then
    ErrMsg = Err.Description
    StrPos = InStr(ErrMsg, " ")            '* search for first blank
    If StrPos > 0 Then
        Print Left(ErrMsg, StrPos)         '* print offending MQAX object name
    Else
        Print Error(Err)                   '* print complete error object
    End If
    Print ""
    Print "WebSphere MQ Completion Code = " & MQSess.CompletionCode
    Print "WebSphere MQ Reason Code = " & MQSess.ReasonCode
    Print "(" & MQSess.ReasonName & ")"
Else
    Print "Visual Basic error:  " & Err
    Print Error(Err)
End If

Exit Sub

End Sub
```

Uruchamianie przykładu MQAXCLSS

Ten przykład umożliwia przeglądanie właściwości i metod menedżerów kolejek i obiektów kolejki.

1. Uruchom menedżer kolejek.
2. Otwórz plik MQAXCLSS.VBP, klikając dwukrotnie ikonę dokumentu w Eksploratorze Windows lub klikając opcję Plik-Otwórz z menu Plik w języku Basic Visual Basic.
3. Uruchom przykład.
4. Wprowadź odpowiednie nazwy menedżerów kolejek i kolejek, a następnie kliknij odpowiednie przyciski.

Przykład MQAXDLST

Przykład Visual Basic MQAXDLST demonstruje użycie listy dystrybucyjnej w celu wysłania tego samego komunikatu do dwóch kolejek przy użyciu jednej operacji put. Aby uruchomić przykład, należy wykonać to samo, co w przypadku przykładu MQAXCLSS.

Przykład MQAX Starter dla programu Microsoft Excel 95 lub nowszego

W tej sekcji opisano sposób uruchamiania przykładu startowego MQAX dla programu Microsoft Excel 95 lub nowszego, MQAXTRIV.XLS.

Uruchamianie komendy MQAXTRIV.XLS , przykład

1. Uruchom menedżer kolejek.

2. W Eksploratorze lub File Managerwybierz ikonę dla przykładowego programu MQAX MQAXTRIV.XLS.
3. Kliknij przycisk w arkuszu kalkulacyjnym.
4. Ekran zostanie zaktualizowany za pomocą komunikatu o powodzeniu (lub niepowodzeniu).

Uruchamianie demonstracji w Banku za pomocą programu MQAX.XLS

Wykonaj poniższe kroki, aby uruchomić demonstracę Banku.

1. Uruchom menedżer kolejek.
2. Uruchom plik komend MQSC IBM WebSphere MQ , BANK.TST. Spowoduje to ustawienie niezbędnych definicji kolejek produktu IBM WebSphere MQ .

   Informacje na temat korzystania z pliku komend MQSC można znaleźć w sekcji Komendy skryptowe (MQSC).
3. Uruchom komendę MQAXBSRV.VBP. Ten przykładowy program jest serwerem symulującym aplikację zaplecza, który musi być uruchomiony za pomocą programu Microsoft Excel.
4. Uruchom program MQAX.XLS. Ten przykład jest demonstracją klienta IBM WebSphere MQ .
5. Wybierz klienta z listy.
6. Kliknij przycisk **Submit** (Wyślij).

Po krótkiej przerwie (około 3 sekundy) wyświetlane są pola zapełniania wartościami i wykresem słupkowym.

Przykład startowy za pomocą przeglądarki WWW kompatybilnej z ActiveX

Uwaga: Aby uruchomić ten przykład, należy uruchomić kompatybilną przeglądarkę WWW ActiveX . Przeglądarka Microsoft Internet Explorer (ale nie Netscape Navigator) jest kompatybilną przeglądarką WWW.

Uruchamianie przykładu HTML

W tym przykładzie pokazano, w jaki sposób można wywołać funkcję MQAX z poziomu języka VBScript i skryptu JavaScript.

1. Uruchom menedżer kolejek.
2. Otwórz plik "MQAXTRIV.HTM", w przeglądarce WWW zgodnej z ActiveX .

   Można to zrobić, klikając dwukrotnie ikonę pliku w Eksploratorze Windows lub wybierając opcję Plik-Otwórz z menu Plik przeglądarki WWW kompatybilnej z ActiveX .
3. Postępuj zgodnie z instrukcjami wyświetlanym na ekranie.

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
U.S.A.

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, Japan

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 innych podmiotów zostały wprowadzone wyłącznie dla wygody użytkowników 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
Koordynator współdziałania z oprogramowaniem, Dział 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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 innych niż produkty IBM pochodzą od dostawców tych produktów, z opublikowanych przez nich zapowiedzi lub innych powszechnie dostępnych źródeł. Firma IBM nie testowała tych produktów i nie może potwierdzić dokładności pomiarów wydajności, kompatybilności ani żadnych innych danych związanych z tymi produktami. Pytania dotyczące możliwości produktów innych podmiotów 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: