7.5

Sviluppo di Applicazioni per IBM WebSphere MQ

Nota

Prima di utilizzare queste informazioni e il prodotto che supportano, leggere le informazioni in “Informazioni particolari” a pagina 1127.

Questa edizione si applica alla versione 7 release 5 di IBM® WebSphere MQ e a tutte le release e modifiche successive, se non diversamente indicato nelle nuove edizioni.

Quando si inviano informazioni a IBM, si concede a IBM un diritto non esclusivo di utilizzare o distribuire le informazioni in qualsiasi modo ritenga appropriato senza incorrere in alcun obbligo verso l'utente.

Indice

Sviluppo delle applicazioni

IBM WebSphere MQ fornisce diversi modi in cui è possibile sviluppare applicazioni per inviare e ricevere messaggi necessari per supportare i processi aziendali. È anche possibile sviluppare applicazioni per gestire i gestori code e le risorse correlate.

Prima di sviluppare applicazioni per IBM WebSphere MQ, assicurati di avere familiarità con i concetti in IBM WebSphere MQ Panoramica tecnicaIBM WebSphere MQ Panoramica tecnica.

È possibile sviluppare applicazioni per IBM WebSphere MQ in diversi linguaggi di programmazione. Per informazioni sui linguaggi di programmazione supportati e sulle relative funzioni, consultare “Scelta del linguaggio di programmazione da utilizzare” a pagina 79.

Consultare le seguenti sezioni per i tipi di applicazioni che è possibile scrivere per IBM WebSphere MQ su piattaforme differenti.

Tipi di applicazione che è possibile scrivere per IBM WebSphere MQ

Queste informazioni riguardano i tipi di applicazioni che possono essere scritte su IBM WebSphere MQ.

I prodotti IBM WebSphere MQ sono gestori code e programmi di abilitazione applicazioni. Supportano IBM Message Queue Interface (MQI) attraverso cui i programmi possono inserire i messaggi su una coda e richiamare i messaggi da una coda.

Con IBM WebSphere MQ per piattaforme nonz/OS , è possibile scrivere applicazioni che:

- Inviare messaggi ad altre applicazioni in esecuzione negli stessi sistemi operativi. Le applicazioni possono essere sullo stesso o su un altro sistema.
- Inviare messaggi alle applicazioni che vengono eseguite su altre piattaforme IBM WebSphere MQ .
- Utilizzare l'accodamento di messaggi da CICS per TXSeries per AIX, TXSeries per HP-UX, TXSeries per Solaris e TXSeries per applicazioni di sistemi Windows .
- Utilizzare l'accodamento dei messaggi dai sistemi Encina per AIX, HP-UX, Solaris e Windows .
- Utilizzare l'accodamento messaggi da Tuxedo per sistemi AIX, AT & T, HP-UX, Solaris e Windows .
- Utilizzare IBM WebSphere MQ come gestore transazioni, coordinando gli aggiornamenti effettuati dai gestori risorse esterni nelle unità di lavoro IBM WebSphere MQ . I seguenti gestori risorse esterni sono supportati e compatibili con l'interfaccia XA X/OPEN
  - DB2
  - Informix
  - Oracle
  - Sybase
- Elaborare diversi messaggi insieme come una singola unità di lavoro di cui è possibile eseguire il commit o il backout.
- Eseguire da un ambiente IBM WebSphere MQ completo o da un ambiente client IBM WebSphere MQ MQI sulle piattaforme riportate di seguito:
  - UNIX and Linux® sistemi
  - Finestre

Concetti correlati

Sicurezza

Concetti dello sviluppo di applicazioni

È possibile utilizzare una scelta di linguaggi procedurali o orientati agli oggetti per scrivere applicazioni IBM WebSphere MQ . Utilizzare i link in questo argomento per informazioni sui concetti di IBM WebSphere MQ che sono utili per sviluppatori di applicazioni.

Prima di iniziare a progettare e scrivere le applicazioni IBM WebSphere MQ , familiarizzare con i concetti di base di IBM WebSphere MQ , consultare gli argomenti in Panoramica tecnica. Per informazioni sui tipi di applicazione che è possibile scrivere per IBM WebSphere MQ, consultare “Sviluppo delle applicazioni” a pagina 7.

Utilizzare i seguenti link per informazioni sui concetti IBM WebSphere MQ specifici per lo sviluppo dell'applicazione:

- “IBM WebSphere MQ messaggi” a pagina 9
- Messaggistica point-to-point
- Introduzione alla messaggistica di pubblicazione / sottoscrizione WebSphere MQ
- “Utilizzo dell'interfaccia della coda messaggi (MQI) in un'applicazione client” a pagina 355
- “Utilizzo dei servizi Web in WebSphere MQ” a pagina 948
- “Programmi di uscita canale per canali di messaggistica” a pagina 399
- “Scenari di supporto transazionale” a pagina 42

Prima di poter eseguire applicazioni che utilizzano MQI, è necessario creare determinati oggetti IBM WebSphere MQ . Per ulteriori informazioni, vedere “Programmi applicativi che utilizzano MQI” a pagina 9.

Concetti correlati

“Progettazione di applicazioni IBM WebSphere MQ” a pagina 90
Una volta stabilito il modo in cui le applicazioni possono sfruttare le piattaforme e gli ambienti disponibili, è necessario decidere come utilizzare le funzioni offerte da WebSphere MQ.

“Programmi di WebSphere MQ di esempio” a pagina 97
Utilizzare questa raccolta di argomenti per informazioni sui programmi WebSphere MQ di esempio su piattaforme differenti.

“Scrittura di un'applicazione di accodamento” a pagina 195
Utilizzare queste informazioni per informazioni sulla scrittura delle applicazioni di accodamento, la connessione e la disconnessione da un gestore code, la pubblicazione / sottoscrizione e l'apertura e la chiusura di oggetti.

“Scrittura delle applicazioni client” a pagina 354
Informazioni necessarie per scrivere le applicazioni client su WebSphere MQ.

“Scelta del linguaggio di programmazione da utilizzare” a pagina 79
Utilizza queste informazioni per scoprire i linguaggi di programmazione e i framework supportati da IBM WebSphere MQe alcune considerazioni per utilizzarli.

“Utilizzo delle classi WebSphere MQ per JMS” a pagina 717
WebSphere MQ classes for Java Message Service (WebSphere MQ classes for JMS) è il fornitore JMS fornito con WebSphere MQ. Oltre ad implementare le interfacce definite nel pacchetto javax.jms , WebSphere MQ classes per JMS fornisce due serie di estensioni all'API JMS.

“Utilizzo di Component Object Model Interface (WebSphere MQ Automation Classes for ActiveX)” a pagina 1035
Le WebSphere MQ Automation Classes for ActiveX (MQAX) sono componenti ActiveX che forniscono classi che è possibile utilizzare nella propria applicazione per accedere a WebSphere MQ.

“Utilizzo di classi WebSphere MQ per Java” a pagina 654
Le classi WebSphere MQ per Java consentono di utilizzare WebSphere MQ in un ambiente Java. Un'applicazione Java può utilizzare le classi WebSphere MQ per Java o WebSphere MQ per JMS per accedere alle risorse WebSphere MQ .

“Usando .Net” a pagina 562
WebSphere Le classi di MQ per .NET consentono a un programma scritto nel framework di programmazione .NET di connettersi a WebSphere MQ come client WebSphere MQ MQI o di connettersi direttamente a un WebSphere MQ .

“Usando C++” a pagina 630
WebSphere MQ fornisce classi C++ equivalenti agli oggetti di WebSphere MQ e alcune classi aggiuntive equivalenti ai tipi di dati array. Fornisce un certo numero di funzioni non disponibili tramite MQI.

“Creazione di un'applicazione IBM WebSphere MQ” a pagina 431
Utilizzare queste informazioni per informazioni sulla generazione di un'applicazione IBM WebSphere MQ su piattaforme differenti.

Programmi applicativi che utilizzano MQI

I programmi applicativi IBM WebSphere MQ necessitano di determinati oggetti prima che possano essere eseguiti correttamente.

Figura 1 a pagina 9 mostra un'applicazione che rimuove i messaggi da una coda, li elabora e invia alcuni risultati a un'altra coda sullo stesso gestore code.

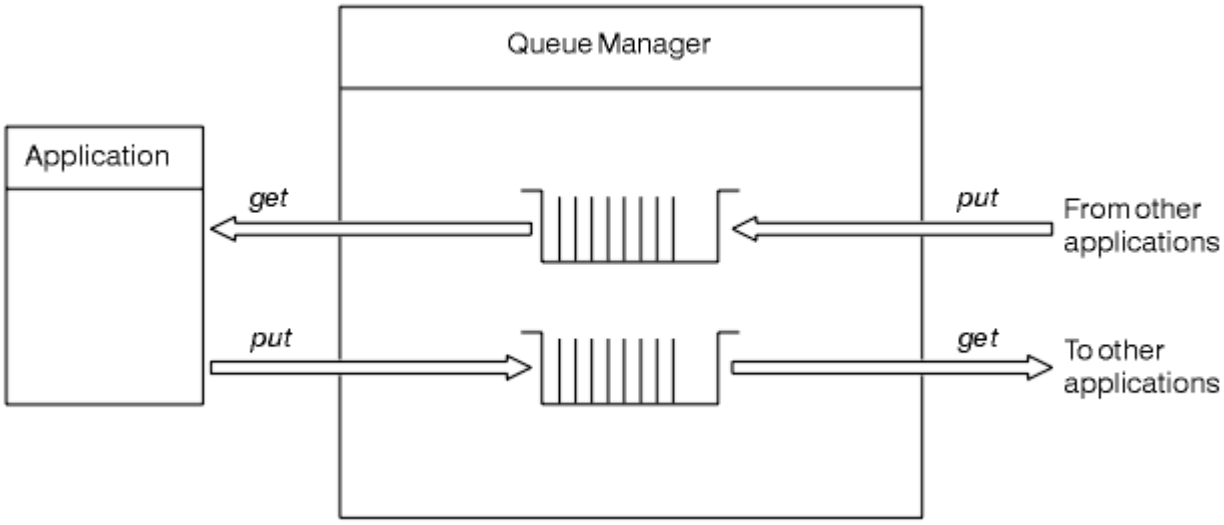


Figura 1. Code, messaggi e applicazioni

Mentre le applicazioni possono inserire i messaggi nelle code locali o remote (utilizzando MQPUT), possono solo ricevere i messaggi direttamente dalle code locali (utilizzando MQGET).

Prima di poter eseguire questa applicazione, è necessario soddisfare le seguenti condizioni:

- Il gestore code deve esistere ed essere in esecuzione.
- È necessario definire la prima coda dell'applicazione da cui devono essere rimossi i messaggi.
- È necessario definire anche la seconda coda, in cui l'applicazione inserisce i messaggi.
- L'applicazione deve essere in grado di connettersi al gestore code. Per fare ciò deve essere collegato a IBM WebSphere MQ. Consultare “Creazione di un'applicazione IBM WebSphere MQ” a pagina 431.
- Anche le applicazioni che inserano i messaggi nella prima coda devono connettersi a un gestore code. Se sono remoti, devono anche essere impostati con code di trasmissione e canali. Questa parte del sistema non viene visualizzata in Figura 1 a pagina 9.

IBM WebSphere MQ messaggi

Queste informazioni introducono il concetto del messaggio IBM WebSphere MQ , le parti del messaggio e il descrittore del messaggio.

I messaggi IBM WebSphere MQ sono composti da due parti:

- Proprietà dei messaggi
- Dati applicazione

Figura 2 a pagina 10 rappresenta un messaggio e mostra come è diviso logicamente in proprietà del messaggio e dati dell'applicazione.

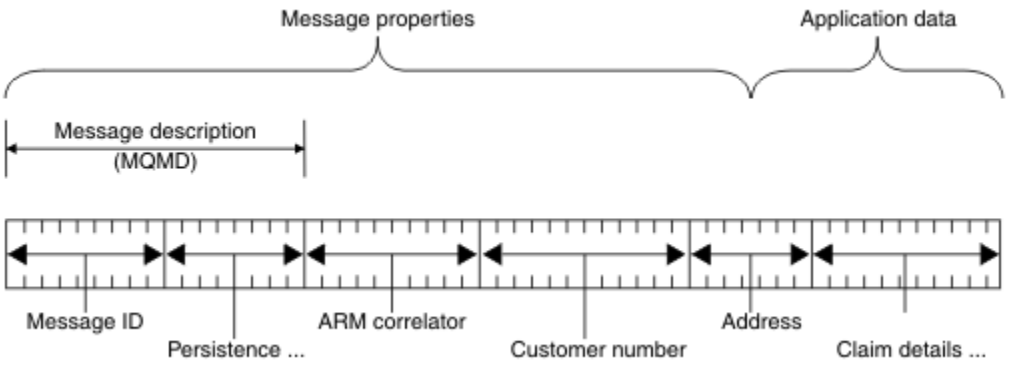


Figura 2. Rappresentazione di un messaggio

I dati dell'applicazione trasmessi in un messaggio di WebSphere MQ non vengono modificati da parte di un gestore code a meno che non venga eseguita la conversione dei dati su di essi. Inoltre, WebSphere MQ non pone alcuna limitazione sul contenuto di questi dati. La lunghezza dei dati in ogni messaggio non può superare il valore dell'attributo *MaxMsgLength* sia della coda che del gestore code.

Su WebSphere MQ per AIX, WebSphere MQ per HP-UX, WebSphere MQ per Linux, WebSphere MQ per Solaris, e WebSphere MQ per Windows, il valore predefinito di *MaxMsgLength* è 100 MB (104 857 600 byte).

In alcune circostanze, rendere i messaggi leggermente più brevi del valore dell'attributo *MaxMsgLength* . Per ulteriori informazioni, fare riferimento a “I dati nel messaggio” a pagina 230.

Si crea un messaggio quando si utilizzano le chiamate MQPUT o MQPUT1 MQI. Come input per queste chiamate, si forniscono le informazioni di controllo (come la priorità del messaggio e il nome di una coda di risposta) e i propri dati, quindi la chiamata inserisce il messaggio in una coda. Per ulteriori informazioni su queste chiamate, fare riferimento a MQPUT e MQPUT1 .

Descrittore messaggio

È possibile accedere alle informazioni di controllo del messaggio utilizzando la struttura MQMD, che definisce il *descrittore del messaggio*.

Per una descrizione completa della struttura MQMD, consultare MQMD - Descrittore messaggio.

Consultare “Contesto messaggio” a pagina 39 per una descrizione su come utilizzare i campi all'interno di MQMD che contengono informazioni sull'origine del messaggio.

Esistono diverse versioni del descrittore del messaggio. Ulteriori informazioni per raggruppare e segmentare i messaggi (consultare “Gruppi di messaggi” a pagina 36) sono fornite nella Versione 2 del descrittore del messaggio (o MQMDE). È lo stesso descrittore del messaggio della Versione 1 ma dispone di campi aggiuntivi. Questi sono descritti in MQMDE - Message descriptor extension.

Tipi di messaggio

Esistono quattro tipi di messaggi definiti da IBM WebSphere MQ.

Questi quattro messaggi sono:

- pacchetto dati
- Messaggi di richiesta
- Messaggi di risposta
- Messaggi di report

- Tipi di messaggio di prospetto
- Opzioni messaggio report

Le applicazioni possono utilizzare i primi tre tipi di messaggi per passare le informazioni tra loro. Il quarto tipo, report, viene utilizzato dalle applicazioni e dai gestori code per riportare informazioni sugli eventi, ad esempio la ricorrenza di un errore.

Ogni tipo di messaggio è identificato da un valore MQMT_ *. È inoltre possibile definire i propri tipi di messaggio. Per l'intervallo di valori che è possibile utilizzare, consultare MsgType.

Datagrammi

Utilizzare un *datagramma* quando non è necessaria una risposta dall'applicazione che riceve il messaggio (ossia, richiama il messaggio dalla coda).

Un esempio di un'applicazione che potrebbe utilizzare i datagrammi è quella che visualizza le informazioni di volo in una lounge dell'aeroporto. Un messaggio potrebbe contenere i dati per un'intera schermata di informazioni sul volo. È improbabile che un'applicazione di questo tipo richieda un riconoscimento per un messaggio perché probabilmente non importa se un messaggio non viene consegnato. L'applicazione invia un messaggio di aggiornamento dopo un breve periodo di tempo.

Messaggi di richiesta

Utilizzare il *messaggio di richiesta* quando si desidera una risposta dall'applicazione che riceve il messaggio.

Un esempio di applicazione che potrebbe utilizzare i messaggi di richiesta è quello che visualizza il saldo di un conto corrente. Il messaggio di richiesta potrebbe contenere il numero del conto e il messaggio di risposta potrebbe contenere il saldo del conto.

Se si desidera collegare il messaggio di risposta al messaggio di richiesta, esistono due opzioni:

- Rendere responsabile l'applicazione che gestisce il messaggio di richiesta per assicurarsi che immetta le informazioni nel messaggio di replica correlato al messaggio di richiesta.
- Utilizzare il campo report nel descrittore del messaggio del messaggio di richiesta per specificare il contenuto dei campi *MsgId* e *CorrelId* del messaggio di risposta:
  - È possibile richiedere che *MsgId* o *CorrelId* del messaggio originale vengano copiati nel campo *CorrelId* del messaggio di risposta (l'azione predefinita è copiare *MsgId*).
  - È possibile richiedere che venga creato un nuovo *MsgId* per il messaggio di replica o che il *MsgId* del messaggio originale venga copiato nel campo *MsgId* del messaggio di replica (l'azione predefinita è la generazione di un nuovo identificativo del messaggio).

Messaggi di risposta

Utilizzare un *messaggio di risposta* quando si risponde ad un altro messaggio.

Quando si crea un messaggio di risposta, rispettare le opzioni impostate nel descrittore del messaggio a cui si risponde. Le opzioni del report specificano il contenuto dei campi identificativo del messaggio (*MsgId*) e identificativo di correlazione (*CorrelId*). Questi campi consentono all'applicazione che riceve la risposta di correlarla alla sua richiesta originale.

Messaggi di report

I *messaggi di report* informano le applicazioni su eventi quali la ricorrenza di un errore durante l'elaborazione di un messaggio.

Possono essere generati da:

- Un gestore code,
- Un agente del canale dei messaggi (ad esempio, se non è in grado di consegnare il messaggio) oppure

- Un'applicazione (ad esempio, se non può utilizzare i dati nel messaggio).

I messaggi di report possono essere generati in qualsiasi momento e potrebbero arrivare su una coda quando l'applicazione non li prevede.

Tipi di messaggi di report

Quando si inserisce un messaggio su una coda, è possibile scegliere di ricevere:

- Un *messaggio di report di eccezioni*. Questo viene inviato in risposta a un messaggio con l'indicatore delle eccezioni impostato. Viene generato dall'MCA (message channel agent) o dall'applicazione.
- Un *messaggio di report scadenza*. Ciò indica che un'applicazione ha tentato di richiamare un messaggio che aveva raggiunto la soglia di scadenza; il messaggio è contrassegnato per essere eliminato. Questo tipo di report viene generato dal gestore code.
- Un *messaggio di report di conferma di arrivo (COA)*. Ciò indica che il messaggio ha raggiunto la coda di destinazione. Viene generato dal gestore code.
- Un *messaggio di report COD (Conferma di consegna)*. Ciò indica che il messaggio è stato richiamato da un'applicazione ricevente. Viene generato dal gestore code.
- Un *messaggio di report PAN (positive action notification)*. Ciò indica che una richiesta è stata eseguita correttamente (ovvero, l'azione richiesta nel messaggio è stata eseguita correttamente). Questo tipo di report viene generato dall'applicazione.
- Un *messaggio di report NAN (negative action notification)*. Ciò indica che una richiesta non è stata eseguita correttamente (ovvero, l'azione richiesta nel messaggio non è stata eseguita correttamente). Questo tipo di report viene generato dall'applicazione.

Nota: Ogni tipo di messaggio di report contiene uno dei seguenti:

- L'intero messaggio originale
- I primi 100 byte di dati nel messaggio originale
- Nessun dato dal messaggio originale

È possibile richiedere più di un tipo di messaggio di report quando si inserisce un messaggio in una coda. Se si seleziona il messaggio di conferma della consegna e le opzioni del messaggio di report di eccezioni, se il messaggio non viene consegnato, si riceve un messaggio di report di eccezioni. Tuttavia, se si seleziona solo l'opzione del messaggio di report di conferma della consegna e il messaggio non viene consegnato, non si riceve un messaggio di report di eccezioni.

I messaggi di report richiesti, quando vengono soddisfatti i criteri per la generazione di un determinato messaggio, sono gli unici che si ricevono.

Opzioni del messaggio di report

È possibile *eliminare* un messaggio dopo che si è verificata un'eccezione. Se si seleziona l'opzione di eliminazione e si è richiesto un messaggio di report di eccezioni, il messaggio di report viene inviato a *ReplyToQ* e *ReplyToQMgr*e il messaggio originale viene eliminato.

Nota: Un vantaggio di questo è che è possibile ridurre il numero di messaggi che vanno alla coda di messaggi non recapitabili. Tuttavia, significa che l'applicazione, a meno che non invii solo messaggi datagramma, deve gestire i messaggi restituiti. Quando viene generato un messaggio di report di eccezioni, eredita la persistenza del messaggio originale.

Se un messaggio di report non può essere consegnato (se la coda è piena, ad esempio), il messaggio di report viene posizionato nella coda di messaggi non recapitabili.

Se si desidera ricevere un messaggio di report, specificare il nome della coda di risposta nel campo *ReplyToQ* ; altrimenti MQPUT o MQPUT1 del messaggio originale non riesce con MQRC_MISSING_REPLY_TO_Q.

È possibile utilizzare altre opzioni di report nel descrittore del messaggio (MQMD) di un messaggio per specificare il contenuto dei campi *MsgId* e *CorrelId* di qualsiasi messaggio di report creato per il messaggio:

- È possibile richiedere che il *MsgId* o il *CorrelId* del messaggio originale vengano copiati nel campo *CorrelId* del messaggio di report. L'azione predefinita consiste nel copiare l'identificativo del messaggio. Utilizzare MQRO_COPY_MSG_ID_TO_CORRELID poiché abilita il mittente di un messaggio a correlare il messaggio di risposta o di report con il messaggio originale. L'identificativo di correlazione del messaggio di risposta o di report è identico all'identificativo del messaggio originale.
- È possibile richiedere che venga generato un nuovo *MsgId* per il messaggio del report o che il *MsgId* del messaggio originale venga copiato nel campo *MsgId* del messaggio del report. L'azione predefinita è la generazione di un nuovo identificativo di messaggio. Utilizzare MQRO_NEW_MSG_ID in quanto garantisce che ogni messaggio nel sistema abbia un identificativo di messaggio diverso e possa essere distinto in modo non ambiguo da tutti gli altri messaggi nel sistema.
- Le applicazioni specializzate potrebbero dover utilizzare MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID. Tuttavia, è necessario progettare l'applicazione che legge i messaggi dalla coda per essere certi che funzioni correttamente quando, ad esempio, la coda contiene più messaggi con lo stesso identificativo.

  Le applicazioni server devono controllare le impostazioni di questi indicatori nel messaggio di richiesta e impostare in modo appropriato i campi *MsgId* e *CorrelId* nel messaggio di risposta o di report.

  Le applicazioni che agiscono come intermediari tra un'applicazione richiedente e un'applicazione server non devono controllare le impostazioni di questi indicatori. Ciò è dovuto al fatto che queste applicazioni in genere devono inoltrare il messaggio all'applicazione server senza modificare i campi *MsgId*, *CorrelId*e *Report* . Ciò consente all'applicazione server di copiare *MsgId* dal messaggio originale nel campo *CorrelId* del messaggio di replica.

Quando si genera un report su un messaggio, le applicazioni server devono verificare se una di queste opzioni è stata impostata.

Per ulteriori informazioni su come utilizzare i messaggi di report, consultare Report.

Per indicare la natura del report, i gestori code utilizzano una serie di codici di feedback. Inseriscono questi codici nel campo *Feedback* del descrittore di messaggi di un messaggio di report. I gestori code possono anche restituire i codici motivo MQI nel campo *Feedback* . IBM WebSphere MQ definisce un intervallo di codici di feedback per le applicazioni da utilizzare.

Per ulteriori informazioni sul feedback e sui codici di errore, consultare Feedback.

Un esempio di programma che potrebbe utilizzare un codice di feedback è quello che controlla i carichi di lavoro di altri programmi che servono una coda. Se c'è più di un'istanza di un programma che serve una coda, e il numero di messaggi che arrivano sulla coda non lo giustifica più, un tale programma può inviare un messaggio di report (con il codice di feedback MQFB_QUIT) a uno dei programmi di servizio per indicare che il programma deve terminare la sua attività. Un programma di controllo potrebbe utilizzare la chiamata MQINQ per scoprire quanti programmi servono una coda.

Report e messaggi segmentati

Non supportato su WebSphere MQ per z/OS .

Se un messaggio è segmentato (consultare “Segmentazione del messaggio” a pagina 264 per una descrizione dei messaggi segmentati) e si richiede la generazione di report, è possibile che si ricevano più report di quelli che si sarebbero ricevuti se il messaggio non fosse stato segmentato.

Per i report generati da WebSphere MQ

Se si segmentano i messaggi o si consente al gestore code di farlo, esiste un solo caso in cui è possibile prevedere di ricevere un singolo report per l'intero messaggio. Ciò si verifica quando si richiedono solo report COD e si è specificato MQGMO_COMPLETE_MSG sull'applicazione di richiamo.

In altri casi, l'applicazione deve essere preparata per gestire diversi report, di solito uno per ciascun segmento.

Nota: Se si segmentano i messaggi e sono necessari solo i primi 100 byte dei dati del messaggio originale da restituire, modificare l'impostazione delle opzioni del report per richiedere i report senza dati per i segmenti che hanno un offset di 100 o più. Se non si esegue questa operazione e si lascia l'impostazione in modo che ogni segmento richieda 100 byte di dati e si richiamano i messaggi di report con un singolo MQGET che specifica MQGMO_COMPLETE_MSG, i report si assemblano in un messaggio di grandi dimensioni contenente 100 byte di dati di lettura ad ogni offset appropriato. Se ciò si verifica, è necessario un buffer di grandi dimensioni oppure è necessario specificare MQGMO_ACCEPT_TRUNCATED_MSG.

Per i report generati dalle applicazioni

Se l'applicazione genera report, copiare sempre le intestazioni WebSphere MQ presenti all'inizio dei dati del messaggio originali nei dati del messaggio del report.

Quindi, aggiungere nessuno, 100 byte o tutti i dati del messaggio originale (o qualsiasi altra quantità che si includa di solito) ai dati del messaggio di report.

È possibile riconoscere le intestazioni WebSphere MQ che devono essere copiate esaminando i nomi formato successivi, iniziando con MQMD e proseguendo attraverso le intestazioni presenti. I seguenti nomi `Format` indicano queste intestazioni WebSphere MQ :

- MQMDE
- MQDLH
- QQMQX
- MQIIH
- MQH*

MQH* indica qualsiasi nome che inizia con i caratteri MQH.

Il nome `Format` si verifica in posizioni specifiche per MQDLH e MQXQH, ma per le altre intestazioni WebSphere MQ si verifica nella stessa posizione. La lunghezza dell'intestazione è contenuta in un campo che si verifica anche nella stessa posizione per MQMDE, MQIMS e tutte le intestazioni MQH*.

Se si utilizza un MQMD Versione 1 e si crea un report su un segmento, un messaggio in un gruppo o un messaggio per cui è consentita la segmentazione, i dati del report devono iniziare con un MQMDE. Impostare il campo *`OriginalLength`* sulla lunghezza dei dati del messaggio originale, escludendo la lunghezza di qualsiasi intestazione WebSphere MQ che si trova.

Recupero di report

Se si richiedono i report COA o COD, è possibile richiedere che vengano riassemblati con MQGMO_COMPLETE_MSG.

MQGET con MQGMO_COMPLETE_MSG viene soddisfatto quando nella coda sono presenti messaggi di report sufficienti (di un singolo tipo, ad esempio COA, e con lo stesso *`GroupId`*) per rappresentare un messaggio originale completo. Ciò è vero anche se i messaggi di report stessi non contengono i dati originali completi; il campo *`OriginalLength`* in ogni messaggio di report fornisce la lunghezza dei dati originali rappresentati da quel messaggio di report, anche se i dati stessi non sono presenti.

È possibile utilizzare questa tecnica anche se sulla coda sono presenti diversi tipi di report (ad esempio, COA e COD), poiché un MQGET con MQGMO_COMPLETE_MSG riassembla i messaggi di report solo se hanno lo stesso codice *Feedback* . Tuttavia, di solito non è possibile utilizzare questa tecnica per i report di eccezioni, poiché, in genere, questi hanno codici *Feedback* differenti.

È possibile utilizzare questa tecnica per ottenere un'indicazione positiva dell'arrivo dell'intero messaggio. Tuttavia, nella maggior parte dei casi è necessario considerare la possibilità che alcuni segmenti arrivino mentre altri potrebbero generare un'eccezione (o la scadenza, se consentito). Non è possibile utilizzare MQGMO_COMPLETE_MSG in questo caso, poiché, in generale, è possibile ottenere codici *Feedback* diversi per segmenti diversi e più di un report per un segmento. Tuttavia, è possibile utilizzare MQGMO_ALL_SEGMENTS_AVAILABLE.

Per consentire ciò, potrebbe essere necessario richiamare i report man mano che arrivano e creare un'immagine nell'applicazione di ciò che è accaduto al messaggio originale. È possibile utilizzare il campo *GroupId* nel messaggio di report per correlare i report con il*GroupId* del messaggio originale e il campo *Feedback* per identificare il tipo di ogni messaggio di report. Il modo in cui si esegue questa operazione dipende dai requisiti dell'applicazione.

Un approccio è il seguente:

- Richiedere report COD e report di eccezioni.
- Dopo un periodo di tempo specifico, verificare se è stata ricevuta una serie completa di report COD utilizzando MQGMO_COMPLETE_MSG. In tal caso, l'applicazione sa che l'intero messaggio è stato elaborato.
- In caso contrario, e i report di eccezione relativi a questo messaggio sono presenti, gestire il problema come per i messaggi non segmentati, ma assicurarsi di ripulire i segmenti orfani ad un certo punto.
- Se ci sono segmenti per cui non ci sono report di alcun tipo, i segmenti originali (o i report) potrebbero essere in attesa che un canale venga riconnesso o la rete potrebbe essere sovraccaricata a un certo punto. Se non è stato ricevuto alcun report di eccezione (o se si pensa che quelli di cui si dispone potrebbero essere solo temporanei), è possibile decidere di lasciare che la propria applicazione attenda un po' più a lungo.

  Come in precedenza, ciò è simile alle considerazioni che si hanno quando si gestiscono messaggi non segmentati, ad eccezione del fatto che è necessario considerare anche la possibilità di ripulire i segmenti orfani.

Se il messaggio originale non è critico (ad esempio, se si tratta di una query o di un messaggio che può essere ripetuto successivamente), impostare una scadenza per garantire che i segmenti orfani vengano rimossi.

Gestori code di livello precedente

Quando un report viene generato da un gestore code che supporta la segmentazione, ma viene ricevuto su un gestore code che *non* supporta la segmentazione, la struttura MQMDE (che identifica *Offset* e *OriginalLength* rappresentata dal report) è sempre inclusa nei dati del report, oltre a zero, 100 byte o tutti i dati originali nel messaggio.

Tuttavia, se un segmento di un messaggio passa attraverso un gestore code che non supporta la segmentazione, se viene generato un report, la struttura MQMDE nel messaggio originale viene trattata esclusivamente come dati. Non viene pertanto incluso nei dati del prospetto se sono stati richiesti zero byte dei dati originali. Senza MQMDE, il messaggio di report potrebbe non essere utile.

Richiedere almeno 100 byte di dati nei report se esiste la possibilità che il messaggio possa viaggiare attraverso un gestore code di livello precedente.

Formato delle informazioni di controllo del messaggio e dei dati del messaggio

Il gestore code è interessato solo al formato delle informazioni di controllo all'interno di un messaggio, mentre le applicazioni che gestiscono il messaggio sono interessate al formato delle informazioni di controllo e dei dati.

Formato delle informazioni di controllo del messaggio

Le informazioni di controllo nei campi stringa di caratteri del descrittore del messaggio devono trovarsi nella serie di caratteri utilizzata dal gestore code.

L'attributo *CodedCharSetId* dell'oggetto gestore code definisce questa serie di caratteri. Le informazioni di controllo devono essere contenute in questa serie di caratteri perché, quando le applicazioni passano i messaggi da un gestore code a un altro, gli agent del canale dei messaggi che trasmettono i messaggi utilizzano il valore di questo attributo per determinare quale conversione dati eseguire.

Formato dei dati del messaggio

È possibile specificare uno dei seguenti elementi:

- Il formato dei dati dell'applicazione
- La serie di caratteri dei dati carattere
- Il formato dei dati numerici

Per eseguire questa operazione, utilizzare questi campi:

Format
Indica al destinatario di un messaggio il formato dei dati dell'applicazione nel messaggio.

Quando il gestore code crea un messaggio, in alcune circostanze utilizza il campo *Format* per identificare il formato di tale messaggio. Ad esempio, quando un gestore code non è in grado di consegnare un messaggio, lo inserisce in una coda di messaggi non recapitabili (messaggi non recapitati). Aggiunge un'intestazione (che contiene ulteriori informazioni di controllo) al messaggio e modifica il campo *Format* per visualizzarlo.

Il gestore code ha un certo numero di *formati integrati* con nomi che iniziano con MQ, ad esempio MQFMT_STRING. Se questi non soddisfano le proprie esigenze, è possibile definire i propri formati (*formati definiti dall'utente*), ma non è necessario utilizzare i nomi che iniziano con MQ per tali formati.

Quando si creano e si utilizzano i propri formati, è necessario scrivere un'uscita di conversione dati per supportare un programma che riceve il messaggio utilizzando MQGMO_CONVERT.

CodedCharSetId
Definisce la serie di caratteri dei dati nel messaggio. Se si desidera impostare questa serie di caratteri su quella del gestore code, è possibile impostare questo campo sulla costante MQCCSI_Q_MGR o MQCCSI_INHERIT.

Quando si ottiene un messaggio da una coda, confrontare il valore del campo *CodedCharSetId* con il valore previsto dall'applicazione. Se i due valori differiscono, potrebbe essere necessario convertire i dati carattere nel messaggio o utilizzare un'uscita del messaggio di conversione dati, se disponibile.

Encoding
Descrive il formato dei dati dei messaggi numerici che contengono numeri interi binari, interi decimali compressi e numeri a virgola mobile. Viene generalmente codificato in base alla particolare macchina su cui è in esecuzione il gestore code.

Quando si inserisce un messaggio su una coda, generalmente si specifica la costante MQENC_NATIVE nel campo *Encoding* . Ciò significa che la codifica dei dati del messaggio è la stessa della macchina su cui è in esecuzione l'applicazione.

Quando si riceve un messaggio da una coda, confrontare il valore del campo *Encoding* nel descrittore del messaggio con il valore della costante MQENC_NATIVE sulla macchina. Se i due valori differiscono, potrebbe essere necessario convertire qualsiasi dato numerico nel messaggio o utilizzare un'uscita del messaggio di conversione dati, se disponibile.

Conversione dati applicazione

È possibile che i dati dell'applicazione debbano essere convertiti nella serie di caratteri e nella codifica richiesta da un'altra applicazione quando si tratta di piattaforme differenti.

Può essere convertito nel gestore code di invio o nel gestore code di ricezione. Se la libreria di formati integrati non soddisfa le proprie esigenze, è possibile definirne di propri. Il tipo di conversione dipende dal formato del messaggio specificato nel campo formato del descrittore del messaggio, MQMD.

Nota: I messaggi con MQFMT_NONE specificato non vengono convertiti.

Conversione sul gestore code di invio

Impostare l'attributo del canale CONVERT su YES se è necessario l'MCA (message channel agent) di invio per convertire i dati dell'applicazione.

La conversione viene eseguita sul gestore code di invio per alcuni formati integrati e per i formati definiti dall'utente se viene fornita un'uscita utente adatta.

Formati integrati

Queste includono:

- Messaggi che sono tutti caratteri (utilizzando il nome formato MQFMT_STRING)
- WebSphere MQ , ad esempio Programmable Command Formats

  WebSphere MQ utilizza i messaggi Programmable Command Format per i messaggi di gestione e gli eventi (in tal caso, il nome del formato utilizzato è MQFMT_ADMIN). È possibile utilizzare lo stesso formato (utilizzando il nome formato MQFMT_PCF) per i propri messaggi e sfruttare la conversione dei dati integrata.

I formati integrati del gestore code hanno tutti nomi che iniziano con MQFMT. Sono elencati e descritti in Formato.

Formati definiti dall'applicazione

Per i formati definiti dall'utente, la conversione dei dati dell'applicazione deve essere eseguita da un programma di uscita di conversione dati (per ulteriori informazioni, consultare “Scrittura delle uscite di conversione dati” a pagina 417). In un ambiente client-server, l'uscita viene caricata sul server e la conversione avviene lì.

Conversione nel gestore code di ricezione

I dati del messaggio dell'applicazione possono essere convertiti dal gestore code di ricezione per i formati integrati e definiti dall'utente.

La conversione viene eseguita durante l'elaborazione di una chiamata MQGET se si specifica l'opzione MQGMO_CONVERT. Per i dettagli, consultare Opzioni

Serie di caratteri codificati

WebSphereI prodotti MQ supportano le serie di caratteri codificati fornite dal sistema operativo sottostante.

Quando si crea un gestore code, il CCSID (coded character set ID) del gestore code utilizzato si basa su quello dell'ambiente sottostante. Se questa è una codepage mista, WebSphere MQ utilizza la parte SBCS della codepage mista come CCSID del gestore code.

Per la conversione dei dati generali, se il sistema operativo sottostante supporta le codepage DBCS, è possibile utilizzare WebSphere MQ .

Consultare la documentazione relativa al proprio sistema operativo per dettagli sulle serie di caratteri codificati supportate.

È necessario considerare la conversione dei dati dell'applicazione, i nomi di formato e le uscite utente quando si scrivono applicazioni che si estendono su più piattaforme. Consultare “Scrittura delle uscite di conversione dati” a pagina 417 per informazioni sul richiamo e la scrittura delle uscite di conversione dati.

Priorità dei messaggi

Impostare la priorità di un messaggio (nel campo *Priority* della struttura MQMD) quando si inserisce il messaggio su una coda. È possibile impostare un valore numerico per la priorità oppure lasciare che il messaggio assuma la priorità predefinita della coda.

L'attributo *MsgDeliverySequence* della coda determina se i messaggi sulla coda vengono memorizzati nella sequenza FIFO (first in, first out) o in FIFO all'interno della sequenza di priorità. Se questo attributo è impostato su MQMDS_PRIORITY, i messaggi vengono accodati con la priorità specificata nel campo *Priority* dei relativi descrittori di messaggi; ma se è impostato su MQMDS_FIFO, i messaggi vengono accodati con la priorità predefinita della coda. I messaggi di uguale priorità vengono memorizzati nella coda in ordine di arrivo.

L'attributo *DefPriority* di una coda imposta il valore di priorità predefinito per i messaggi inseriti in tale coda. Questo valore viene impostato quando la coda viene creata, ma può essere modificato successivamente. Le code alias e le definizioni locali delle code remote, possono avere priorità predefinite diverse dalle code di base in cui vengono risolte. Se nel percorso di risoluzione è presente più di una definizione di coda (consultare “Risoluzione nomi” a pagina 217), la priorità predefinita viene presa dal valore (al momento dell'operazione di inserimento) dell'attributo di *DefPriority* della coda specificata nel comando di apertura.

Il valore dell'attributi *MaxPriority* del gestore code è la massima priorità che è possibile assegnare a un messaggio elaborato da tale gestore code. Non è possibile modificare il valore di questo attributo. In WebSphere MQ, l'attributo ha il valore 9; è possibile creare messaggi con priorità comprese tra 0 (il più basso) e 9 (il più alto).

Proprietà dei messaggi

Utilizzare le proprietà del messaggio per consentire a un'applicazione di selezionare i messaggi da elaborare o per richiamare le informazioni su un messaggio senza accedere alle intestazioni MQMD o MQRFH2 . Inoltre, facilitano la comunicazione tra le applicazioni WebSphere MQ e JMS.

Una proprietà del messaggio è un dato associato a un messaggio, costituito da un nome testuale e un valore di un tipo particolare. Le proprietà dei messaggi vengono utilizzate dai selettori dei messaggi per filtrare le pubblicazioni sugli argomenti o per richiamare in modo selettivo i messaggi dalle code. Le proprietà del messaggio possono essere utilizzate per includere i dati di business o le informazioni di stato senza doverle memorizzare nei dati dell'applicazione. Le applicazioni non devono accedere ai dati nelle intestazioni MQ Message Descriptor (MQMD) o MQRFH2 perché è possibile accedere ai campi in queste strutture di dati come proprietà del messaggio utilizzando le chiamate alla funzione MQI (Message Queue Interface).

L'utilizzo delle proprietà del messaggio in WebSphere MQ imita l'utilizzo delle proprietà in JMS. Ciò significa che è possibile impostare le proprietà in un'applicazione JMS e richiamarle in un'applicazione WebSphere MQ procedurale o viceversa. Per rendere una proprietà disponibile per un'applicazione JMS, assegnarle il prefisso "usr"; è quindi disponibile (senza il prefisso) come proprietà utente del messaggio JMS. Ad esempio, la WebSphere MQ *usr.myproperty* (una stringa di caratteri) è accessibile a un'applicazione JMS utilizzando la chiamata JMS `message.getStringProperty('myproperty')`. Notare che le applicazioni JMS non sono in grado di accedere alle proprietà con il prefisso "usr" se contengono due o più U+002E (".") caratteri. Una proprietà senza prefisso e senza U+002E (".") viene considerato come se avesse il prefisso "usr". Al contrario, è possibile accedere a una proprietà utente impostata in un'applicazione JMS in un'applicazione WebSphere MQ aggiungendo "usr." al nome proprietà richiesto in una chiamata MQINQMP.

Proprietà del messaggio e lunghezza del messaggio

Utilizzare l'attributo del gestore code *MaxPropertiesLength* per controllare la dimensione delle proprietà che possono fluire con qualsiasi messaggio in un gestore code WebSphere MQ .

In generale, quando si usa MQSETMP per impostare le proprietà, la dimensione di una proprietà è la lunghezza del nome della proprietà in byte, più la lunghezza del valore della proprietà in byte come passato nella chiamata MQSETMP. È possibile che la serie di caratteri del nome della proprietà e il valore della proprietà vengano modificati durante la trasmissione del messaggio alla relativa destinazione perché possono essere convertiti in Unicode; in questo caso, la dimensione della proprietà potrebbe essere modificata.

In una chiamata MQPUT o MQPUT1 , le proprietà del messaggio non vengono conteggiate per la lunghezza del messaggio per la coda e il gestore code, ma vengono conteggiate per la lunghezza delle proprietà percepite dal gestore code (indipendentemente dal fatto che siano state impostate utilizzando o meno le chiamate MQI della proprietà del messaggio).

Se la dimensione delle proprietà supera la lunghezza massima, il messaggio viene rifiutato con MQRC_PROPERTIES_TOO_BIG. Poiché la dimensione delle proprietà dipende dalla relativa rappresentazione, è necessario impostare la lunghezza massima delle proprietà ad un livello lordo.

È possibile che un'applicazione inserisca correttamente un messaggio con un buffer maggiore del valore di *MaxMsgLength*, se il buffer include le proprietà. Questo perché, anche quando rappresentate come elementi MQRFH2 , le proprietà del messaggio non vengono conteggiate per la lunghezza del messaggio. I campi di intestazione MQRFH2 vengono aggiunti alla lunghezza delle proprietà solo se una o più cartelle sono contenute e ogni cartella nell'intestazione contiene proprietà. Se una o più cartelle sono contenute nell'intestazione MQRFH2 e qualsiasi cartella non contiene proprietà, i campi dell'intestazione MQRFH2 vengono conteggiati per la lunghezza del messaggio.

Su una chiamata MQGET, le proprietà del messaggio non vengono conteggiate nella lunghezza del messaggio per quanto riguarda la coda e il gestore code. Tuttavia, poiché le proprietà vengono conteggiate separatamente, è possibile che il buffer restituito da una chiamata MQGET sia maggiore del valore dell'attributo *MaxMsgLength* .

Non fare in modo che le applicazioni interrogino il valore di *MaxMsgLength* e quindi assegnino un buffer di questa dimensione prima di chiamare MQGET; assegnare invece un buffer che si consideri sufficientemente grande. Se MQGET ha esito negativo, assegnare un buffer guidato dalla dimensione del parametro *DataLength* .

Il parametro *DataLength* della chiamata MQGET restituisce la lunghezza in byte dei dati dell'applicazione e tutte le proprietà restituite nel buffer fornito, se non è stato specificato un handle del messaggio nella struttura MQGMO.

Il parametro *Buffer* della chiamata MQPUT contiene i dati del messaggio dell'applicazione da inviare e tutte le proprietà rappresentate nei dati del messaggio.

Quando si passa a un gestore code precedente alla versione 7.0 del prodotto, le proprietà del messaggio, eccetto quelle nel descrittore del messaggio, vengono conteggiate per la lunghezza del messaggio. Pertanto, si consiglia di aumentare il valore dell'attributo *MaxMsgLength* dei canali che vanno su un sistema precedente alla versione 7.0 come necessario, per compensare il fatto che potrebbero essere inviati più dati per ogni messaggio. In alternativa, è possibile ridurre la coda o il gestore code *MaxMsgLength*, in modo che il livello complessivo dei dati inviati intorno al sistema rimanga lo stesso.

Esiste un limite di lunghezza di 100 MB per le proprietà del messaggio, escluso il descrittore del messaggio o l'estensione per ogni messaggio.

La dimensione di una proprietà nella sua rappresentazione interna è la lunghezza del nome, più la dimensione del suo valore, più alcuni dati di controllo per la proprietà. Ci sono anche alcuni dati di controllo per la serie di proprietà dopo che una proprietà è stata aggiunta al messaggio.

Nomi proprietà

Un nome proprietà è una stringa di caratteri. Alcune restrizioni si applicano alla sua lunghezza e alla serie di caratteri che possono essere utilizzati.

Un nome proprietà è una stringa di caratteri sensibile al maiuscolo / minuscolo, limitata a +4095 caratteri a meno che non sia diversamente limitato dal contesto. Questo limite è contenuto nella costante MQ_MAX_PROPERTY_NAME_LENGTH.

Se si supera questa lunghezza massima quando si utilizza una chiamata MQI della proprietà del messaggio, la chiamata ha esito negativo con codice motivo MQRC_PROPERTY_NAME_LENGTH_ERR.

Poiché non esiste una lunghezza massima del nome proprietà in JMS, è possibile per un'applicazione JMS impostare un nome proprietà JMS valido che non sia un nome proprietà WebSphere MQ valido quando viene memorizzato in una struttura MQRFH2 .

In questo caso, quando analizzato, vengono utilizzati solo i primi 4095 caratteri del nome della proprietà; i seguenti caratteri vengono troncati. Ciò potrebbe far sì che un'applicazione che utilizza i selettori non riesca a corrispondere a una stringa di selezione o a corrispondere a una stringa quando non previsto, poiché più di una proprietà potrebbe troncare allo stesso nome. Quando un nome di proprietà viene troncato, WebSphereMQ emette un messaggio di log degli errori.

Tutti i nomi proprietà devono seguire le regole definite dalla specifica del linguaggio Java per gli identificativi Java, con l'eccezione che il carattere Unicode U+002E (.) è consentito come parte del nome,

ma non l'inizio. Le regole per gli identificatori Java sono uguali a quelle contenute nella specifica JMS per nomi di proprietà.

I caratteri spazio e gli operatori di comparazione non sono consentiti. I valori null incorporati sono consentiti in un nome proprietà ma non sono consigliati. Se si utilizzano valori null incorporati, ciò impedisce l'utilizzo della costante MQVS_NULL_TERMINATED quando viene utilizzata con la struttura MQCHARV per specificare le stringhe di lunghezza variabile.

Mantenere i nomi delle proprietà semplici, poiché le applicazioni possono selezionare i messaggi in base ai nomi delle proprietà e la conversione tra la serie di caratteri del nome e del selettore potrebbe causare un errore imprevisto della selezione.

I nomi delle proprietà WebSphere MQ utilizzano il carattere U+002E (.) per il raggruppamento logico delle proprietà. Divide lo spazio dei nomi per le proprietà. Le proprietà con i seguenti prefissi, in qualsiasi combinazione di lettere minuscole o maiuscole, sono riservate per l'uso da parte del prodotto:

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

Un buon modo per evitare conflitti di nomi consiste nel garantire che tutte le applicazioni prefissino le proprietà del messaggio con il nome del dominio Internet. Ad esempio, se si sta sviluppando un'applicazione utilizzando il nome dominio "`ourcompany.com`" è possibile denominare tutte le proprietà con il prefisso "`com.ourcompany`". Questa convenzione di denominazione consente anche di selezionare facilmente le proprietà; ad esempio, un'applicazione può richiedere informazioni su tutte le proprietà del messaggio che iniziano con "`com.ourcompany.%`".

Per ulteriori informazioni sull'utilizzo dei nomi delle proprietà, consultare Limitazioni dei nomi delle proprietà .

Limitazioni nome proprietà
Quando si assegna un nome a una proprietà, è necessario osservare alcune regole.

Le seguenti limitazioni si applicano ai nomi proprietà:

1. Una proprietà non deve iniziare con le seguenti stringhe:
   - "JMS" - riservato all'utilizzo da parte delle classi WebSphere MQ per JMS.
   - "usr.JMS" - non valido.

   Le uniche eccezioni sono le seguenti proprietà che forniscono i sinonimi per le proprietà JMS:

Proprietà	**Sinonimo di**
JMSCorrelationID	Root .MQMD.CorrelId o jms.Cid
JMSDeliveryMode	Root .MQMD.Persistence o jms.Dlv
JMSDestination	jms.Dst
JMSExpiration	Root .MQMD.Expiry o jms.Exp
JMSMessageID	Root .MQMD.MsgId
JMSPriority	Root .MQMD.Priority o jms.Pri
JMSRedelivered	Root .MQMD.BackoutCount

Proprietà	Sinonimo di
JMSReplyTo (una stringa codificata come URI)	Root .MQMD.ReplyToQ o Root .MQMD.ReplyToQMgr o jms.Rto
JMSTimestamp	Root .MQMD.PutDate o Root .MQMD.PutTime o jms.Tms
JMSType	mcd.Type o mcd.Set o mcd.Fmt
JMSXAppID	Root .MQMD.PutApplName
JMSXDeliveryCount	Root .MQMD.BackoutCount
JMSXGroupID	Root .MQMD.GroupId o jms.Gid
JMSXGroupSeq	Root .MQMD.MsgSeqNumber o jms.Seq
JMSXUserID	Root .MQMD.UserIdentifier

Questi sinonimi consentono ad un'applicazione MQI di accedere alle proprietà JMS in modo simile a WebSphere MQ classes per l'applicazione client JMS. Di queste proprietà, solo JMSCorrelationID, JMSReplyTo, JMSType, JMSXGroupIDe JMSXGroupSeq possono essere impostati utilizzando MQI.

Le proprietà JMS_IBM_ * disponibili in WebSphere MQ classes for JMS non sono disponibili mediante MQI. Le applicazioni MQI possono accedere ai campi a cui fanno riferimento le proprietà JMS_IBM_ * in altri modi.

2. Una proprietà non deve essere richiamata, in una combinazione di caratteri minuscoli o maiuscoli, "NULL", "TRUE", "FALSE", "NOT", "E", "OR", "BETWEEN", "LIKE", "IN", "IS" e "ESCAPE". Questi sono i nomi delle parole chiave SQL utilizzate nelle stringhe di selezione.
3. Un nome di proprietà che inizia con "mq " in qualsiasi combinazione di lettere minuscole o maiuscole e non iniziando "mq_usr" può contenere solo un "." U+002E). Più "." non sono consentiti nelle proprietà con tali prefissi.
4. Due "." i caratteri devono contenere altri caratteri; non è possibile avere un punto vuoto nella gerarchia. Allo stesso modo, un nome proprietà non può terminare con un "." punto.
5. Se un'applicazione imposta la proprietà "a.b" e quindi la proprietà "a.b.c", non è chiaro se nella gerarchia "b" contenga un valore o un altro raggruppamento logico. Una gerarchia di questo tipo è "contenuto misto" e questo non è supportato. L'impostazione di una proprietà che causa contenuto misto non è consentita.

Queste limitazioni di sono applicate dal meccanismo di convalida nel modo seguente:

- I nomi delle proprietà vengono convalidati quando si imposta una proprietà utilizzando la chiamata MQSETMP - Impostare proprietà messaggio , se la convalida è stata richiesta quando è stato creato l'handle del messaggio. Se viene eseguito un tentativo di convalida di una proprietà e non riesce a causa di un errore nella specifica del nome della proprietà, il codice di completamento è MQCC_FAILED con il seguente motivo:
  - MQRC_PROPERTY_NAME_ERROR per i motivi 1-4.
  - MQRC_MIXED_CONTENT_NOT_ALLOWED per il motivo 5.
- Non è garantito che i nomi delle proprietà specificate direttamente come elementi MQRFH2 vengano convalidati dalla chiamata MQPUT.

Campi del descrittore del messaggio come proprietà
La maggior parte dei campi descrittori di messaggi può essere considerata come proprietà. Il nome della proprietà viene creato aggiungendo un prefisso al nome del campo descrittore del messaggio.

Se un'applicazione MQI desidera identificare una proprietà del messaggio contenuta in un campo del descrittore del messaggio, ad esempio, in una stringa del selettore o utilizzando le API della proprietà del messaggio, utilizzare la seguente sintassi:

Nome della proprietà	campo Descrittore messaggio
Root.MQMD. < Campo>	< Campo>

Specificare <Field> con lo stesso caso dei campi della struttura MQMD nella dichiarazione del linguaggio C. Ad esempio, il nome proprietà Root.MQMD.AccountingToken accede al campo AccountingToken del descrittore del messaggio.

I campi StrucId e Version del descrittore del messaggio non sono accessibili utilizzando la sintassi visualizzata.

I campi del descrittore del messaggio non vengono mai rappresentati in un'intestazione di MQRFH2 come per altre proprietà.

Se i dati del messaggio iniziano con un MQMDE rispettato dal gestore code, è possibile accedere ai campi MQMDE utilizzando la notazione Root.MQMD.<Field> descritta. In questo caso, i campi MQMDE vengono trattati come parte logicamente di MQMD dalla prospettiva delle proprietà. Consultare la sezione "MQMDE specificato nelle chiamate MQPUT e MQPUT1 " in Panoramica di MQMDE.

Valori e tipi di dati della proprietà

Una proprietà può essere un valore booleano, una stringa di byte, una stringa di caratteri o un numero intero o a virgola mobile. La proprietà può memorizzare qualsiasi valore valido nell'intervallo del tipo di dati, a meno che non sia altrimenti limitato dal contesto.

Il tipo di dati di un valore di proprietà deve essere uno dei seguenti:

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

Una proprietà può esistere ma non ha un valore definito; è una proprietà null. Una proprietà null è diversa da una proprietà byte (MQBYTE []) o da una proprietà stringa di caratteri (MQCHAR []) in quanto ha un valore definito ma vuoto, ossia uno con un valore di lunghezza zero.

La stringa di byte non è un tipo di dati proprietà valido in JMS o XMS. Si consiglia di non utilizzare le proprietà della stringa di byte nella cartella < usr>.

Selezione dei messaggi dalle code

È possibile selezionare i messaggi dalle code utilizzando i campi MsgId e CorrelId su una chiamata MQGET oppure utilizzando una SelectionString su una chiamata MQOPEN o MQSUB.

Selettori

Un selettore messaggi è una stringa a lunghezza variabile utilizzata da un'applicazione per registrare il suo interesse solo per quei messaggi che hanno proprietà che soddisfano la query SQL (Structured Query Language) rappresentata dalla stringa di selezione.

Selezione utilizzando le chiamate di funzione MQSUB e MQOPEN

Si utilizza il *SelectionString*, che è una struttura di tipo MQCHARV, per effettuare selezioni utilizzando le chiamate MQSUB e MQOPEN.

La struttura *SelectionString* viene utilizzata per passare una stringa di selezione a lunghezza variabile al gestore code.

Il CCSID associato alla stringa del selettore viene impostato tramite il campo VSCCSID della struttura MQCHARV. Il valore utilizzato deve essere un CCSID supportato per le stringhe selettore. Consultare Conversione codepage per un elenco di codepage supportate.

Se si specifica un CCSID per cui non esiste alcuna conversione Unicode supportata WebSphere MQ , si verifica un errore di MQRC_SOURCE_CCSID_ERROR. Questo errore viene restituito nel momento in cui il selettore viene presentato al gestore code, ossia nella chiamata MQSUB, MQOPEN o MQPUT1 .

Il valore predefinito per il campo *VSCCSID* è MQCCSI_APPL, che indica che il CCSID della stringa di selezione è uguale al CCSID del gestore code o al CCSID del client se è connesso tramite un client. La costante MQCCSI_APPL può tuttavia essere sovrascritta da un'applicazione che la ridefinisce prima della compilazione.

Se il selettore MQCHARV rappresenta una stringa NULL, non viene effettuata alcuna selezione per tale consumatore di messaggi e i messaggi vengono consegnati come se un selettore non fosse stato utilizzato.

La lunghezza massima di una stringa di selezione è limitata solo da quanto può essere descritto dal campo MQCHARV *VSLength*.

La SelectionString viene restituita sull'output da una chiamata MQSUB utilizzando l'opzione di sottoscrizione MQSO_RESUME, se è stato fornito un buffer ed è presente una lunghezza buffer positiva in VSBufSize. Se non si fornisce un buffer, viene restituita solo la lunghezza della stringa di selezione nel campo VSLength di MQCHARV. Se il buffer fornito è inferiore allo spazio richiesto per restituire il campo, nel buffer fornito vengono restituiti solo VSBufSize byte.

Un'applicazione non può modificare una stringa di selezione senza prima chiudere l'handle alla coda (per MQOPEN) o la sottoscrizione (per MQSUB). Una nuova stringa di selezione può essere specificata in una chiamata MQOPEN o MQSUB successiva.

MQOPEN
Utilizzare MQCLOSE per chiudere l'handle aperto, quindi specificare una nuova stringa di selezione su una chiamata MQOPEN successiva.

MQSUB
Utilizzare MQCLOSE per chiudere l'handle di sottoscrizione restituito (hSub), quindi specificare una nuova stringa di selezione su una chiamata MQSUB successiva.

Figura 3 a pagina 24 mostra il processo di selezione utilizzando la chiamata MQSUB.

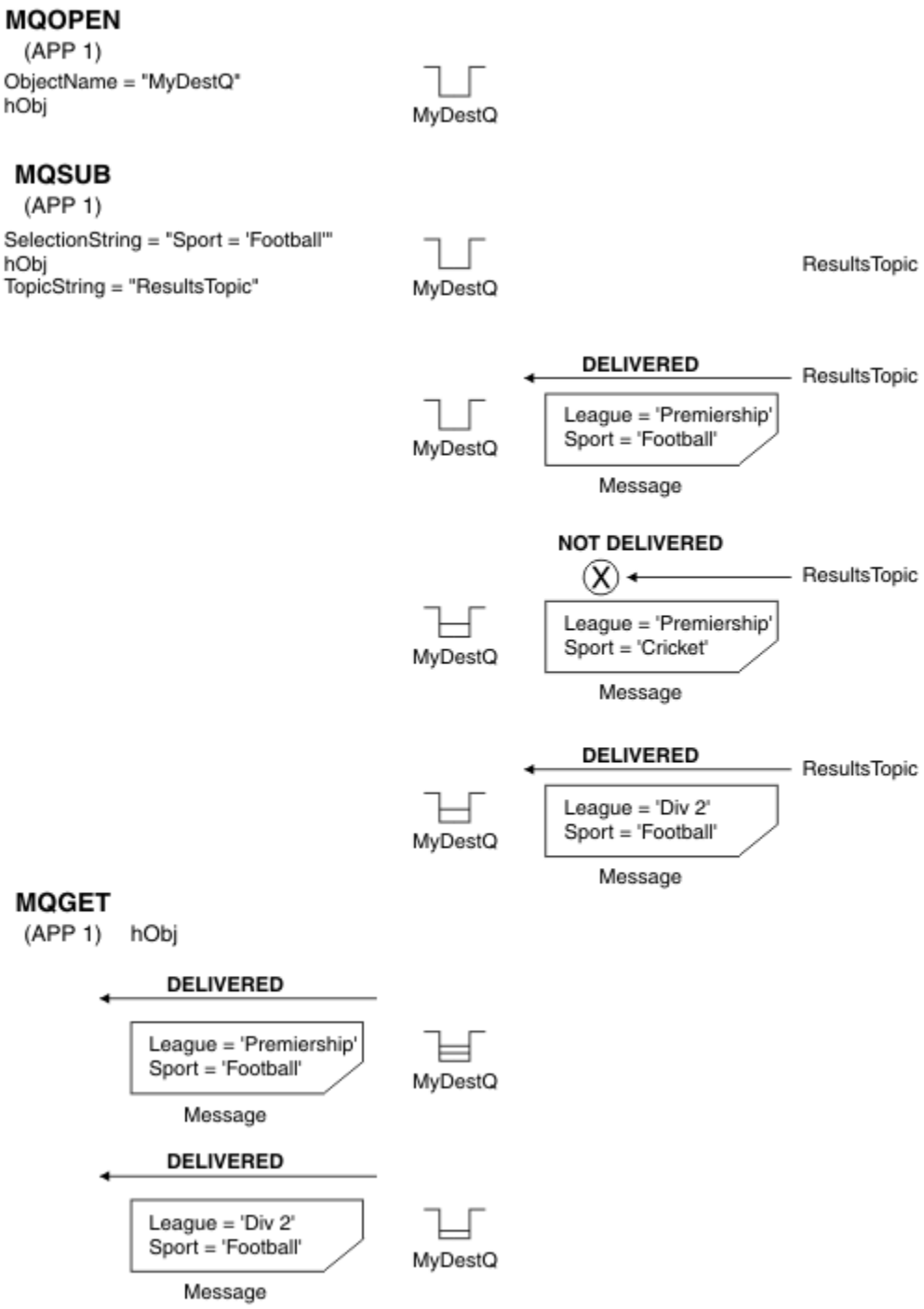


Figura 3. Selezione mediante chiamata MQSUB

È possibile passare un selettore sulla chiamata a MQSUB utilizzando il campo *SelectionString* nella struttura MQSD. L'effetto del passaggio in un selettore su MQSUB è che solo i messaggi pubblicati sull'argomento sottoscritto, che corrispondono a una stringa di selezione fornita, vengono resi disponibili sulla coda di destinazione.

Figura 4 a pagina 25 mostra il processo di selezione utilizzando la chiamata MQOPEN.

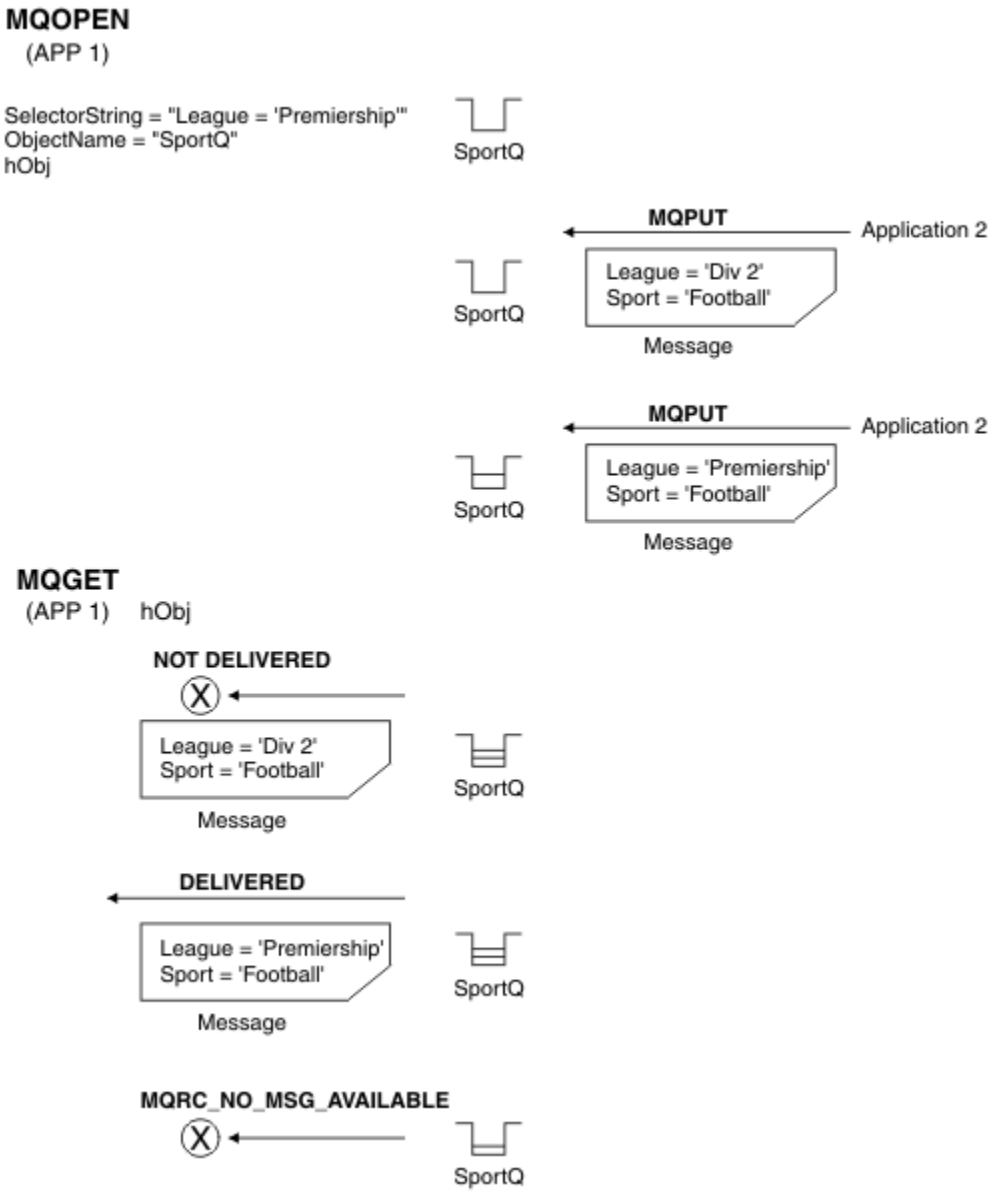


Figura 4. Selezione mediante chiamata MQOPEN

È possibile passare un selettore sulla chiamata a MQOPEN utilizzando il campo *SelectionString* nella struttura MQOD. L'effetto del passaggio in un selettore sulla chiamata MQOPEN è che solo i messaggi sulla coda aperta, che corrispondono a un selettore, vengono consegnati al consumatore di messaggi.

L'utilizzo principale per il selettore nella chiamata MQOPEN è per il caso point - to - point in cui un'applicazione può decidere di ricevere solo quei messaggi su una coda che corrispondono a un selettore. L'esempio precedente mostra uno scenario semplice in cui due messaggi vengono inseriti in una coda aperta da MQOPEN, ma solo uno viene ricevuto dall'applicazione che lo riceve, poiché è l'unico che corrisponde a un selettore.

Notare che le chiamate MQGET successive risultano in MQRC_NO_MSG_AVAILABLE poiché non esistono ulteriori messaggi sulla coda che corrispondono al selettore fornito.

Comportamento selezione
Panoramica del comportamento di selezione IBM WebSphere MQ .

I campi in una struttura MQMDE vengono considerati come le proprietà del messaggio per le proprietà del descrittore del messaggio corrispondenti se MQMD:

- Ha formato MQFMT_MD_EXTENSION
- È immediatamente seguito da una struttura MQMDE valida
- È la versione uno o contiene solo i campi della versione due predefiniti

È possibile che una stringa di selezione si risolva in TRUE o FALSE prima che venga eseguita qualsiasi corrispondenza con le proprietà del messaggio. Ad esempio, potrebbe essere il caso se la stringa di selezione è impostata su "TRUE <>FALSE". Tale valutazione iniziale è garantita solo quando non vi sono riferimenti di proprietà del messaggio nella stringa di selezione.

Se una stringa di selezione si risolve in TRUE prima che vengano considerate le proprietà del messaggio, vengono consegnati tutti i messaggi pubblicati nell'argomento sottoscritto dal consumer. Se una stringa di selezione si risolve in FALSE prima che vengano prese in considerazione le proprietà del messaggio, viene restituito un codice motivo di MQRC_SELECTOR_ALWAYS_FALSE e il codice di completamento MQCC_FAILED nella chiamata della funzione che ha presentato il selettore.

Anche se un messaggio non contiene proprietà del messaggio (diverse dalle proprietà dell'intestazione), può essere ancora idoneo per la selezione. Se una stringa di selezione fa riferimento a una proprietà del messaggio che non esiste, si presume che questa proprietà abbia il valore NULL o 'Sconosciuto'.

Ad esempio, un messaggio potrebbe ancora soddisfare una stringa di selezione come `'Color IS NULL'`, dove `'Color'` non esiste come proprietà del messaggio nel messaggio.

La scelta può essere eseguita solo sulle proprietà associate a un messaggio, non sul messaggio stesso, a meno che non sia disponibile un provider di selezione messaggi esteso. La selezione può essere eseguita sul payload del messaggio solo se è disponibile un provider di selezione del messaggio esteso.

Ad ogni proprietà del messaggio è associato un tipo. Quando si esegue una selezione, è necessario verificare che i valori utilizzati nelle espressioni per verificare le proprietà del messaggio siano del tipo corretto. Se si verifica una mancata corrispondenza del tipo, l'espressione in questione si risolve in FALSE.

È responsabilità dell'utente assicurarsi che la stringa di selezione e le proprietà del messaggio utilizzino tipi compatibili.

I criteri di selezione continuano ad essere applicati per conto dei sottoscrittori durevoli inattivi, in modo che vengano conservati solo i messaggi che corrispondono alla stringa di selezione originariamente fornita.

Le stringhe di selezione non sono modificabili quando una sottoscrizione durevole viene ripresa con alter (MQSO_ALTER). Se viene presentata una stringa di selezione diversa quando un sottoscrittore durevole riprende l'attività, allora MQRC_SELECTOR_NOT_ALTERABLE viene restituito all'applicazione.

Le applicazioni ricevono un codice di ritorno di MQRC_NO_MSG_AVAILABLE se non è presente alcun messaggio su una coda che soddisfa i criteri di selezione.

Se un'applicazione ha specificato una stringa di selezione contenente valori di proprietà, solo i messaggi che contengono proprietà corrispondenti sono idonei per la selezione. Ad esempio, un sottoscrittore specifica una stringa di selezione "a = 3" e viene pubblicato un messaggio che non contiene proprietà o proprietà in cui 'a' non esiste o non è uguale a 3. Il sottoscrittore non riceve il messaggio nella coda di destinazione.

Prestazioni di messaggistica

La selezione dei messaggi da una coda richiede IBM WebSphere MQ per ispezionare in modo sequenziale ogni messaggio sulla coda. I messaggi vengono ispezionati fino a quando non viene trovato un messaggio che corrisponde ai criteri di selezione o non ci sono più messaggi da esaminare. Pertanto, le prestazioni della messaggistica subiscono un peggioramento se la selezione dei messaggi viene utilizzata su code profonde.

Per ottimizzare la selezione di messaggi su code profonde quando la selezione è basata su JMSCorrelationID o JMSMessageID, utilizzare una stringa di selezione del modulo `JMSCorrelationID = ...` o `JMSMessageID = ...` e fare riferimento ad una sola proprietà.

Questo metodo offre un miglioramento significativo delle prestazioni per la selezione su JMSCorrelationID e offre un miglioramento delle prestazioni marginale per JMSMessageID.

Utilizzo di selettori complessi

I selettori possono contenere molti componenti, ad esempio:

a e b o c e d o e e f o g e h o i e j ... o y e z

L'utilizzo di tali selettori complessi può avere gravi implicazioni sulle prestazioni e requisiti di risorse eccessivi. Come tale, IBM WebSphere MQ proteggerà il sistema non riuscendo a elaborare selettori eccessivamente complessi che potrebbero causare una carenza di risorse di sistema. La protezione può verificarsi dopo circa 100 test su alcune piattaforme, quindi i selettori che si avvicinano a quel numero di componenti potrebbero riscontrare errori. Si raccomanda che l'uso di selettori con molti componenti sia accuratamente provato e testato sulle piattaforme appropriate per garantire che i limiti di protezione non vengano raggiunti.

Le prestazioni e complessità dei selettori possono essere migliorate semplificandoli utilizzando ulteriori parentesi per combinare i componenti. Ad esempio:

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

Concetti correlati

Sintassi del selettore messaggi
Un selettore di messaggi WebSphere MQ è una stringa con sintassi basata su un sottoinsieme della sintassi dell'espressione condizionale SQL92 .

Selezione del contenuto di un messaggio
È possibile sottoscrivere in base a una selezione del contenuto del payload del messaggio (noto anche come filtro del contenuto), ma la decisione su quali messaggi devono essere consegnati a tale sottoscrizione non può essere eseguita direttamente da WebSphere MQ; è invece richiesto un provider di selezione dei messaggi esteso, ad esempio IBM Integration Bus, per elaborare i messaggi.

Sintassi del selettore messaggi
Un selettore di messaggi WebSphere MQ è una stringa con sintassi basata su un sottoinsieme della sintassi dell'espressione condizionale SQL92 .

L'ordine in cui viene valutato un selettore di messaggi è da sinistra a destra all'interno di un livello di precedenza. È possibile utilizzare le parentesi per modificare questo ordine. I valori letterali selettori predefiniti e i nomi degli operatori vengono scritti qui in maiuscolo; tuttavia, non sono sensibili al maiuscolo / minuscolo.

WebSphere MQ verifica la correttezza sintattica di un selettore messaggi nel momento in cui viene presentato. Se la sintassi della stringa di selezione non è corretta o il nome di una proprietà non è valido e non è disponibile un provider di selezione dei messaggi estesi, `MQRC_SELECTION_NOT_AVAILABLE` viene restituito all'applicazione. Se la sintassi della stringa di selezione non è corretta o il nome di una proprietà non è valido quando viene ripresa una sottoscrizione, viene restituito un `MQRC_SELECTOR_SYNTAX_ERROR` all'applicazione. Se la convalida del nome della proprietà è stata disabilitata quando la proprietà è stata impostata (impostando `MQCMHO_NONE` invece di `MQCMHO_VALIDATE`) e un'applicazione successivamente inserisce un messaggio con un nome proprietà non valido, questo messaggio non viene mai selezionato.

Un selettore può contenere:

- Valori letterali:
  - I valori letterali stringa sono racchiusi tra virgolette singole. Due virgolette singole consecutive rappresentano una virgoletta singola. Gli esempi sono 'literal' e 'literal' '. Come i letterali stringa Java, questi utilizzano la codifica dei caratteri Unicode. Non è possibile utilizzare le virgolette doppie per racchiudere un letterale stringa. Qualsiasi sequenza di byte può essere utilizzata tra virgolette singole.

– Una stringa di byte è una o più coppie di caratteri esadecimali racchiusi tra virgolette doppie e preceduti da 0x. Gli esempi sono "0x2F1C" o "0XD43A". La lunghezza di una stringa di byte deve essere almeno un byte. Se una stringa di byte del selettore corrisponde a una proprietà del messaggio di tipo MQTYPE_BYTE_STRING, non viene eseguita alcuna azione speciale sullo zero iniziale o finale. I byte vengono trattati come un altro carattere. Anche l'endianità non è considerata. La lunghezza delle stringhe di byte del selettore e della proprietà deve essere uguale e la sequenza di byte deve essere la stessa.

  Esempi di selezioni di stringhe di byte (si supponga che *myBytes* = 0AFC23) che corrispondono sono:

  - "myBytes = "0x0AFC23"" = TRUE

  Le seguenti selezioni di stringa non corrispondono:

  - "myBytes = "0xAFC23"" = MQRC_SELECTOR_SYNTAX_ERROR (perché il numero di byte non è multiplo di due)
  - "myBytes = "0x0AFC2300"" = FALSE (perché lo zero finale è significativo nel confronto)
  - "myBytes = "0x000AFC23"" = FALSE (perché lo zero iniziale è significativo nel confronto)
  - "myBytes = "0x23FC0A"" = FALSE (perché l'endianità non viene considerata)
– I numeri esadecimali iniziano con uno zero, seguiti da un xmaiuscolo o minuscolo. Il resto della costante letterale contiene uno o più caratteri esadecimali validi. Esempi sono 0xA, 0xAF, 0X2020.
– Uno zero iniziale seguito da una o più cifre nell'intervallo 0 - 7 viene sempre interpretato come l'inizio di un numero ottale. Non è possibile rappresentare un numero decimale con prefisso zero come questo, ad esempio, 09 restituisce un errore di sintassi perché 9 non è una cifra ottale valida. Esempi di numeri ottali sono 0177, 0713.
– Una costante letterale numerica esatta è un valore numerico senza un punto decimale, ad esempio 57, -957e +62. Una costante letterale numerica esatta può avere una L maiuscola o minuscola finale; ciò non influisce sul modo in cui il numero viene memorizzato o interpretato. WebSphere MQ supporta numeri esatti compresi nell'intervallo -9,223,372,036,854,775,808 - 9,223,372,036,854,775,807.
– Un valore letterale numerico approssimativo è un valore numerico in notazione scientifica, ad esempio 7E3 o -57.9E2, oppure un valore numerico con un decimale, ad esempio 7., -95.7o +6.2. WebSphere MQ supporta i numeri compresi tra -1.797693134862315E+308 e 1.797693134862315E+308.

  Il significando deve seguire un segno facoltativo (+ o -). Il significando deve essere un numero intero o una frazione. Una parte frazionaria del significando non deve necessariamente avere una cifra iniziale.

  Un E maiuscolo o minuscolo indica l'inizio di un esponente facoltativo. L'esponente ha una radice decimale e la parte numerica dell'esponente può essere preceduta da un segno facoltativo.

  Le costanti letterali numeriche approssimate possono terminare con un carattere F o D (non sensibile al maiuscolo / minuscolo). Questa sintassi esiste per supportare il metodo cross - language di tagging di numeri di precisione singoli o doppi. Questi caratteri sono facoltativi e non influiscono sul modo in cui una costante letterale numerica approssimativa viene memorizzata o elaborata. Questi numeri vengono sempre memorizzati ed elaborati utilizzando la doppia precisione.
– I letterali booleani TRUE e FALSE.

Nota: Le rappresentazioni IEEE-754 non finite come NaN, +Infinity, -Infinity non sono supportate nelle stringhe di selezione. Non è quindi possibile utilizzare questi valori come operandi in un'espressione. Lo zero negativo viene trattato come lo zero positivo per le operazioni matematiche.

- Identificativi:

Un identificativo è una sequenza di caratteri a lunghezza variabile che deve iniziare con un carattere di inizio identificativo valido, seguito da zero o più caratteri di parte identificativo validi. Le regole per i nomi identificativo sono uguali a quelle per i nomi delle proprietà del messaggio, consultare “Nomi proprietà” a pagina 19 e “Limitazioni nome proprietà” a pagina 20 per ulteriori informazioni.

Nota: La selezione può essere eseguita sul payload del messaggio solo se è disponibile un provider di selezione del messaggio esteso.

Gli identificatori sono riferimenti di campi di intestazione o riferimenti di proprietà. Il tipo di valore di una proprietà in un selettore di messaggi deve corrispondere al tipo utilizzato per impostare la proprietà, anche se la promozione numerica viene eseguita dove possibile. Se si verifica una mancata corrispondenza del tipo, il risultato dell'espressione è FALSE. Se si fa riferimento a una proprietà che non esiste in un messaggio, il suo valore è `NULL`.

Le conversioni di tipo che si applicano ai metodi get per le proprietà non si applicano quando una proprietà viene utilizzata in un'espressione del selettore messaggi. Ad esempio, se si imposta una proprietà come valore stringa e si utilizza un selettore per eseguire la query come valore numerico, l'espressione restituisce FALSE.

I nomi campo e proprietà JMS associati ai nomi proprietà o ai nomi campo `MQMD` sono anche identificatori validi in una stringa di selezione. WebSphere MQ associa i nomi di proprietà e campo JMS riconosciuti ai valori delle proprietà del messaggio. Per ulteriori informazioni, fare riferimento a “Selettori di messaggi in JMS” a pagina 809. Ad esempio, la stringa di selezione `"JMSPriority >="` viene selezionata nella proprietà `Pri` della cartella `jms` del messaggio corrente.

- Eccedenza / insufficienza:

  Sia per i numeri decimali che per i numeri numerici approssimativi, non sono definiti i seguenti valori:

  – Specifica di un numero non compreso nell'intervallo definito
  – Specifica di un'espressione aritmetica che causerebbe un overflow o un underflow

  Non vengono eseguite verifiche per queste condizioni.

- Spazio vuoto:

  Definito come spazio, avanzamento pagina, nuova riga, ritorno a capo, tabulazione orizzontale o tabulazione verticale. I seguenti caratteri Unicode sono riconosciuti come spazi vuoti:

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

- Espressioni:

  – Un selettore è un'espressione condizionale. Un selettore che assume il valore di true corrisponde; un selettore che assume il valore di false o unknown non corrisponde.
  – Le espressioni aritmetiche sono costituite da se stesse, da operazioni aritmetiche, da identificativi (il valore dell'identificativo viene trattato come una costante letterale numerica) e da costanti letterali numeriche.
  – Le espressioni condizionali sono composte da se stesse, operazioni di confronto e operazioni logiche.

- La parentesi standard (), per impostare l'ordine in cui vengono valutate le espressioni, è supportata.

- Operatori logici in ordine di precedenza: NOT, AND, OR.
- Operatori di confronto: =, >, >=, <, <=, <> (non uguale).
  - Due stringhe di byte sono uguali solo se le stringhe sono della stessa lunghezza e la sequenza di byte è uguale.
  - È possibile confrontare solo i valori dello stesso tipo. Un'eccezione è che è valido per confrontare i valori numerici esatti e i valori numerici approssimati (la conversione del tipo richiesta è definita dalle regole della promozione numerica Java). Se si tenta di confrontare tipi diversi, il selettore è sempre false.
  - La stringa e il confronto booleano sono limitati a = e <>. Due stringhe sono uguali solo se contengono la stessa sequenza di caratteri.
- Operatori aritmetici in ordine di precedenza:
  - +, - unario.
  - * moltiplicazione e / divisione.
  - + addizione e - sottrazione.
  - Le operazioni aritmetiche su un valore NULL non sono supportate. Se vengono tentati, il selettore completo è sempre false.
  - Le operazioni aritmetiche devono utilizzare la promozione numerica Java.
- Operatore di confronto arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 e arithmetic-expr3 :
  - `Age BETWEEN 15 and 19` è equivalente a `age >= 15 AND age <= 19`.
  - `Age NOT BETWEEN 15 and 19` è equivalente a `age < 15 OR age > 19`.
  - Se una qualsiasi delle espressioni di un'operazione BETWEEN è NULL, il valore dell'operazione è false. Se una delle espressioni di un'operazione NOT BETWEEN è NULL, il valore dell'operazione è true.
- operatore di confronto identificativo `[NOT] IN (string-literal1, string-literal2,...)` dove l'identificativo ha un valore Stringa o NULL .
  - `Country IN ('UK', 'US', 'France')` è true per `'UK'` e false per `'Peru'`. Equivale all'espressione `(Country = 'UK') OR (Country = 'US') OR (Country = 'France')`.
  - `Country NOT IN ('UK', 'US', 'France')` è false per `'UK'` e true per `'Peru'`. Equivale all'espressione `NOT ((Country = 'UK') OR (Country = 'US') OR (Country = 'France'))`.
  - Se l'identificativo di un'operazione IN o NOT IN è NULL, il valore dell'operazione è sconosciuto.
- `identifier [NOT] LIKE` *`pattern-value`* `[ESCAPE` *`escape-character`*`]` operatore di confronto, dove `identifier` ha un valore stringa. *pattern - value* è una stringa letterale, dove _ sta per qualsiasi carattere singolo e % sta per qualsiasi sequenza di caratteri (inclusa la sequenza vuota). Tutti gli altri personaggi rappresentano se stessi. Il *carattere di escape* facoltativo è un valore letterale stringa di caratteri singolo utilizzato per eseguire l'escape del significato speciale di _ e % in *valore - modello*. L'operatore LIKE deve essere utilizzato solo per confrontare due valori stringa.
  - `phone LIKE '12%3'` è true per 123 e 12993 e false per 1234.
  - `word LIKE 'l_se'` è true per la perdita e false per la perdita.
  - `underscored LIKE '\_%' ESCAPE '\'` è true per `_foo` e false per `bar`.
  - `phone NOT LIKE '12%3'` è false per 123 e 12993 e true per 1234.
  - Se l'identificativo di un'operazione LIKE o NOT LIKE è NULL, il valore dell'operazione è sconosciuto.

  Nota: L'operatore LIKE deve essere utilizzato per confrontare due valori stringa. Il valore di `Root.MQMD.CorrelId` è una schiera di byte a 24 byte, non una stringa di caratteri. La stringa del selettore `Root.MQMD.CorrelId LIKE 'ABC%'` viene accettata dal parser come sintatticamente valida, ma viene valutata come false. Quando si confronta un array di byte con una stringa di caratteri, LIKE non può essere utilizzato.
- test dell'operatore di confronto `identifier IS NULL` per un valore del campo di intestazione NULL o per un valore della proprietà mancante.

- L'operatore di confronto `identifier IS NOT NULL` verifica l'esistenza di un valore di campo di intestazione non null o di un valore di proprietà.
- Valori null

  La valutazione delle espressioni del selettore che contengono valori `NULL` è definita dalla semantica SQL 92 `NULL` , in sintesi:

  – SQL considera un valore `NULL` come sconosciuto.
  – Il confronto o l'aritmetica con un valore sconosciuto produce sempre un valore sconosciuto.
  – Gli operatori `IS NULL` e `IS NOT NULL` convertono un valore sconosciuto in valori `TRUE` e `FALSE` .

  Gli operatori booleani utilizzano la logica a tre valori (`T=TRUE, F=FALSE, U=UNKNOWN`)

Tabella 1. Risultato dell'operatore booleano quando la logica è `A AND B`

Operatore A	**Operatore B**	**Risultato (A E 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

Tabella 2. Risultato dell'operatore booleano quando la logica è `A OR B`

Operatore A	**Operatore B**	**Risultato (`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

Tabella 3. Risultato dell'operatore booleano quando la logica è `NOT A`

Operatore A	**Risultato (`NOT A`)**
T	F
F	T
U	U

Il seguente selettore di messaggi seleziona i messaggi con un tipo di messaggio di auto, un colore di blu e un peso maggiore di 2500 libbre:

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

Sebbene SQL supporti il confronto decimale fisso e l'aritmetica, i selettori dei messaggi non lo fanno. Questo è il motivo per cui le costanti letterali numeriche esatte sono limitate a quelle senza un decimale. È anche il motivo per cui ci sono numeri con un decimale come rappresentazione alternativa per un valore numerico approssimativo.

I commenti SQL non sono supportati.

Concetti correlati

Comportamento selezione
Panoramica del comportamento di selezione IBM WebSphere MQ .

Selezione del contenuto di un messaggio
È possibile sottoscrivere in base a una selezione del contenuto del payload del messaggio (noto anche come filtro del contenuto), ma la decisione su quali messaggi devono essere consegnati a tale sottoscrizione non può essere eseguita direttamente da WebSphere MQ; è invece richiesto un provider di selezione dei messaggi esteso, ad esempio IBM Integration Bus, per elaborare i messaggi.

“Proprietà dei messaggi” a pagina 18
Utilizzare le proprietà del messaggio per consentire a un'applicazione di selezionare i messaggi da elaborare o per richiamare le informazioni su un messaggio senza accedere alle intestazioni MQMD o MQRFH2 . Inoltre, facilitano la comunicazione tra le applicazioni WebSphere MQ e JMS.

Riferimenti correlati

MsgHandle
MQBUFMH - Converti buffer in handle del messaggio

Regole e limitazioni della stringa di selezione
Familiarizzare con queste regole su come vengono interpretate le stringhe di selezione e le limitazioni di caratteri per evitare potenziali problemi quando si utilizzano i selettori.

- L'equivalenza viene verificata utilizzando un singolo carattere uguale; ad esempio, `a = b` è corretto, mentre `a == b` è errato.
- Un operatore utilizzato da molti linguaggi di programmazione per rappresentare 'non uguale a' è `!=`. Questa rappresentazione non è un sinonimo valido per <>; ad esempio, `a <> b` è valido, mentre `a != b` non è valido.
- Le virgolette singole vengono riconosciute solo se il ' Viene utilizzato il carattereU+0027. Allo stesso modo, le virgolette doppie, valide solo quando utilizzate per racchiudere stringhe di byte, devono utilizzare il carattere " (U+0022).
- I simboli &, &&, | e || non sono sinonimi per congiunzione / disgiunzione logica; ad esempio, `a && b` deve essere specificato come `a AND b`.
- I caratteri jolly * e ? non sono sinonimi per % e _.
- I selettori contenenti espressioni composte come `20 < b < 30` non sono validi. Il programma di analisi valuta gli operatori che hanno la stessa precedenza da sinistra a destra. L'esempio diventa quindi `(20 < b) < 30`, che non ha senso. Invece, l'espressione deve essere scritta come `(b > 20) AND (b < 30)`.
- Le stringhe di byte devono essere racchiuse tra virgolette doppie; se vengono utilizzate virgolette singole, la stringa di byte viene considerata una stringa letterale. Il numero di caratteri (non il numero che i caratteri rappresentano) che seguono `0x` deve essere un multiplo di due.
- La parola chiave `IS` non è un sinonimo del carattere uguale. Pertanto, le stringhe di selezione `a IS 3` e `b IS 'red'` non sono valide. La parola chiave `IS` esiste solo per supportare i casi `IS NULL` e `IS NOT NULL` .

Concetti correlati

Considerazioni su UTF-8 e Unicode quando si utilizzano i selettori di messaggi

Considerazioni su UTF-8 e Unicode quando si utilizzano i selettori di messaggi

I caratteri, non racchiusi tra virgolette singole, che costituiscono le parole chiave riservate di una stringa di selezione devono essere immessi in Basic Latin Unicode (che va dal carattere U+0000 a U+0007F). Non è valido utilizzare altre rappresentazioni di punti di codice di caratteri alfanumerici. Ad esempio, il numero 1 deve essere espresso come U+0031 in Unicode, non è valido per utilizzare l'equivalente cifra a larghezza intera U+FF11 o l'equivalente arabo U+0661.

I nomi delle proprietà del messaggio possono essere specificati utilizzando qualsiasi sequenza valida di caratteri Unicode. I nomi delle proprietà dei messaggi contenuti nelle stringhe di selezione codificati in UTF-8 verranno convalidati anche se contengono caratteri multi - byte. La convalida di UTF-8 a più byte è rigorosa ed è necessario assicurarsi che per i nomi delle proprietà del messaggio siano utilizzate sequenze UTF-8 valide.

Non viene eseguita alcuna elaborazione supplementare sui valori o sui nomi delle proprietà durante il confronto per l'uguaglianza. Ciò significa, ad esempio, che non avviene alcuna pre / de - composizione e alle legature non viene dato alcun significato speciale. Ad esempio, il carattere umlaut precomposto U+00FC non è considerato equivalente a U+0075 + U+0308 e la sequenza di caratteri ff non è considerata equivalente a Unicode U+FB00 (LATIN SMALL LIGATURE FF)

I dati della proprietà racchiusi tra virgolette singole possono essere rappresentati da qualsiasi sequenza di byte e non vengono convalidati.

Concetti correlati

Regole e limitazioni della stringa di selezione
Familiarizzare con queste regole su come vengono interpretate le stringhe di selezione e le limitazioni di caratteri per evitare potenziali problemi quando si utilizzano i selettori.

Selezione del contenuto di un messaggio
È possibile sottoscrivere in base a una selezione del contenuto del payload del messaggio (noto anche come filtro del contenuto), ma la decisione su quali messaggi devono essere consegnati a tale sottoscrizione non può essere eseguita direttamente da WebSphere MQ; è invece richiesto un provider di selezione dei messaggi esteso, ad esempio IBM Integration Bus, per elaborare i messaggi.

Quando un'applicazione viene pubblicata su una stringa di argomenti, dove uno o più sottoscrittori hanno una stringa di selezione selezionata sul contenuto del messaggio, WebSphere MQ richiederà che il provider di selezione dei messaggi estesi analizzi la pubblicazione e informi WebSphere MQ se la pubblicazione corrisponde ai criteri di selezione specificati da ciascun sottoscrittore con un filtro del contenuto.

Se il provider di selezione del messaggio esteso determina che la pubblicazione corrisponde alla stringa di selezione del sottoscrittore, il messaggio continuerà ad essere consegnato al sottoscrittore.

Se il provider di selezione dei messaggi estesi determina che la pubblicazione non corrisponde, il messaggio non viene consegnato al sottoscrittore. Ciò potrebbe causare l'esito negativo della chiamata MQPUT o MQPUT1 con codice motivo MQRC_PUBLICATION_FAILURE. Se il provider di selezione dei messaggi estesi non è in grado di analizzare la pubblicazione, viene restituito il codice motivo MQRC_CONTENT_ERROR e la chiamata MQPUT o MQPUT1 ha esito negativo.

Se il provider di selezione dei messaggi estesi non è disponibile o non è in grado di determinare se il sottoscrittore (subscriber) deve ricevere la pubblicazione, viene restituito il codice motivo MQRC_SELECTION_NOT_AVAILABLE e la chiamata MQPUT o MQPUT1 ha esito negativo.

Quando una sottoscrizione viene creata con un filtro del contenuto e il provider di selezione dei messaggi estesi non è disponibile, la chiamata MQSUB ha esito negativo con codice di errore MQRC_SELECTION_NOT_AVAILABLE. Se viene ripresa una sottoscrizione con un filtro del contenuto e il provider di selezione dei messaggi estesi non è disponibile, la chiamata MQSUB restituisce un avviso di MQRC_SELECTION_NOT_AVAILABLE, ma la sottoscrizione può essere ripresa.

Concetti correlati

Comportamento selezione
Panoramica del comportamento di selezione IBM WebSphere MQ .

Sintassi del selettore messaggi
Un selettore di messaggi WebSphere MQ è una stringa con sintassi basata su un sottoinsieme della sintassi dell'espressione condizionale SQL92 .

Utilizzo asincrono dei messaggi IBM WebSphere MQ

L'utilizzo asincrono utilizza una serie di estensioni MQI (Message Queue Interface), MQI richiama MQCB e MQCTL, che consentono a un'applicazione MQI di essere scritta per utilizzare i messaggi da una serie di code. I messaggi vengono consegnati all'applicazione richiamando una 'unità di codice', identificata dall'applicazione che trasmette il messaggio o un token che rappresenta il messaggio.

Negli ambienti applicativi più semplici, l'unità di codice è definita da un puntatore di funzione, mentre in altri ambienti l'unità di codice può essere definita da un nome di programma o modulo.

Nell'utilizzo asincrono dei messaggi, vengono utilizzati i termini seguenti:

consumatore di messaggi
Un costrutto di programmazione che consente di definire un programma o una funzione da richiamare con un messaggio quando ne diventa disponibile uno che corrisponde al requisito delle applicazioni.

Gestore eventi
Un costrutto di programmazione che consente di definire un programma o una funzione da richiamare quando si verifica un evento asincrono, come la sospensione del gestore code.

Callback
Un termine generico utilizzato per fare riferimento a una routine Message Consumer o Event Handler.

Il consumo asincrono può semplificare la progettazione e l'implementazione di nuove applicazioni, in particolare quelle che elaborano più code di input o sottoscrizioni. Tuttavia, se si sta utilizzando più di una coda di input e si stanno elaborando i messaggi in sequenza di priorità, la sequenza di priorità viene osservata indipendentemente all'interno di ciascuna coda: è possibile che si ricevano messaggi a bassa priorità da una coda prima dei messaggi ad alta priorità da un'altra. L'ordine dei messaggi tra più code non è garantito. Notare inoltre che se si utilizzano le uscite API, potrebbe essere necessario modificarle per includere le chiamate MQCB e MQCTL.

Le seguenti illustrazioni forniscono un esempio di come utilizzare questa funzione.

Figura 5 a pagina 35 mostra un'applicazione a più thread che utilizza i messaggi da due code. L'esempio mostra tutti i messaggi consegnati a una funzione singola.

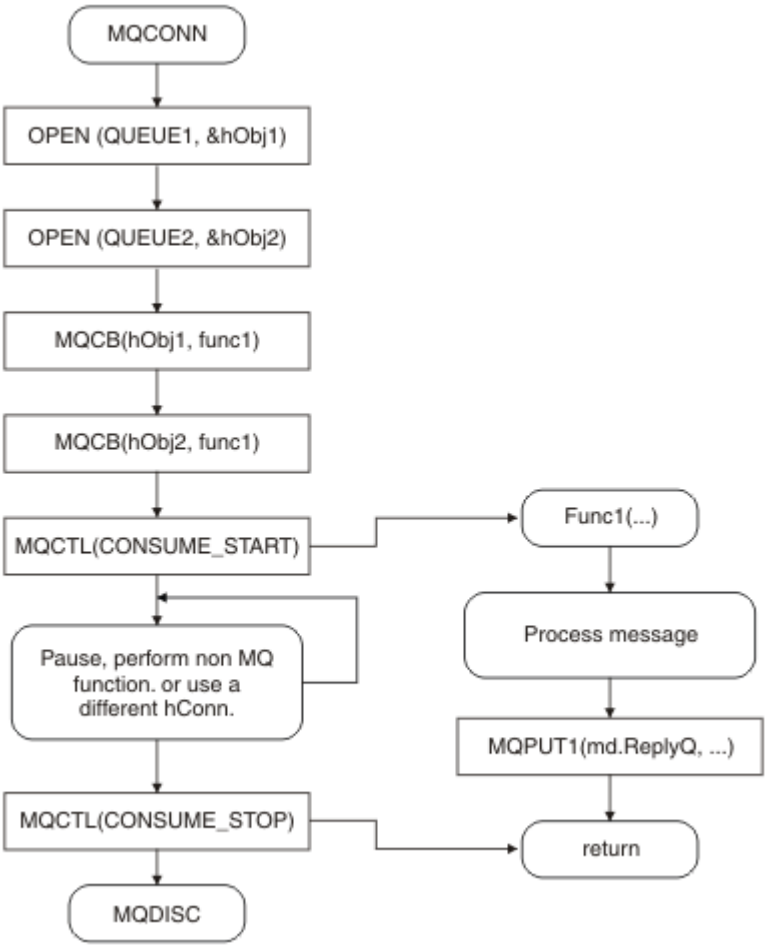


Figura 5. Applicazione basata sui messaggi standard che utilizza due code

Figura 6 a pagina 36 Questo flusso di esempio mostra un'applicazione a thread singolo che utilizza messaggi da due code. L'esempio mostra tutti i messaggi consegnati a una funzione singola.

La differenza rispetto al caso asincrono è che il controllo non ritorna all'emittente di MQCTL fino a quando tutti i consumer non si sono disattivati; vale a dire che un consumer ha emesso una richiesta MQCTL STOP o il gestore code è inattivo.

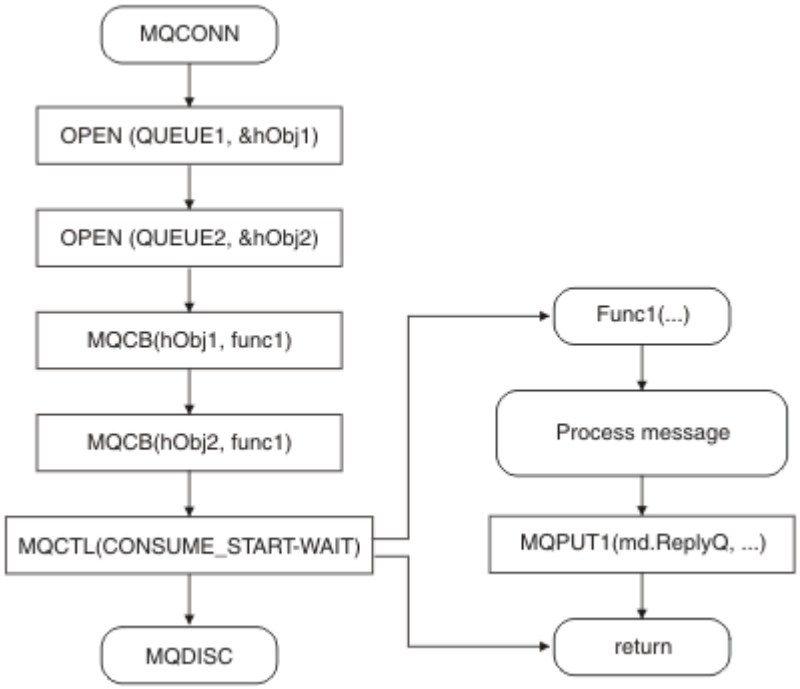


Figura 6. L'applicazione basata sui messaggi a thread singolo utilizza due code

Gruppi di messaggi

I messaggi possono verificarsi all'interno dei gruppi per consentire l'ordinamento dei messaggi.

I gruppi di messaggi consentono a più messaggi di essere contrassegnati come correlati l'uno all'altro e un ordine logico da applicare al gruppo (consultare “Ordinamento logico e fisico” a pagina 246). Su piattaforme diverse da z/OS, un concetto correlato, “Segmentazione del messaggio” a pagina 264 consente la suddivisione di messaggi di grandi dimensioni in segmenti più piccoli. Non è possibile utilizzare messaggi raggruppati o segmentati durante l'inserimento in un argomento.

La gerarchia all'interno di un gruppo è la seguente:

Gruppo
Questo è il livello più alto nella gerarchia ed è identificato da *GroupId*. Consiste in uno o più messaggi che contengono lo stesso *GroupId*. Questi messaggi possono essere memorizzati ovunque nella coda.

Nota: Il termine *messaggio* viene utilizzato qui per indicare un elemento su una coda, come ad esempio viene restituito da un singolo MQGET che non specifica MQGMO_COMPLETE_MSG.

Figura 7 a pagina 36 mostra un gruppo di messaggi logici:

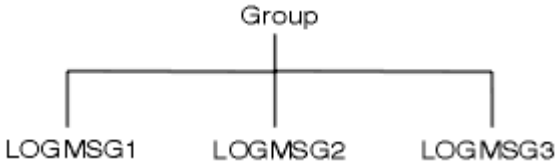


Figura 7. Gruppo di messaggi logici

Aprendo una coda e specificando MQOO_BIND_ON_GROUP, si forzano tutti i messaggi in un gruppo inviati a questa coda ad essere inviati alla stessa istanza della coda. Per ulteriori informazioni sull'opzione BIND_ON_GROUP, consultare Gestione delle affinità dei messaggi.

Messaggio logico

I messaggi logici all'interno di un gruppo vengono identificati dai campi *GroupId* e *MsgSeqNumber* . Il *MsgSeqNumber* inizia da 1 per il primo messaggio all'interno di un gruppo e se un messaggio non si trova in un gruppo, il valore del campo è 1.

Utilizzare messaggi logici all'interno di un gruppo per:

- Verificare l'ordine (se ciò non è garantito nelle circostanze in cui il messaggio viene trasmesso).
- Consentire alle applicazioni di raggruppare messaggi simili (ad esempio, quelli che devono essere elaborati dalla stessa istanza del server).

Ogni messaggio all'interno di un gruppo è costituito da un messaggio fisico, a meno che non sia suddiviso in segmenti. Ogni messaggio è logicamente un messaggio separato e solo i campi *GroupId* e *MsgSeqNumber* in MQMD devono avere una relazione con altri messaggi nel gruppo. Gli altri campi in MQMD sono indipendenti; alcuni potrebbero essere identici per tutti i messaggi nel gruppo, mentre altri potrebbero essere diversi. Ad esempio, i messaggi in un gruppo possono avere nomi di formato, CCSID e codifiche differenti.

Segmento

I segmenti vengono utilizzati per gestire i messaggi che sono troppo grandi per l'applicazione di inserimento o di richiamo o per il gestore code (inclusi i gestori code che intervengono attraverso i quali il messaggio passa). Per ulteriori informazioni, consultare “Segmentazione del messaggio” a pagina 264.

Un singolo messaggio viene suddiviso in messaggi più piccoli denominati *segmenti*. Un segmento di un messaggi è identificato dai campi *GroupId*, *MsgSeqNumber*e *Offset* . Il campo *Offset* inizia da zero per il primo segmento all'interno di un messaggio.

Ogni segmento è costituito da un messaggio fisico che potrebbe appartenere a un gruppo (Figura 8 a pagina 37 mostra un esempio di messaggi all'interno di un gruppo). Un segmento è logicamente parte di un singolo messaggio, quindi solo i campi *MsgId*, *Offset*e *SegmentFlag* in MQMD devono essere diversi tra segmenti separati dello stesso messaggio. Se un segmento non riesce ad arrivare, viene restituito il codice motivo MQRC_INCOMPLETE_GROUP o MQRC_INCOMPLETE_MSG come appropriato.

Figura 8 a pagina 37 mostra un gruppo di messaggi logici, alcuni dei quali sono segmentati:

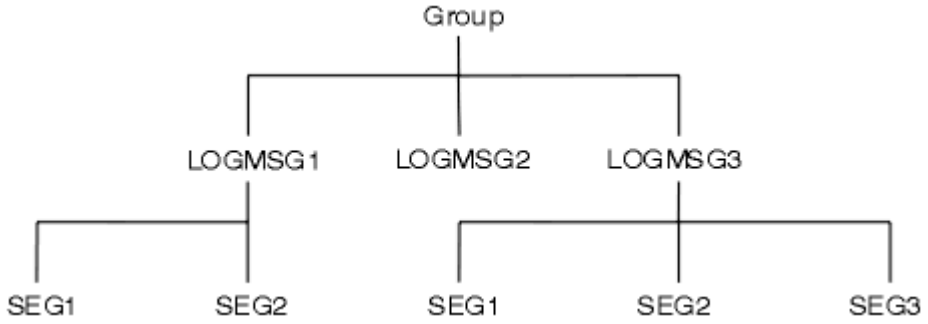


Figura 8. Messaggi segmentati

Non è possibile utilizzare messaggi segmentati o raggruppati con Pubblicazione / Sottoscrizione.

Per una descrizione dei messaggi logici e fisici, consultare “Ordinamento logico e fisico” a pagina 246. Per ulteriori informazioni sulla segmentazione dei messaggi, consultare “Segmentazione del messaggio” a pagina 264.

Persistenza messaggio

I messaggi persistenti vengono scritti nei log e nei file di dati della coda.

Se un gestore code viene riavviato dopo un errore, recupera questi messaggi persistenti come necessario dai dati registrati. I messaggi non persistenti vengono eliminati se un gestore code si arresta, se l'arresto è dovuto a un comando dell'operatore o a un malfunzionamento di una parte del sistema.

Quando si crea un messaggio, se si inizializza il descrittore del messaggio (MQMD) utilizzando i valori predefiniti, la persistenza per il messaggio viene ricavata dall'attributo *DefPersistence* della coda specificata nel comando MQOPEN. In alternativa, è possibile impostare la persistenza del messaggio utilizzando il campo di *Persistence* della struttura MQMD per definire il messaggio come persistente o non persistente.

Le prestazioni dell'applicazione vengono influenzate quando si utilizzano i messaggi persistenti; l'estensione dell'effetto dipende dalle caratteristiche delle prestazioni del sottosistema I/O della macchina e dal modo in cui si utilizzano le opzioni del punto di sincronizzazione su ciascuna piattaforma:

- Un messaggio persistente, al di fuori dell'unità di lavoro corrente, viene scritto su disco su ogni operazione put e get. Consultare “Commit e backout delle unità di lavoro” a pagina 325.
- In IBM WebSphere MQ su sistemi UNIX , IBM WebSphere MQ su sistemi Linux , e IBM WebSphere MQ per Windows, un messaggio persistente all'interno dell'unità di lavoro corrente viene registrato solo quando viene eseguito il commit dell'unità di lavoro (e l'unità di lavoro potrebbe contenere molte operazioni della coda).

I messaggi non persistenti possono essere utilizzati per la messaggistica rapida. Consultare Sicurezza dei messaggi per ulteriori informazioni sui messaggi rapidi.

Nota: Una combinazione di scrittura di messaggi persistenti all'interno di un'unità di lavoro e scrittura di messaggi persistenti all'esterno di un'unità o di un lavoro, può causare problemi di prestazioni potenzialmente gravi per le applicazioni. Ciò è particolarmente vero quando la stessa coda di destinazione viene utilizzata per entrambe le operazioni.

Messaggi che non vengono recapitati

Quando un gestore code non è in grado di inserire un messaggio su una coda, esistono diverse opzioni.

È possibile:

- Tentare nuovamente di inserire il messaggio nella coda.
- Richiedere che il messaggio venga restituito al mittente.
- Inserire il messaggio nella coda di messaggi non recapitabili.

Per ulteriori informazioni, fare riferimento a “Gestione degli errori del programma” a pagina 550.

Messaggi di cui è stato eseguito il backout

Quando si elaborano i messaggi da una coda sotto il controllo di un'unità di lavoro, l'unità di lavoro può essere composta da uno o più messaggi. Se si verifica un backout, i messaggi che sono stati richiamati dalla coda vengono reintegrati nella coda e possono essere elaborati di nuovo in un'altra unità di lavoro. Se l'elaborazione di un particolare messaggio sta causando il problema, viene eseguito nuovamente il backout dell'unità di lavoro. Ciò può causare un loop di elaborazione. I messaggi inseriti in una coda vengono rimossi dalla coda.

Un'applicazione può rilevare i messaggi rilevati in tale loop verificando il campo *BackoutCount* di MQMD. L'applicazione può correggere la situazione o inviare un'avvertenza a un operatore.

Su WebSphere MQ per WebSphere MQ per Windows, WebSphere MQ su sistemi UNIX , WebSphere MQ su sistemi Linux il conteggio di backout sopravvive sempre ai riavvii del gestore code. Qualsiasi modifica all'attributo *HardenGetBackout* viene ignorata.

Per ulteriori informazioni sul commit e il backout dei messaggi, consultare “Commit e backout delle unità di lavoro” a pagina 325.

Coda di risposta e gestore code

Vi sono occasioni in cui è possibile ricevere messaggi in risposta a un messaggio inviato:

- Un messaggio di risposta in risposta ad un messaggio di richiesta
- Un messaggio di report su un evento o una scadenza imprevisti

- Un messaggio di report su un evento COA (Conferma di arrivo) o COD (Conferma di consegna)
- Un messaggio di report su un evento PAN (Positive Action Notification) o NAN (Negative Action Notification)

Utilizzando la struttura MQMD, specificare il nome della coda a cui si desidera inviare i messaggi di risposta e di report nel campo *ReplyToQ* . Specificare il nome del gestore code proprietario della coda di risposta nel campo *ReplyToQMgr* .

Se si lascia vuoto il campo *ReplyToQMgr* , il gestore code imposta il contenuto dei seguenti campi nel descrittore del messaggio sulla coda:

ReplyToQ
Se *ReplyToQ* è una definizione locale di una coda remota, il campo *ReplyToQ* è impostato sul nome della coda remota; altrimenti, questo campo non viene modificato.

ReplyToQMgr
Se *ReplyToQ* è una definizione locale di una coda remota, il campo *ReplyToQMgr* è impostato sul nome del gestore code che possiede la coda remota; altrimenti, il campo *ReplyToQMgr* è impostato sul nome del gestore code a cui è connessa l'applicazione.

Nota: È possibile richiedere che un gestore code compia più di un tentativo di recapito di un messaggio ed è possibile richiedere che il messaggio venga eliminato in caso di errore. Se il messaggio, dopo non essere stato consegnato, non deve essere eliminato, il gestore code remoto inserisce il messaggio nella relativa coda di messaggi non recapitabili (messaggio non recapitato) (consultare “Utilizzo della coda dei messaggi non recapitabili (messaggi non recapitati)” a pagina 553).

Contesto messaggio

Le informazioni relative al *contesto del messaggio* consentono all'applicazione che richiama il messaggio di individuare il creatore del messaggio.

L'applicazione di richiamo potrebbe voler:

- Verificare che l'applicazione mittente disponga del livello di autorizzazione corretto
- Eseguire alcune funzioni di contabilità in modo che possa addebitare l'applicazione di invio per qualsiasi lavoro che deve eseguire
- Conserva una traccia di verifica di tutti i messaggi che ha utilizzato

Quando si utilizza la chiamata MQPUT o MQPUT1 per inserire un messaggio in una coda, è possibile specificare che il gestore code deve aggiungere alcune informazioni di contesto predefinite al descrittore del messaggio. Le applicazioni che dispongono del livello di autorizzazione appropriato possono aggiungere ulteriori informazioni di contesto. Per ulteriori informazioni su come specificare le informazioni di contesto, consultare “Controllo delle informazioni di contesto” a pagina 232.

Il contesto utente viene utilizzato dal gestore code durante la generazione dei seguenti tipi di messaggi di report:

- Conferma alla consegna
- Scadenza

Quando vengono generati questi messaggi di report, il contesto utente viene controllato per l'autorizzazione + put e + passid sulla destinazione del report. Se il contesto utente non dispone di autorizzazione sufficiente, il messaggio di report viene inserito nella coda di messaggi non recapitabili, se ne è stato definito uno. Se non è presente una coda di messaggi non instradabili, il messaggio di report viene eliminato.

Tutte le informazioni di contesto vengono memorizzate nei campi di contesto del descrittore del messaggio. Il tipo di informazioni ricade nelle informazioni di identità, origine e contesto utente.

contesto di identità

Le informazioni *Contesto identità* identificano l'utente dell'applicazione che per primo ha inserito il messaggio su una coda. Le applicazioni opportunamente autorizzate possono impostare i seguenti campi:

- Il gestore code riempie il campo *UserIdentifier* con un nome che identifica l'utente; il modo in cui il gestore code può eseguire questa operazione dipende dall'ambiente in cui è in esecuzione l'applicazione.
- Il gestore code riempie il campo *AccountingToken* con un token o un numero determinato dall'applicazione che ha inserito il messaggio.
- Le applicazioni possono utilizzare il campo *ApplIdentityData* per qualsiasi informazione aggiuntiva che desiderano includere sull'utente (ad esempio, una password codificata).

Un SID (systems security identifier) Windows viene memorizzato nel campo *AccountingToken* quando viene creato un messaggio in WebSphere MQ per Windows. Il SID può essere utilizzato per integrare il campo *UserIdentifier* e stabilire le credenziali di un utente.

Per informazioni su come il gestore code compila i campi *UserIdentifier* e *AccountingToken* , consultare le descrizioni di questi campi in UserIdentifier e AccountingToken.

Le applicazioni che inoltrano i messaggi da un gestore code a un altro devono anche trasmettere le informazioni di contesto dell'identità in modo che altre applicazioni conoscano l'identità del mittente del messaggio.

Contesto di origine

Le informazioni *Contesto origine* descrivono l'applicazione che inserisce il messaggio nella coda in cui il messaggio è *attualmente* memorizzato. Il descrittore del messaggio contiene i seguenti campi per le informazioni sul contesto di origine:

PutApplType	Il tipo di applicazione che inserisce il messaggio (ad esempio, una transazione CICS).
PutApplName	Il nome dell'applicazione che inserisce il messaggio (ad esempio, il nome di un lavoro o di una transazione).
PutDate	La data in cui il messaggio è stato inserito sulla coda.
PutTime	L'ora in cui il messaggio è stato inserito sulla coda.
ApplOriginData	Qualsiasi informazione aggiuntiva che un'applicazione desidera includere sull'origine del messaggio. Ad esempio, potrebbe essere impostato da applicazioni autorizzate in modo appropriato per indicare se i dati di identità sono attendibili.

Le informazioni sul contesto di origine vengono generalmente fornite dal gestore code. GMT (Greenwich Mean Time) viene utilizzato per i campi *PutDate* e *PutTime* . Consultare le descrizioni di questi campi in PutDate e PutTime.

Un'applicazione con autorizzazione sufficiente può fornire il proprio contesto. Ciò consente di conservare le informazioni di account quando un singolo utente ha un ID utente differente su ciascuno dei sistemi che elaborano un messaggio originato.

Oggetti WebSphere MQ

Queste informazioni forniscono dettagli sugli oggetti WebSphere MQ che includono: gestori code, gruppi di condivisione code, code, oggetti argomento di gestione, elenchi nomi, definizioni di processo, oggetti informazioni di autenticazione, canali, classi di memorizzazione, listener e servizi.

I gestori code definiscono le proprietà (note come attributi) di questi oggetti. I valori di questi attributi influiscono sul modo in cui WebSphere MQ elabora questi oggetti. Dalle proprie applicazioni, si utilizza MQI (Message Queue Interface) per controllare questi oggetti. Gli oggetti vengono identificati da un *descrittore oggetto* (MQOD) quando vengono indirizzati da un programma.

Quando si utilizzano i comandi WebSphere MQ per definire, modificare o eliminare oggetti, ad esempio, il gestore code verifica di disporre del livello di autorizzazione richiesto per eseguire queste operazioni. Allo stesso modo, quando un'applicazione utilizza la chiamata MQOPEN per aprire un oggetto, il gestore code controlla che l'applicazione disponga del livello di autorizzazione richiesto prima di consentire l'accesso a tale oggetto. Le verifiche vengono effettuate sul nome dell'oggetto da aprire.

Concetti correlati

“Controllo delle informazioni di contesto” a pagina 232
Quando si utilizza la chiamata MQPUT o MQPUT1 per inserire un messaggio in una coda, è possibile specificare che il gestore code deve aggiungere alcune informazioni di contesto predefinite al descrittore del messaggio. Le applicazioni che dispongono del livello di autorizzazione appropriato possono aggiungere ulteriori informazioni di contesto. È possibile utilizzare il campo opzioni nella struttura MQPMO per controllare le informazioni di contesto.

Riferimenti correlati

“Opzioni MQOPEN relative al contesto del messaggio” a pagina 223
Se si desidera poter associare le informazioni di contesto a un messaggio quando le si inserisce in una coda, è necessario utilizzare una delle opzioni di contesto del messaggio quando si apre la coda.

Preparazione ed esecuzione di applicazioni Microsoft Transaction Server

Per preparare un'applicazione MTS da eseguire come applicazione client WebSphere MQ MQI, seguire queste istruzioni in base al proprio ambiente.

Per informazioni generali su come sviluppare applicazioni Microsoft Transaction Server (MTS) che accedono alle risorse WebSphere MQ , consultare la sezione relativa a MTS nel Centro assistenza WebSphere MQ .

Per preparare un'applicazione MTS da eseguire come applicazione client MQI WebSphere MQ , effettuare una delle seguenti operazioni per ogni componente dell'applicazione:

- Se il componente utilizza i bind in linguaggio C per MQI, seguire le istruzioni in “Preparazione di programmi C in Windows” a pagina 461 ma collegare il componente alla libreria mqicxa.lib invece di mqic.lib.
- Se il componente utilizza le classi C++ di WebSphere MQ , seguire le istruzioni riportate in “Creazione di programmi C+ + su Windows” a pagina 654 ma collegare il componente alla libreria imqx23vn.lib invece di imqc23vn.lib.
- Se il componente utilizza i bind del linguaggio Visual Basic per MQI, seguire le istruzioni in “Preparazione dei programmi Visual Basic in Windows” a pagina 465 , ma quando si definisce il progetto Visual Basic, immettere `MqType=3` nel campo **Argomenti di compilazione condizionale** .
- Se il componente utilizza WebSphere MQ Automation Classes for ActiveX (MQAX), definire una variabile di ambiente, GMQ_MQ_LIB, con il valore `mqic32xa.dll` .

  È possibile definire la variabile di ambiente dall'interno dell'applicazione oppure è possibile definirla in modo che il suo ambito sia esteso a tutto il sistema. Tuttavia, definendolo a livello di sistema, è possibile che qualsiasi applicazione MQAX esistente, che non definisce la variabile di ambiente dall'interno dell'applicazione, si comporti in modo non corretto.

Utilizzo di IBM WebSphere MQ con WebSphere Application Server

Utilizzare questo argomento per comprendere l'utilizzo di IBM WebSphere MQ con WebSphere Application Server.

Le applicazioni scritte in Java in esecuzione in WebSphere Application Server possono utilizzare la specifica JMS (Java Messaging Service) per eseguire la messaggistica. La messaggistica point - to - point in questo ambiente può essere fornita da un gestore code IBM WebSphere MQ

Un vantaggio dell'utilizzo di un gestore code IBM WebSphere MQ per fornire la messaggistica point - to - point è che la connessione delle applicazioni JMS può partecipare pienamente alla funzionalità di una rete IBM WebSphere MQ , che consente alle applicazioni di scambiare messaggi con gestori code in esecuzione su una moltitudine di piattaforme.

Le applicazioni possono utilizzare il *trasporto client* o il *trasporto bind* per l'oggetto factory di connessione code. Per il *trasporto bind* il gestore code deve esistere localmente all'applicazione che richiede una connessione. Se il gestore code non è locale per l'applicazione, è necessario installare l' *Allegato client* per consentire all'applicazione di connettersi a un gestore code in esecuzione su un'altra macchina o immagine.

Per impostazione predefinita, i messaggi JMS conservati nelle code IBM WebSphere MQ utilizzano un'intestazione MQRFH2 per contenere alcune delle informazioni di intestazione del messaggio JMS. Molte applicazioni IBM WebSphere MQ legacy non possono elaborare i messaggi con queste intestazioni e richiedono le proprie intestazioni caratteristiche, ad esempio MQCIH per le applicazioni CICS Bridge o MQWIH per IBM WebSphere MQ Workflow. Per ulteriori dettagli su queste considerazioni speciali, consultare “Associazione dei messaggi JMS sui messaggi WebSphere MQ” a pagina 812.

Scenari di supporto transazionale

Utilizzando il supporto transazionale, è possibile consentire alle proprie applicazioni di lavorare in modo affidabile con database.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Questa sezione introduce il supporto transazionale. Il lavoro richiesto per abilitare le proprie applicazioni all'utilizzo di IBM WebSphere MQ con un prodotto database abbraccia le aree di programmazione delle applicazioni e di gestione del sistema. Utilizzare le informazioni qui insieme a “Commit e backout delle unità di lavoro” a pagina 325.

Iniziamo introducendo le unità di lavoro che formano le transazioni, quindi descriviamo i modi in cui si abilita IBM WebSphere MQ a coordinare le transazioni con i database.

Concetti correlati

“Introduzione delle unità di lavoro” a pagina 42
Questo argomento introduce e definisce i concetti generali di unità di lavoro, commit, backout e punto di sincronizzazione. Contiene inoltre due scenari che illustrano le unità di lavoro globali.

IBM WebSphere MQ e HP NonStop TMF

Introduzione delle unità di lavoro

Questo argomento introduce e definisce i concetti generali di unità di lavoro, commit, backout e punto di sincronizzazione. Contiene inoltre due scenari che illustrano le unità di lavoro globali.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Quando un programma inserisce i messaggi nelle code all'interno di un'unità di lavoro, tali messaggi vengono resi visibili ad altri programmi solo quando il programma *esegue il commit* dell'unità di lavoro. Per eseguire il commit di un'unità di lavoro, tutti gli aggiornamenti devono essere eseguiti correttamente per preservare l'integrità dei dati.

Se il programma rileva un errore e decide di non rendere l'operazione di inserimento permanente, può *eseguire il backout* dell'unità di lavoro. Quando un programma esegue un backout, WebSphere MQ ripristina le code rimuovendo i messaggi inseriti nelle code da tale unità di lavoro.

Allo stesso modo, quando un programma riceve messaggi da una o più code all'interno di un'unità di lavoro, tali messaggi rimangono nelle code fino a quando il programma non esegue il commit dell'unità di lavoro, ma i messaggi non sono disponibili per essere richiamati da altri programmi. I messaggi vengono eliminati permanentemente dalle code quando il programma esegue il commit dell'unità di lavoro. Se il programma esegue il backout dell'unità di lavoro, WebSphere MQ ripristina le code rendendo i messaggi disponibili per essere richiamati da altri programmi.

La decisione di eseguire il commit o il backout delle modifiche viene presa, nel caso più semplice, alla fine di un'attività. Tuttavia, può essere più utile per un'applicazione sincronizzare le modifiche dei dati in altri punti logici all'interno di un'attività. Questi punti logici sono denominati punti di sincronizzazione (o punti di sincronizzazione) e il periodo di elaborazione di una serie di aggiornamenti tra due punti di sincronizzazione è denominato *unità di lavoro*. Diverse chiamate MQGET e MQPUT possono essere parte di una singola unità di lavoro.

Con WebSphere MQ, è necessario distinguere tra unità di lavoro *locali* e *globali* :

Unità di lavoro locali
Si tratta di quelle in cui le uniche azioni vengono immesse e richiamate dalle code WebSphere MQ e il coordinamento di ogni unità di lavoro viene fornito nel gestore code utilizzando un processo di *commit a fase singola* .

Utilizzare le unità di lavoro locali quando le uniche risorse da aggiornare sono le code gestite da un unico gestore code WebSphere MQ . Viene eseguito il commit degli aggiornamenti utilizzando il verbo MQCMIT o il backout utilizzando MQBACK.

Non sono presenti attività di gestione del sistema, diverse dalla gestione log, coinvolte nell'utilizzo delle unità di lavoro locali. Nelle applicazioni, in cui si utilizzano le chiamate MQPUT e MQGET con MQCMIT e MQBACK, provare utilizzando le opzioni MQPMO_SYNCPOINT e MQGMO_SYNCPOINT. (Per informazioni sulla gestione dei log, consultare Gestione dei file di log .)

Unità di lavoro globali
Sono quelle in cui vengono aggiornate anche altre risorse, come le tabelle in un database relazionale. Quando è coinvolto più di un *gestore risorse* , è necessario il software *gestore transazioni* che utilizza un processo di *commit a due fasi* per coordinare l'unità di lavoro globale.

Utilizzare le unità di lavoro globali quando è necessario includere anche aggiornamenti al software del gestore database relazionali, come Db2, Oracle, Sybasee Informix.

Esistono diversi scenari possibili per l'utilizzo delle unità di lavoro globali. Di seguito sono riportati due scenari:

1. Nel primo, il gestore code stesso funge da gestore transazioni. In questo scenario, i verbi MQI controllano le unità di lavoro globali; vengono avviati nelle applicazioni che utilizzano il verbo MQBEGIN e quindi ne viene eseguito il commit utilizzando MQCMIT o ne viene eseguito il backout utilizzando MQBACK.
2. Nel secondo, il ruolo di gestore transazioni viene eseguito da altro software, ad esempio TXSeries, Encinao Tuxedo. In questo scenario, un'API fornita dal software del gestore transazioni viene utilizzata per controllare l'unità di lavoro (ad esempio, EXEC CICS SYNCPOINT for TXSeries).

Le seguenti sezioni descrivono tutte le fasi necessarie per utilizzare le unità di lavoro globali, organizzate in base ai due scenari:

- “Scenario 1: il gestore code esegue il coordinamento” a pagina 43
- “Scenario 2: Altri software forniscono il coordinamento” a pagina 70

Scenario 1: il gestore code esegue il coordinamento

Nello scenario 1, il gestore code agisce come gestore transazioni. In questo scenario, i verbi MQI controllano le unità di lavoro globali; vengono avviati nelle applicazioni che utilizzano il verbo MQBEGIN e quindi ne viene eseguito il commit utilizzando MQCMIT o ne viene eseguito il backout utilizzando MQBACK.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Livello di isolamento

In IBM WebSphere MQ, un messaggio su una coda potrebbe essere visibile prima di un aggiornamento del database, in base alla progettazione di isolamento della transazione implementata nel database.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Quando un gestore code IBM WebSphere MQ funziona come un gestore transazioni XA, per coordinare gli aggiornamenti ai gestori risorse XA, viene seguito il seguente protocollo di commit:

1. Preparare tutti i gestori risorse XA.
2. Eseguire il commit del gestore risorse del gestore code IBM WebSphere MQ .
3. Eseguire il commit di altri gestori risorse.

Tra i passi 2 e 3, un'applicazione potrebbe visualizzare un messaggio di cui è stato eseguito il commit nella coda, ma la riga corrispondente nel database non riflette questo messaggio.

Non si tratta di un problema se il database è configurato in modo che le chiamate dell'API del database dell'applicazione siano in attesa del completamento degli aggiornamenti in sospeso.

È possibile risolvere questo problema configurando il database in maniera diversa. Il tipo di configurazione necessaria viene definito "livello di isolamento". Per ulteriori informazioni sui livelli di isolamento, fare riferimento alla documentazione del database. In alternativa, è possibile configurare il gestore code per eseguire il commit dei gestori risorse nel seguente ordine inverso:

1. Preparare tutti i gestori risorse XA.
2. Eseguire il commit di altri gestori risorse.
3. Eseguire il commit del gestore risorse del gestore code IBM WebSphere MQ .

Quando si modifica il protocollo, viene eseguito l'ultimo commit del gestore code IBM WebSphere MQ , in modo che le applicazioni che leggono i messaggi dalle code visualizzino un messaggio solo dopo il completamento dell'aggiornamento del database corrispondente.

Per configurare il gestore code per utilizzare questo protocollo modificato, impostare la variabile di ambiente **AMQ_REVERSE_COMMIT_ORDER** .

Impostare questa variabile di ambiente nell'ambiente da cui viene eseguito il **strmqm** per avviare il gestore code. Ad esempio, eseguire quanto segue nella shell appena prima di avviare il gestore code:

```
export AMQ_REVERSE_COMMIT_ORDER=1
```

Nota: L'impostazione di questa variabile di ambiente potrebbe causare una voce di log supplementare per transazione, quindi ciò avrà un piccolo impatto sulle prestazioni di ciascuna transazione.

Coordinamento database

Quando il gestore code coordina le unità di lavoro globali, diventa possibile integrare gli aggiornamenti del database all'interno delle unità di lavoro. Ossia, è possibile scrivere un'applicazione MQI e SQL mista e utilizzare i verbi MQCMIT e MQBACK per eseguire il commit o il rollback delle modifiche alle code e ai database insieme.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Il gestore code ottiene questo risultato utilizzando il protocollo di commit a due fasi descritto in *X/Open Distributed Transaction Processing: The XA Specification*. Quando deve essere eseguito il commit di un'unità di lavoro, il gestore code chiede prima a ciascun gestore database partecipante se è pronto a eseguire il commit dei propri aggiornamenti. Solo se tutti i partecipanti, incluso il gestore code stesso, sono pronti per il commit, viene eseguito il commit di tutti gli aggiornamenti della coda e del database. Se un partecipante non può preparare gli aggiornamenti, viene eseguito il backout dell'unità di lavoro.

In generale, un'unità di lavoro globale viene implementata in un'applicazione con il seguente metodo (in pseudocodice):

MQBEGIN
MQGET (includere l'indicatore MQGMO_SYNCPOINT nelle opzioni del messaggio)
MQPUT (includere l'indicatore MQPMO_SYNCPOINT nelle opzioni del messaggio)
INSERT SQL
MQCMIT

Lo scopo di MQBEGIN è di indicare l'inizio di un'unità di lavoro globale. Lo scopo di MQCMIT è di indicare la fine dell'unità di lavoro globale e di completarla con tutti i gestori risorse partecipanti, utilizzando il protocollo di commit a due fasi.

Quando l'unità di lavoro (nota anche come *transazione*) viene completata correttamente utilizzando MQCMIT, tutte le azioni intraprese all'interno di tale unità di lavoro vengono rese permanenti o irreversibili. Se, per qualsiasi motivo, l'unità di lavoro ha esito negativo, tutte le azioni vengono invece ripristinate. Non è possibile rendere permanente un'azione in un'unità di lavoro, mentre ne viene ripristinata un'altra. Questo è il principio di un'unità di lavoro: o tutte le azioni all'interno dell'unità di lavoro sono rese permanenti o nessuna di esse lo è.

Nota:

1. Il programmatore dell'applicazione può forzare il ripristino di un'unità di lavoro richiamando MQBACK. L'unità di lavoro viene ripristinata dal gestore code anche se l'applicazione o il database *ha esito negativo* prima di richiamare MQCMIT.
2. Se un'applicazione richiama MQDISC senza richiamare MQCM, il gestore code si comporta come se MQCMIT fosse stato richiamato ed esegue il commit dell'unità di lavoro.

Tra MQBEGIN e MQCMIT, il gestore code non effettua alcuna chiamata al database per aggiornare le sue risorse. In altre parole, l'unico modo in cui le tabelle di un database vengono modificate è mediante il codice (ad esempio, SQL INSERT nello pseudocodice).

Il supporto di ripristino completo viene fornito se il gestore code perde il contatto con uno dei gestori database durante il protocollo di commit. Se un gestore database diventa non disponibile mentre è in dubbio, ovvero è stato preparato correttamente per il commit, ma deve ancora ricevere una decisione di commit o di backout, il gestore code ricorda il risultato dell'unità di lavoro fino a quando tale risultato non è stato consegnato correttamente al database. Allo stesso modo, se il gestore code termina con operazioni di commit incomplete in sospeso, tali operazioni vengono ricordate al riavvio del gestore code. Se un'applicazione viene terminata in modo imprevisto, l'integrità dell'unità di lavoro non viene compromessa, ma il risultato dipende da dove nel processo l'applicazione è terminata, come descritto in Tabella 5 a pagina 46.

Ciò che accade quando il database o il programma applicativo ha esito negativo viene riepilogato nelle seguenti tabelle:

Tabella 4. Cosa accade quando un server di database ha esito negativo

Ricorrenza di errore	**Risultato**
Prima della chiamata dell'applicazione a MQCMIT.	L'unità di lavoro viene ripristinata.
Durante la chiamata dell'applicazione a MQCMIT, **prima** tutti i database hanno indicato che sono stati preparati correttamente.	Viene eseguito il backout dell'unità di lavoro con un codice motivo MQRC_BACKED_OUT.
Durante la chiamata dell'applicazione a MQCMIT, **dopo** tutti i database hanno indicato di aver preparato correttamente, ma prima di tutti hanno indicato di aver eseguito correttamente il commit.	L'unità di lavoro è conservata in uno stato ripristinabile dal gestore code, con un codice motivo MQRC_OUTCOME_PENDING.
Durante la chiamata dell'applicazione a MQCMIT, **dopo** che tutti i database hanno indicato di aver eseguito correttamente il commit.	L'unità di lavoro viene sottoposta a commit con un codice motivo MQRC_NONE.
Dopo la chiamata dell'applicazione a MQCMIT.	L'unità di lavoro viene sottoposta a commit con un codice motivo MQRC_NONE.

Tabella 5. Cosa succede quando un programma di applicazione ha esito negativo	
Ricorrenza di errore	**Risultato**
Prima della chiamata dell'applicazione a MQCMIT.	L'unità di lavoro viene ripristinata.
Durante la chiamata dell'applicazione a MQCMIT, **prima** che il gestore code abbia ricevuto la richiesta MQCMIT dell'applicazione.	L'unità di lavoro viene ripristinata.
Durante la chiamata dell'applicazione a MQCMIT, **dopo** che il gestore code ha ricevuto la richiesta MQCMIT dell'applicazione.	Il gestore code tenta di eseguire il commit utilizzando il commit a due fasi (in base al fatto che i prodotti database eseguano e eseguano correttamente il commit delle loro parti dell'unità di lavoro).

Nel caso in cui il codice motivo restituito da MQCMIT sia MQRC_OUTCOME_PENDING, l'unità di lavoro viene ricordata dal gestore code fino a quando non è stato in grado di ristabilire il contatto con il server di database e di eseguire il commit della sua parte dell'unità di lavoro. Fare riferimento a “Considerazioni quando si perde il contatto con il gestore risorse XA” a pagina 62 per informazioni su come e quando eseguire il recupero.

Il gestore code comunica con i gestori database utilizzando l'interfaccia XA come descritto in *X/Open Distributed Transaction Processing: The XA Specification*. Esempi di queste chiamate di funzione sono xa_open, xa_start, xa_end, xa_prepare e xa_commit. Vengono utilizzati i termini *gestore transazioni* e *gestore risorse* nello stesso modo in cui vengono utilizzati nella specifica XA.

Limitazioni
Esistono delle limitazioni al supporto di coordinamento del database.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Sono applicate le seguenti limitazioni:

- La capacità di coordinare gli aggiornamenti del database all'interno delle unità di lavoro WebSphere MQ è **non** supportata in un'applicazione client MQI. L'utilizzo di MQBEGIN in un'applicazione client non riesce. Un programma che richiama MQBEGIN deve essere eseguito come applicazione *server* sulla stessa macchina del gestore code.

  Nota: Un'applicazione *server* è un programma che è stato collegato con le librerie server WebSphere MQ necessarie; un'applicazione *client* è un programma che è stato collegato con le librerie client WebSphere MQ necessarie. Consultare “Creazione di applicazioni per client MQI WebSphere MQ” a pagina 360 e “Creazione di un'applicazione IBM WebSphere MQ” a pagina 431 per dettagli sulla compilazione e il collegamento dei programmi.
- Il server di database può risiedere su una macchina diversa dal server del gestore code, purché il client del database sia installato sulla stessa macchina del gestore code e supporti questa funzione. Consultare la documentazione del prodotto database per determinare se il software client può essere utilizzato per i sistemi di commit a due fasi.
- Sebbene il gestore code si comporti come un gestore risorse (per essere coinvolto nelle unità di lavoro globali dello Scenario 2), non è possibile fare in modo che un gestore code coordini un altro gestore code all'interno delle relative unità di lavoro globali dello Scenario 1.

File di caricamento switch

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Il file di caricamento switch è una libreria condivisa (una DLL su sistemi Windows) caricata dal codice nell'applicazione IBM WebSphere MQ e dal gestore code. Il suo scopo è semplificare il caricamento della libreria condivisa del client del database e restituire i puntatori alle funzioni XA.

I dettagli del file di caricamento switch devono essere specificati prima dell'avvio del gestore code. I dettagli vengono inseriti nel file qm.ini su sistemi Windows, UNIX and Linux .

- Su sistemi Windows e Linux (x86 e x86-64), utilizzare IBM WebSphere MQ Explorer per aggiornare il file qm.ini .
- Su tutti gli altri sistemi modificare il file, qm.ini, direttamente.

L'origine C per il file di caricamento switch viene fornita con l'installazione di IBM WebSphere MQ se supporta le unità di lavoro globali Scenario 1. L'origine contiene una funzione denominata MQStart. Quando il file di caricamento switch viene caricato, il gestore code richiama questa funzione, che restituisce l'indirizzo di una struttura denominata *switch XA*.

La struttura di commutazione XA esiste nella libreria condivisa del client del database e contiene una serie di puntatori di funzione, come descritto in Tabella 6 a pagina 47:

Tabella 6. Puntatori funzione switch XA

Nome puntatore funzione	**funzione XA**	**Finalità**
xa_open_entry	xa_open	Connetti al database
xa_close_entry	xa_close	Disconnetti dal database
xa_start_entry	xa_start	Avviare un ramo di un'unità di lavoro globale
xa_end_entry	xa_end	Sospende un ramo di un'unità di lavoro globale
xa_rollback_entry	xa_rollback	Eseguire il rollback di un ramo di un'unità di lavoro globale
xa_prepare_entry	xa_prepare	Preparazione al commit di un ramo di un'unità di lavoro globale
xa_commit_entry	xa_commit	Eseguire il commit di un ramo di un'unità di lavoro globale
xa_recover_entry	xa_recover	Scopri dal database se ha un'unità di lavoro in dubbio
xa_forget_entry	xa_forget	Consente a un database di dimenticare un ramo di un'unità di lavoro globale
xa_complete_entry	xa_complete	Completare un ramo di un'unità di lavoro globale

Durante la prima chiamata MQBEGIN nell'applicazione, il codice IBM WebSphere MQ che viene eseguito come parte di MQBEGIN carica il file di caricamento switch e richiama la funzione xa_open nella libreria condivisa del database. Allo stesso modo, durante l'avvio del gestore code e in altre successive occasioni, alcuni processi del gestore code caricano il file di caricamento switch e richiamano xa_open.

È possibile ridurre il numero di chiamate xa_ * utilizzando la *registrazione dinamica*. Per una descrizione completa di questa tecnica di ottimizzazione, consultare “Registrazione dinamica XA” a pagina 67.

Configurazione del sistema per il coordinamento del database

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per

andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

È necessario eseguire diverse attività prima che un gestore database possa partecipare a unità globali di lavori coordinate dal gestore code. Questi sono descritti qui di seguito:

- “Installazione e configurazione del database” a pagina 48
- “Creazione di file di caricamento switch” a pagina 48
- “Aggiunta di informazioni di configurazione al gestore code” a pagina 49
- “Scrittura e modifica delle applicazioni” a pagina 51
- “Verifica del sistema” a pagina 51

Installazione e configurazione del database
Per installare e configurare il prodotto database, consultare la documentazione del prodotto. Questi argomenti in questa sezione descrivono problemi di configurazione generali e come sono correlati all'interoperabilità tra WebSphere MQ ed il database.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Connessioni al database

Un'applicazione che stabilisce una connessione standard al gestore code è associata a un thread in un processo dell'agent del gestore code locale separato. Una connessione che non è una connessione *fastpath* è una connessione *standard* in questo contesto. Per ulteriori informazioni, consultare “Connessione a un gestore code utilizzando la chiamata MQCONNX” a pagina 209.)

Quando l'applicazione emette `MQBEGIN`, sia esso il processo dell'agent richiamano la funzione xa_open nella libreria del client del database. In risposta a ciò, il codice della libreria client database *connette* al database che deve essere coinvolto nell'unità di lavoro *sia dall'applicazione che dai processi del gestore code*. Queste connessioni al database vengono mantenute finché l'applicazione rimane connessa al gestore code.

Si tratta di una considerazione importante se il database supporta solo un numero limitato di utenti o connessioni, poiché vengono effettuate due connessioni al database per supportare l'unico programma applicativo.

Configurazione client/server

La libreria client del database caricata in WebSphere MQ gestore code e processi applicazione **deve** essere in grado di inviare e ricevere dal server. Accertarsi che:

- I file di configurazione client/server del database hanno i dettagli corretti
- Le variabili di ambiente pertinenti sono impostate nell'ambiente del gestore code **e** i processi dell'applicazione

Creazione di file di caricamento switch
WebSphere MQ viene fornito con un makefile di esempio, utilizzato per creare file di caricamento switch per i gestori database supportati.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Il makefile di esempio, insieme a tutti i file di origine C associati richiesti per creare i file di caricamento switch, viene installato nelle seguenti directory:

- Per WebSphere MQ per Windows, nella directory *MQ_INSTALLATION_PATH*\tools\c\samples\xatm\
- Per WebSphere MQ per sistemi UNIX and Linux , nella directory *MQ_INSTALLATION_PATH*/samp/xatm/

I moduli di origine di esempio utilizzati per creare i file di caricamento switch sono:

- Per DB2, db2swit.c
- Per Oracle, oraswit.c
- Per Informix, infswit.c
- Per Sybase, sybswit.c

Quando si generano i file di caricamento switch, installare i file di caricamento switch a 32 - bit in /var/mqm/exits e installare i file di caricamento switch a 64 - bit in /var/mqm/exits64.

Se si dispone di gestori code a 32 bit, il file make di esempio, xaswit.mak, installa un file di caricamento switch a 32 bit in /var/mqm/exits.

Se si dispone di gestori code a 64 bit, il file make di esempio, xaswit.mak, installa un file di caricamento switch a 32 bit in /var/mqm/exitse un file di caricamento switch a 64 bit in /var/mqm/exits64.

Sicurezza file

È possibile che il sistema operativo non riesca a caricare il file di caricamento switch da WebSphere MQ, per motivi esterni al controllo di WebSphere MQ. In questo caso, i messaggi di errore vengono scritti nei log di errori di WebSphere MQ e, potenzialmente, la chiamata MQBEGIN potrebbe non riuscire. Per garantire che il sistema operativo non abbia esito negativo nel caricamento del file di caricamento switch, è necessario soddisfare i seguenti requisiti:

1. Il file di caricamento switch deve essere disponibile nell'ubicazione fornita nel file qm.ini .
2. Il file di caricamento switch deve essere accessibile a tutti i processi che devono caricarlo, inclusi i processi del gestore code e i processi dell'applicazione.
3. Tutte le librerie da cui dipende il file di caricamento switch, incluse le librerie fornite dal database, devono essere presenti e accessibili.

Aggiunta di informazioni di configurazione al gestore code
Una volta creato un file di caricamento switch per il gestore database e collocato in un percorso sicuro, è necessario specificare tale ubicazione per il gestore code.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Per specificare l'ubicazione, effettuare quanto segue:

- Su sistemi Windows e Linux (piattaformex86 e x86-64) utilizzare WebSphere MQ Explorer. Specificare i dettagli del file di caricamento switch nel pannello delle proprietà del gestore code, nel gestore risorse XA.
- Su tutti gli altri sistemi specificare i dettagli del file di caricamento switch nella stanza XAResourceManager nel file qm.ini del gestore code.

Aggiungere una sezione XAResourceManager per il database che verrà coordinato dal gestore code. Il caso più comune è che ci sia solo un database e quindi solo una stanza XAResourceManager . Per i dettagli delle configurazioni più complicate che coinvolgono più database, consultare “Configurazioni di più database” a pagina 61. Gli attributi della stanza XAResourceManager sono i seguenti:

Nome=nome
Stringa scelta dall'utente che identifica il gestore risorse. In effetti, fornisce un nome alla stanza XAResourceManager . Il nome è obbligatorio e può avere una lunghezza massima di 31 caratteri.

Il nome scelto deve essere univoco; deve esistere solo una stanza XAResourceManager con questo nome in questo file qm.ini . Il nome deve essere significativo anche perché il gestore code lo utilizza per fare riferimento a questo gestore risorse sia nei messaggi di log degli errori del gestore code che nell'output quando viene utilizzato il comando `dspmqtrn` . (Per ulteriori informazioni, consultare “Visualizzazione delle unità di lavoro in sospeso con il comando dspmqtrn” a pagina 63 .)

Una volta scelto un nome e avviato il gestore code, non modificare l'attributo Nome. Per ulteriori dettagli sulla modifica delle informazioni di configurazione, consultare “Modifica delle informazioni di configurazione” a pagina 66.

SwitchFile= nome
Questo è il nome del file di caricamento switch XA creato in precedenza. Questo è un attributo obbligatorio. Il codice nel gestore code e nei processi dell'applicazione WebSphere MQ tenta di caricare il file di caricamento switch in due occasioni:

1. All'avvio del gestore code
2. Quando si effettua la prima chiamata a MQBEGIN nel processo dell'applicazione WebSphere MQ

Gli attributi di sicurezza e di autorizzazioni del file di caricamento switch devono consentire a questi processi di eseguire questa azione.

XAOpenString= stringa
Questa è una stringa di dati che il codice WebSphere MQ invia alle chiamate alla funzione xa_open del gestore database. Questo è un attributo facoltativo; se viene omesso, viene utilizzata una stringa di lunghezza zero.

Il codice nel gestore code e nei processi di applicazione WebSphere MQ richiama la funzione xa_open in due occasioni:

1. All'avvio del gestore code
2. Quando si effettua la prima chiamata a MQBEGIN nel processo dell'applicazione WebSphere MQ

Il formato per questa stringa è particolare per ogni prodotto database e verrà descritto nella relativa documentazione. In generale, la stringa xa_open contiene le informazioni di autenticazione (nome utente e password) per consentire una connessione al database sia nel gestore code che nei processi dell'applicazione.

XACloseString= stringa
Questa è una stringa di dati che il codice WebSphere MQ invia alle chiamate alla funzione xa_close del gestore database. Questo è un attributo facoltativo; se viene omesso, viene utilizzata una stringa di lunghezza zero.

Il codice nel gestore code e nei processi dell'applicazione WebSphere MQ richiama la funzione xa_close in due occasioni:

1. All'avvio del gestore code
2. Quando si effettua una chiamata a MQDISC nel proprio processo dell'applicazione WebSphere MQ , dopo aver precedentemente effettuato una chiamata a MQBEGIN

Il formato per questa stringa è particolare per ogni prodotto database e verrà descritto nella relativa documentazione. In generale, la stringa è vuota ed è comune omettere l'attributo XACloseString dalla stanza XAResourceManager .

ThreadOfControl=THREAD |PROCESS
Il valore di controllo ThreadOfpuò essere THREAD o PROCESS. Il gestore code lo utilizza per scopi di serializzazione. Questo è un attributo facoltativo; se viene omesso, viene utilizzato il valore PROCESS.

Se il codice client del database consente ai thread di richiamare le funzioni XA senza serializzazione, il valore per il controllo ThreadOfpuò essere THREAD. Il gestore code presuppone che possa richiamare le funzioni XA nella libreria condivisa del client del database da più thread contemporaneamente, se necessario.

Se il codice client del database non consente ai thread di richiamare le funzioni XA in questo modo, il valore per ThreadOfControl deve essere PROCESS. In questo caso, il gestore code serializza tutte le chiamate alla libreria condivisa del client database in modo che venga effettuata una sola chiamata

alla volta dall'interno di un determinato processo. È anche necessario assicurarsi che l'applicazione esegua una serializzazione simile se viene eseguita con più thread.

Si noti che questo problema, relativo alla capacità del prodotto database di gestire in questo modo i processi a più thread, è un problema per il fornitore del prodotto. Consultare la documentazione del prodotto database per i dettagli su come impostare l'attributo di controllo ThreadOfsu THREAD o PROCESS. Si consiglia, se possibile, di impostare il controllo ThreadOfsu THREAD. In caso di dubbio, l'opzione *safer* consiste nell'impostarla su PROCESS, anche se si perderanno i potenziali vantaggi di prestazioni dell'utilizzo di THREAD.

Scrittura e modifica delle applicazioni
Come implementare un'unità di lavoro globale.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

I programmi applicativi di esempio per le unità di lavoro globali Scenario 1 forniti con un'installazione WebSphere MQ sono descritti in “Introduzione delle unità di lavoro” a pagina 42.

In generale, un'unità di lavoro globale viene implementata in un'applicazione con il seguente metodo (in pseudocodice):

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

Lo scopo di MQBEGIN è di indicare l'inizio di un'unità di lavoro globale. Lo scopo di MQCMIT è di indicare la fine dell'unità di lavoro globale e di completarla con tutti i gestori risorse partecipanti, utilizzando il protocollo di commit a due fasi.

Tra MQBEGIN e MQCMIT, il gestore code non effettua alcuna chiamata al database per aggiornare le sue risorse. In altre parole, l'unico modo in cui le tabelle di un database vengono modificate è mediante il codice (ad esempio, SQL INSERT nello pseudocodice).

Il ruolo del gestore code, per quanto riguarda il database, è quello di indicare quando è stata avviata un'unità di lavoro globale, quando è stata terminata e se è necessario eseguire il commit o il rollback dell'unità di lavoro globale.

Per quanto riguarda l'applicazione, il gestore code esegue due ruoli: un gestore risorse (dove le risorse sono messaggi sulle code) e il gestore transazioni per l'unità di lavoro globale.

Iniziare con i programmi di esempio forniti e utilizzare le varie WebSphere MQ e le chiamate API del database effettuate in tali programmi. Le chiamate API interessate sono documentate in “Programmi di WebSphere MQ di esempio” a pagina 97, Tipi di dati utilizzati in MQIe (nel caso dell'API del database) nella documentazione del database.

Verifica del sistema
L'utente sa se l'applicazione e il sistema sono configurati correttamente solo eseguendoli durante il test. È possibile verificare la configurazione del sistema (la comunicazione corretta tra il gestore code e il database) creando ed eseguendo un programma di esempio fornito.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Configurazione di Db2
Informazioni di supporto e configurazione di DB2 .

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

I livelli supportati di Db2 sono definiti nella pagina IBM WebSphere MQ requisiti di sistema dettagliati .

Nota: Le istanze a 32 bit di Db2 non sono supportate sulle piattaforme in cui il gestore code è a 64 bit.

Eseguire quanto segue:

1. Controllare le impostazioni della variabile di ambiente.
2. Creare il file di caricamento switch Db2 .
3. Aggiungere le informazioni di configurazione del gestore risorse.
4. Modificare i parametri di configurazione di Db2 , se necessario.

Leggere queste informazioni insieme alle informazioni generali fornite in “Configurazione del sistema per il coordinamento del database” a pagina 47.

Avviso: Se si esegue `db2profile` su piattaforme UNIX and Linux , vengono impostate le variabili di ambiente LIBPATH e LD_LIBRARY_PATH. Si consiglia di `unset` queste variabili di ambiente, consultare la guida *Quick Beginnings* appropriata.

Verifica delle impostazioni della variabile d'ambiente Db2

Assicurarsi che le variabili di ambiente Db2 siano impostate per i processi del gestore code ***e in*** i processi dell'applicazione. In particolare, è necessario impostare sempre la variabile di ambiente DB2INSTANCE ***prima*** di avviare il gestore code. La variabile di ambiente DB2INSTANCE identifica l'istanza di Db2 contenente i database Db2 che vengono aggiornati. Ad esempio:

- Su sistemi UNIX and Linux , utilizzare:

```
export DB2INSTANCE=db2inst1
```

- Su sistemi Windows , utilizzare:

```
set DB2INSTANCE=DB2
```

Su Windows con un database Db2 , è necessario aggiungere l'utente MUSR_MQADMIN al gruppo DB2USERS per consentire l'avvio del gestore code.

Creazione del file di caricamento switch Db2

Il modo più semplice per creare il file di caricamento switch Db2 consiste nell'utilizzare il file di esempio xaswit.mak, fornito da WebSphere MQ per creare i file di caricamento switch per una varietà di prodotti database.

Su sistemi Windows , è possibile trovare xaswit.mak nella directory *MQ_INSTALLATION_PATH*`\tools\c\samples\xatm` *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ . Per creare il file di caricamento switch Db2 con Microsoft Visual C + +, utilizzare:

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

Il file switch generato viene inserito in `c:\Program Files\IBM\WebSphere MQ\exits`.

È possibile trovare xaswit.mak nella directory *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Modificare xaswit.mak per *rimuovere il commento* dalle righe appropriate alla versione di Db2 che si sta utilizzando. Quindi, eseguire il makefile utilizzando il comando:

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

Il file di caricamento switch a 32 bit generato viene inserito in `/var/mqm/exits`.

Il file di caricamento dello switch a 64 bit generato si trova in `/var/mqm/exits64`.

Aggiunta delle informazioni di configurazione del gestore risorse per Db2

È necessario modificare le informazioni di configurazione per il gestore code per dichiarare Db2 come partecipante nelle unità di lavoro globali. La modifica delle informazioni di configurazione in questo modo è descritta più dettagliatamente in "Aggiunta di informazioni di configurazione al gestore code" a pagina 49.

- Su sistemi Windows e Linux (piattaformex86 e x86-64), utilizzare WebSphere MQ Explorer. Specificare i dettagli del file di caricamento switch nel pannello delle proprietà del gestore code, nel gestore risorse XA.
- Su tutti gli altri sistemi specificare i dettagli del file di caricamento switch nella stanza XAResourceManager nel file qm.ini del gestore code.

Figura 9 a pagina 53 è un esempio UNIX , che mostra una voce XAResourceManager in cui il database da coordinare è denominato `mydbname`, questo nome viene specificato in `XAOpenString`:

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

Figura 9. Voce XAResourceManager di esempio per Db2 su piattaforme UNIX

Nota:

1. `ThreadOfControl=THREAD` non può essere utilizzato con Db2 versioni precedenti alla versione 8. Impostare `ThreadOfControl` e il parametro XAOpenString `toc` su una delle seguenti combinazioni:
   - `ThreadOfControl=THREAD` e `toc=t`
   - `ThreadOfControl=PROCESS` e `toc=p`

   Se si utilizza il file di caricamento switch XA jdbcdb2 per abilitare il coordinamento JDBC/JTA, è necessario utilizzare `ThreadOfControl=PROCESS` e `toc=p`.

Modifica dei parametri di configurazione di Db2

Per ciascun database Db2 coordinato dal gestore code, è necessario impostare i privilegi del database, modificare il parametro tp_mon_name e reimpostare il parametro maxappls. A tal fine, effettuare i seguenti passi:

Imposta privilegi database

I processi del gestore code vengono eseguiti con mqm utente e gruppo effettivi sui sistemi UNIX and Linux . Sui sistemi Windows , vengono eseguiti come utente che ha avviato il gestore code. Può essere uno dei seguenti:

1. L'utente che ha emesso il comando `strmqm` o
2. L'utente con cui viene eseguito il server IBM MQSeries Service COM

Per impostazione predefinita, questo utente è denominato MUSR_MQADMIN.

Se non sono stati specificati un nome utente e una password nella stringa xa_open, **l'utente con cui è in esecuzione il gestore code** viene utilizzato da Db2 per autenticare la chiamata xa_open. Se questo utente (ad esempio, l'utente mqm sui sistemi UNIX and Linux) non dispone di privilegi minimi nel database, il database rifiuta di autenticare la chiamata xa_open.

Le stesse considerazioni si applicano al processo di applicazione. Se non sono stati specificati un nome utente e una password nella stringa xa_open, l'utente con cui è in esecuzione l'applicazione viene utilizzato da Db2 per autenticare la chiamata xa_open effettuata durante il primo MQBEGIN. Ancora una volta, questo utente deve disporre di privilegi minimi nel database per funzionare.

Ad esempio, fornire all'utente mqm l'autorità di collegamento nel database mydbname emettendo i seguenti comandi Db2 :

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

Consultare “Considerazioni sulla sicurezza” a pagina 62 per ulteriori informazioni sulla sicurezza.

Windows **Modificare il parametro TP_MON_NAME**

Solo per sistemi Db2 per Windows , modificare il parametro di configurazione TP_MON_NAME per denominare la DLL utilizzata da Db2 per richiamare il gestore code per la registrazione dinamica.

Utilizzare il comando `db2 update dbm cfg using TP_MON_NAME mqmax` per denominare MQMAX.DLL come libreria utilizzata da Db2 per richiamare il gestore code. Deve essere presente in una directory all'interno di PATH.

Reimpostare il parametro maxappls

Potrebbe essere necessario rivedere l'impostazione per il parametro *maxappls* , che limita il numero massimo di applicazioni che possono essere connesse a un database. Fare riferimento a “Installazione e configurazione del database” a pagina 48.

Configurazione di Oracle

Informazioni di configurazione e supporto Oracle .

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Completare i seguenti passi:

1. Controllare le impostazioni della variabile di ambiente.
2. Creare il file di caricamento switch Oracle .
3. Aggiungere le informazioni di configurazione del gestore risorse.
4. Modificare i parametri di configurazione di Oracle , se necessario.

Un elenco corrente dei livelli di Oracle supportati da IBM WebSphere MQ viene fornito nella pagina IBM WebSphere MQ requisiti di sistema dettagliati .

Verifica delle impostazioni delle variabili di ambiente Oracle

Verificare che le variabili di ambiente Oracle siano impostate per i processi del gestore code e per i processi dell'applicazione. In particolare, impostare sempre le seguenti variabili di ambiente prima di avviare il gestore code:

ORACLE_HOME

La directory home Oracle . Ad esempio, su sistemi UNIX and Linux , utilizzare:

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

Su sistemi Windows , utilizzare:

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

ORACLE_SID
Il SID Oracle utilizzato. Se si utilizza Net8 per la connettività client/server, potrebbe non essere necessario impostare questa variabile di ambiente. Consultare la documentazione Oracle .

L'esempio successivo è un esempio di impostazione di questa variabile di ambiente su sistemi UNIX and Linux :

```
export ORACLE_SID=sid1
```

L'equivalente sui sistemi Windows è:

```
set ORACLE_SID=sid1
```

Nota: La variabile di ambiente PATH deve essere impostata in modo da includere la directory dei file binari (ad esempio, `ORACLE_INSTALL_DIR/VERSION/32BIT_NAME/bin` o `ORACLE_INSTALL_DIR/VERSION/64BIT_NAME/bin`), altrimenti potrebbe essere visualizzato un messaggio che indica che le librerie oraclient mancano dalla macchina.

Se si eseguono gestori code su sistemi Windows a 64 bit, è necessario installare sia i client Oracle a 64 bit che a 32 bit. È necessario installare entrambi i client perché il gestore code viene eseguito come processi a 32 bit che utilizzano un file di caricamento switch a 32 bit, che a sua volta deve avviare una dll client Oracle a 32 bit.

Il file di caricamento degli switch, caricato dai gestori code a 64 bit, deve accedere alle librerie client a 64 bit Oracle . I gestori code a 32 bit devono accedere al client Oracle a 32 bit quando IBM WebSphere MQ è in esecuzione su un sistema a 64 bit Windows .

Creazione del file di caricamento switch Oracle

Per creare il file di caricamento switch Oracle , utilizzare il file di esempio `xaswit.mak`, fornito da IBM WebSphere MQ per creare i file di caricamento switch per diversi prodotti database. Sui sistemi Windows , è possibile trovare `xaswit.mak` nella directory `C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm`. Per creare il file di caricamento switch Oracle con Microsoft Visual C + +, utilizzare: `nmake /f xaswit.mak oraswit.dll`

Il file switch generato viene collocato in *MQ_INSTALLATION_PATH*`\exits`.*MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato IBM WebSphere MQ .

È possibile trovare `xaswit.mak` nella directory *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato IBM WebSphere MQ .

Modificare `xaswit.mak` per eliminare il commento dalle righe appropriate alla versione di Oracle che si sta utilizzando. Quindi, eseguire il makefile utilizzando il comando:

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

Il file di caricamento dello switch a 32 bit generato si trova in `/var/mqm/exits`.

Il file di caricamento switch a 64 bit generato viene collocato in `/var/mqm/exits64`.

Aggiunta delle informazioni di configurazione del gestore risorse per Oracle

È necessario modificare le informazioni di configurazione per il gestore code per dichiarare Oracle come partecipante alle unità di lavoro globali. La modifica delle informazioni di configurazione per il gestore code in questo modo è descritta in modo più dettagliato in “Aggiunta di informazioni di configurazione al gestore code” a pagina 49.

- Su sistemi Windows e Linux (piattaformex86 e x86-64), utilizzare IBM WebSphere MQ Explorer. Specificare i dettagli del file di caricamento switch nel pannello delle proprietà del gestore code, nel gestore risorse XA.
- Su tutti gli altri sistemi specificare i dettagli del file di caricamento switch nella stanza XAResourceManager del file `qm.ini` del gestore code.

Figura 10 a pagina 56 è un esempio di sistema UNIX and Linux che mostra una voce XAResourceManager . È necessario aggiungere un `LogDir` alla stringa di apertura XA in modo che tutte le informazioni di errore e traccia vengano registrate nello stesso luogo.

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

Figura 10. Voce XAResourceManager di esempio per Oracle su piattaforme UNIX and Linux

Nota:

1. In Figura 10 a pagina 56, la stringa xa_open è stata utilizzata con quattro parametri. Ulteriori parametri possono essere inclusi come descritto nella documentazione di Oracle.
2. Quando si utilizza il parametro IBM WebSphere MQ `ThreadOfControl=THREAD` , è necessario utilizzare il parametro Oracle `+threads=true` nella stanza XAResourceManager .

Consultare *Oracle8 Server Application Developer's Guide* per ulteriori informazioni sulla stringa xa_open.

Modifica dei parametri di configurazione di Oracle

Per ogni database Oracle coordinato dal gestore code, è necessario esaminare il numero massimo di sessioni e impostare i privilegi del database. A tale scopo, effettuare le seguenti operazioni:

Esaminare il numero massimo di sessioni
Potrebbe essere necessario esaminare le impostazioni LICENSE_MAX_SESSIONS e PROCESSES per prendere in considerazione le connessioni aggiuntive richieste dai processi appartenenti al gestore code. Per ulteriori dettagli, vedere “Installazione e configurazione del database” a pagina 48.

Imposta privilegi database
Il nome utente Oracle specificato nella stringa xa_open deve disporre dei privilegi per accedere alla vista DBA_PENDING_TRANSACTIONS, come descritto nella documentazione Oracle .

Il privilegio necessario può essere fornito utilizzando il seguente comando di esempio:

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

Configurazione Informix

Informazioni di supporto e configurazione di Informix .

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Completare i seguenti passi:

1. Assicurarsi di aver installato l'SDK client Informix appropriato:
   - I gestori code a 32 bit e le applicazioni richiedono un SDK client Informix a 32 bit.
   - Le applicazioni e i gestori code a 64 bit richiedono un SDK client Informix a 64 bit.
2. Assicurarsi che i database Informix vengano creati correttamente.

3. Controllare le impostazioni della variabile di ambiente.
4. Creare un file di caricamento switch Informix .
5. Aggiungere le informazioni di configurazione del gestore risorse.

Un elenco corrente di livelli di Informix supportati da WebSphere MQ viene fornito alla paginaIBM WebSphere MQ requisiti di sistema dettagliati .

Verifica della creazione corretta dei database Informix

Ogni database Informix che deve essere coordinato da un gestore code WebSphere MQ deve essere creato specificando il parametro `log` . Ad esempio:

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

I gestori code WebSphere MQ non sono in grado di coordinare i database Informix che non dispongono del parametro `log` specificato al momento della creazione. Se un gestore code tenta di coordinare un database Informix che non ha il parametro `log` specificato durante la creazione, la chiamata xa_open a Informix ha esito negativo e vengono generati alcuni errori FFST .

Verifica delle impostazioni della variabile d'ambiente Informix

Assicurarsi che le variabili di ambiente Informix siano impostate per i processi del gestore code ***e in*** per i processi dell'applicazione. In particolare, impostare sempre le seguenti variabili di ambiente **prima** di avviare il gestore code:

INFORMIXDIR
La directory di installazione del prodotto Informix .

- Per le applicazioni UNIX and Linux a 32 bit, utilizzare i seguenti comandi:

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

- Per le applicazioni UNIX and Linux a 64 bit, utilizzare il seguente comando:

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

- Per le applicazioni Windows , utilizzare il seguente comando:

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

Per i sistemi che hanno gestori code a 64 bit che devono supportare sia applicazioni a 32 bit che a 64 bit, è necessario che siano installati sia gli SDK client a 32 bit Informix che a 64 bit. Il makefile di esempio `xaswit.mak`, utilizzato per la creazione di un file di caricamento switch, imposta anche entrambe le directory di installazione del prodotto.

SERVER INFORMIX
Il nome del server Informix. Ad esempio, su sistemi UNIX and Linux , utilizzare:

```
export INFORMIXSERVER=hostname_1
```

Su sistemi Windows , utilizzare:

```
set INFORMIXSERVER=hostname_1
```

ONCONFIG
Il nome del file di configurazione del server Informix . Ad esempio, su sistemi UNIX and Linux , utilizzare:

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

Su sistemi Windows , utilizzare:

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

Creazione del Informix file di caricamento switch

Per creare Informix file di caricamento switch, utilizzare il file di esempio xaswit.mak, fornito da WebSphere MQ per creare i file di caricamento switch per vari prodotti database. Su sistemi Windows , è possibile trovare xaswit.mak nella directory *MQ_INSTALLATION_PATH*`\tools\c\samples\xatm` *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ . Per creare il file di caricamento switch Informix con Microsoft Visual C + +, utilizzare:

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

Il file switch generato viene inserito in `c:\Program Files\IBM\WebSphere MQ\exits`.

È possibile trovare xaswit.mak nella directory *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Modificare xaswit.mak per *annullare il commento* delle righe appropriate alla versione di Informix che si sta utilizzando. Quindi, eseguire il makefile utilizzando il comando:

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

Il file di caricamento dello switch a 32 bit generato si trova in `/var/mqm/exits`.

Il file di caricamento switch a 64 bit generato viene collocato in `/var/mqm/exits64`.

Aggiunta delle informazioni di configurazione del gestore risorse per Informix

È necessario modificare le informazioni di configurazione per il gestore code per dichiarare Informix come partecipante alle unità di lavoro globali. La modifica delle informazioni di configurazione per il gestore code in questo modo viene descritta più dettagliatamente in “Aggiunta di informazioni di configurazione al gestore code” a pagina 49.

- Su sistemi Windows e Linux (piattaformex86 e x86-64), utilizzare WebSphere MQ Explorer. Specificare i dettagli del file di caricamento switch nel pannello delle proprietà del gestore code, nel gestore risorse XA.
- Su tutti gli altri sistemi specificare i dettagli del file di caricamento switch nella stanza XAResourceManager del file qm.ini del gestore code.

Figura 11 a pagina 58 è un esempio UNIX , che mostra una voce qm.ini XAResourceManager in cui il database da coordinare è denominato `mydbname`, questo nome è specificato in `XAOpenString`:

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

Figura 11. Voce di esempio XAResourceManager per Informix su piattaforme UNIX

Nota: Per impostazione predefinita, il file xaswit.mak di esempio su piattaforme UNIX crea un file di caricamento switch che utilizza librerie Informix con thread. È necessario assicurarsi che ThreadOfControl sia impostato su THREAD quando si utilizzano queste librerie Informix . In Figura 11 a pagina 58, l'attributo della stanza XAResourceManager del file qm.ini ThreadOfControl è impostato su THREAD. Quando viene specificato THREAD, è necessario creare le applicazioni utilizzando le librerie Informix con thread e le librerie API con thread WebSphere MQ .

L'attributo XAOpenString deve contenere il nome del database, seguito dal simbolo @ e seguito dal nome server Informix .

Per utilizzare le librerie Informix senza thread, è necessario assicurarsi che l'attributo della stanza XAResourceManager del file qm.ini ThreadOfControl sia impostato su PROCESS. È inoltre necessario apportare le modifiche riportate di seguito all'esempio xaswit.mak:

1. Eliminare il commento dalla generazione di un file di caricamento switch senza thread.
2. Impostare come commento la generazione del file di caricamento switch con thread.

Configurazione Sybase

Informazioni di supporto e configurazione Sybase .

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Completare i seguenti passi:

1. Assicurarsi di aver installato le librerie XA Sybase , ad esempio installando l'opzione XA DTM.
2. Controllare le impostazioni della variabile di ambiente.
3. Abilitare il supporto XA Sybase
4. Creare il file di caricamento switch Sybase .
5. Aggiungere le informazioni di configurazione del gestore risorse.

Nella pagina IBM WebSphere MQ requisiti di sistema dettagliati viene fornito un elenco corrente di livelli di Sybase supportati da WebSphere MQ .

Verifica delle impostazioni delle variabili di ambiente Sybase

Assicurarsi che le variabili di ambiente Sybase siano impostate per i processi del gestore code ***e in*** per i processi dell'applicazione. In particolare, impostare sempre le seguenti variabili di ambiente **prima** di avviare il gestore code:

Sybase
L'ubicazione dell'installazione del prodotto Sybase . Ad esempio, su sistemi UNIX and Linux , utilizzare:

```
export SYBASE=/sybase
```

Su sistemi Windows , utilizzare:

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

SYBASE_OCS
La directory in SYBASE in cui sono installati i file client Sybase . Ad esempio, su sistemi UNIX and Linux , utilizzare:

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

Su sistemi Windows , utilizzare:

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

Abilitazione del supporto XA Sybase

All'interno del Sybase file di configurazione XA $SYBASE/$SYBASE_OCS/xa_config, definire un LRM (Logical Resource Manager) per ogni connessione al server Sybase in fase di aggiornamento. Un esempio del contenuto di $SYBASE/$SYBASE_OCS/xa_config viene mostrato in Figura 12 a pagina 60.

```
# The first line must always be a comment
[xa]
  LRM=lrmname
  server=servername
```

Figura 12. Contenuto di esempio di $SYBASE/ $SYBASE_OCS/xa_config

Creazione del file di caricamento switch Sybase

Per creare il file di caricamento degli switch Sybase , utilizzare i file di esempio forniti con WebSphere MQ.Su sistemi Windows , è possibile trovare xaswit.mak nella directory `C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm`. Per creare il file di caricamento switch Sybase con Microsoft Visual C + +, utilizzare:

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

Il file switch generato viene inserito in `c:\Program Files\IBM\WebSphere MQ\exits`.

È possibile trovare xaswit.mak nella directory *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Modificare xaswit.mak per *eliminare il commento* dalle righe appropriate alla versione di Sybase che si sta utilizzando. Quindi, eseguire il makefile utilizzando il comando:

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

Il file di caricamento switch a 32 bit generato viene inserito in `/var/mqm/exits`.

Il file di caricamento dello switch a 64 bit generato si trova in `/var/mqm/exits64`.

Aggiunta delle informazioni di configurazione del gestore risorse per Sybase

È necessario modificare le informazioni di configurazione per il gestore code per dichiarare Sybase come partecipante nelle unità di lavoro globali. La modifica delle informazioni di configurazione è descritta più dettagliatamente in “Aggiunta di informazioni di configurazione al gestore code” a pagina 49.

- Su sistemi Windows e Linux (piattaformex86 e x86-64), utilizzare WebSphere MQ Explorer. Specificare i dettagli del file di caricamento switch nel pannello delle proprietà del gestore code, nel gestore risorse XA.
- Su tutti gli altri sistemi specificare i dettagli del file di caricamento switch nella stanza XAResourceManager nel file qm.ini del gestore code.

La Figura 13 a pagina 61 mostra un UNIX and Linux esempio, che utilizza il database associato alla definizione LRM *lrmname* nel file di configurazione XA Sybase , $SYBASE/$SYBASE_OCS/`xa_config`. Includere un nome file di log se si desidera che le chiamate della funzione XA vengano registrate:

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

Figura 13. Voce di esempio XAResourceManager per Sybase su piattaforme UNIX and Linux

Utilizzo di programmi a più thread con Sybase

Se si utilizzano programmi a più thread con WebSphere MQ unità di lavoro globali che incorporano gli aggiornamenti di Sybase, ***è necessario*** utilizzare il valore THREAD per il parametro di controllo ThreadOf. Assicurarsi inoltre di collegare il proprio programma (e il file di caricamento switch) con le librerie thread - safe Sybase (versioni _r). L'uso del valore THREAD per il parametro di controllo ThreadOfviene mostrato in Figura 13 a pagina 61.

Configurazioni di più database

Se si desidera configurare il gestore code in modo che gli aggiornamenti a più database possano essere inclusi nelle unità di lavoro globali, aggiungere una sezione XAResourceManager per ciascun database.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Se i database sono tutti gestiti dallo stesso gestore database, ogni stanza definisce un database separato. Ogni stanza specifica lo stesso *SwitchFile*, ma il contenuto di *XAOpenString* è diverso poiché specifica il nome del database in fase di aggiornamento. Ad esempio, le stanze visualizzate in Figura 14 a pagina 61 configurano il gestore code con i database Db2 *MQBankDB* e *MQFeeDB* su sistemi UNIX and Linux .

Importante: Non è possibile avere più stanze che puntano allo stesso database. Questa configurazione non funziona in nessuna circostanza e se si prova questa configurazione non riesce.

Si riceveranno errori nel formato `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
```

Figura 14. Voci XAResourceManager di esempio per più database Db2

Se i database da aggiornare sono gestiti da gestori database differenti, aggiungere una sezione XAResourceManager per ciascuno. In questo caso, ogni stanza specifica un *SwitchFile*differente. Ad esempio, se *MQFeeDB* è gestito da Oracle invece di DB2, utilizzare le seguenti stanze sui sistemi 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
```

Figura 15. Voci XAResourceManager di esempio per un database DB2 e Oracle

In linea di principio, non esiste un limite al numero di istanze del database che possono essere configurate con un singolo gestore code.

Nota: Per informazioni sul supporto per l'inclusione dei database Informix in più aggiornamenti di database all'interno delle unità di lavoro globali, controllare il file readme del prodotto.

Considerazioni sulla sicurezza

Considerazioni per l'esecuzione del database nel modello XA.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Le seguenti informazioni sono fornite solo come guida. In tutti i casi, fare riferimento alla documentazione fornita con il gestore database per determinare le implicazioni di sicurezza dell'esecuzione del database nel modello XA.

Un processo di applicazione indica l'avvio di un'unità di lavoro globale utilizzando il comando MQBEGIN . La prima chiamata MQBEGIN che un'applicazione emette si connette a tutti i database partecipanti richiamando il codice della libreria client al punto di ingresso xa_open. Tutti i gestori database forniscono un meccanismo per fornire un ID utente e una password in XAOpenString. Questa è l'unica volta che vengono trasmesse le informazioni di autenticazione.

Notare che, sulle piattaforme UNIX and Linux , le applicazioni fastpath devono essere eseguite con un ID utente effettivo di mqm durante le chiamate MQI.

Considerazioni quando si perde il contatto con il gestore risorse XA

Il gestore code tollera i gestori database non disponibili. Ciò significa che è possibile avviare e arrestare il gestore code indipendentemente dal server di database. Quando il contatto viene ripristinato, il gestore code e il database vengono risincronizzati. È anche possibile utilizzare il comando rsvmqtrn per risolvere manualmente le unità di lavoro in dubbio.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Nelle normali operazioni, è necessaria solo una quantità minima di gestione dopo aver completato i passi di configurazione. Il lavoro di gestione è reso più semplice perché il gestore code tollera i gestori database non disponibili. Ciò significa in particolare che:

- Il gestore code può essere avviato in qualsiasi momento senza prima avviare ciascun gestore database.
- Il gestore code non deve essere arrestato e riavviato se uno dei gestori database diventa non disponibile.

Ciò consente di avviare e arrestare il gestore code indipendentemente dal server di database.

Ogni volta che si perde il contatto tra il gestore code e un database, è necessario risincronizzarli quando entrambi diventano nuovamente disponibili. La risincronizzazione è il processo mediante il quale vengono

completate tutte le unità di lavoro in dubbio che coinvolgono tale database. In generale, ciò si verifica automaticamente senza la necessità di intervento dell'utente. Il gestore code richiede al database un elenco di unità di lavoro in dubbio. Indica quindi al database di eseguire il commit o il rollback di ciascuna di queste unità di lavoro in dubbio.

Quando un gestore code viene avviato, viene risincronizzato con ciascun database. Quando un singolo database diventa non disponibile, solo tale database deve essere risincronizzato la volta successiva che il gestore code nota che è nuovamente disponibile.

Il gestore code riacquista automaticamente il contatto con un database precedentemente non disponibile quando vengono avviate nuove unità di lavoro globali con MQBEGIN. Esegue questa operazione richiamando la funzione xa_open nella libreria del client di database. Se questa chiamata xa_open ha esito negativo, MQBEGIN restituisce un codice di completamento di MQCC_WARNING e un codice motivo di MQRC_PARTICIPANT_NOT_AVAILABLE. È possibile ritentare la chiamata MQBEGIN successivamente.

Non continuare a tentare un'unità di lavoro globale che implica aggiornamenti a un database che ha indicato un errore durante MQBEGIN. Non ci sarà una connessione al database tramite cui è possibile effettuare gli aggiornamenti. Le uniche opzioni disponibili sono la chiusura del programma o la ripetizione periodica di MQBEGIN, nella speranza che il database diventi nuovamente disponibile.

In alternativa, è possibile utilizzare il comando `rsvmqtrn` per risolvere esplicitamente tutte le unità di lavoro in dubbio.

Unità di lavoro in dubbio

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Un database potrebbe essere lasciato con unità di lavoro in dubbio se il contatto con il gestore code viene perso dopo che il gestore database è stato istruito per la preparazione. Finché il server di database non riceve il risultato dal gestore code (commit o rollback), deve mantenere i blocchi del database associati agli aggiornamenti.

Poiché questi blocchi impediscono ad altre applicazioni di aggiornare o leggere i record del database, la risincronizzazione deve essere eseguita il più presto possibile.

Se, per qualche motivo, non è possibile attendere la risincronizzazione automatica del gestore code con il database, è possibile utilizzare le funzioni fornite dal gestore database per eseguire manualmente il commit o il rollback degli aggiornamenti del database. In *X/Open Distributed Transaction Processing: The XA Specification*, ciò viene chiamato prendendo una decisione *euristica* . Utilizzarla solo come ultima risorsa a causa della possibilità di compromettere l'integrità dei dati; è possibile, ad esempio, eseguire erroneamente il rollback degli aggiornamenti del database quando tutti gli altri partecipanti hanno eseguito il commit degli aggiornamenti.

È molto meglio riavviare il gestore code o utilizzare il comando `rsvmqtrn` quando il database è stato riavviato per avviare la risincronizzazione automatica.

Visualizzazione delle unità di lavoro in sospeso con il comando dspmqtrn

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Mentre un gestore database non è disponibile, è possibile utilizzare il comando **dspmqtrn** per verificare lo stato delle unità di lavoro globali in sospeso che coinvolgono tale database.

Il comando **dspmqtrn** visualizza solo le unità di lavoro in cui uno o più partecipanti sono in dubbio. I partecipanti sono in attesa della decisione del gestore code di eseguire il commit o il rollback degli aggiornamenti preparati.

Per ognuna di queste unità di lavoro globali, lo stato di ciascun partecipante viene visualizzato nell'output da **dspmqtrn**. Se l'unità di lavoro non ha aggiornato le risorse di un particolare gestore risorse, non viene visualizzata.

Per quanto riguarda un'unità di lavoro in dubbio, si dice che un gestore risorse abbia eseguito una delle seguenti operazioni:

Preparato
Il gestore risorse è pronto per eseguire il commit degli aggiornamenti.

Con commit
Il gestore risorse ha eseguito il commit dei propri aggiornamenti.

Sottoposto a rollback
Il gestore risorse ha eseguito il rollback dei propri aggiornamenti.

Ha partecipato
Il gestore risorse è un partecipante, ma non ha preparato, eseguito il commit o eseguito il rollback dei suoi aggiornamenti.

Quando il gestore code viene riavviato, richiede a ogni database che dispone di una stanza XAResourceManager un elenco delle relative unità di lavoro globali in dubbio. Se il database non è stato riavviato o non è disponibile, il gestore code non può ancora consegnare al database i risultati finali per tali unità di lavoro. Il risultato delle unità di lavoro in dubbio viene consegnato al database alla prima occasione quando il database è nuovamente disponibile.

In questo caso, il gestore database viene notificato come in stato *preparato* fino a quando non si verifica la risincronizzazione.

Ogni volta che il comando `dspmqtrn` visualizza un'unità di lavoro in dubbio, elenca prima tutti i possibili gestori risorse che potrebbero partecipare. A questi viene assegnato un identificativo univoco, *RMId*, che viene utilizzato al posto del *Nome* dei gestori risorse durante la notifica del loro stato rispetto a un'unità di lavoro in dubbio.

Output dspmqtrn di esempio mostra il risultato dell'immissione del seguente comando:

```
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 [3291A50600002013746573 74]
         bqual [00000001]
AMQ7105: Resource manager 0 has committed.
AMQ7104: Resource manager 1 has prepared.
AMQ7104: Resource manager 2 has prepared.
```

dove *Numero transazione* è l'ID della transazione che può essere utilizzato con il comando rsvmqtrn . Consultare il prodotto AMQ7000-7999: WebSphere MQ per ulteriori informazioni sul messaggio AMQ7056 . Le variabili *XID* fanno parte della *Specifica X/Open XA*; per le informazioni più aggiornate su questa specifica, consultare: https://publications.opengroup.org/c193.

Figura 16. Output dspmqtrn di esempio

L'output in Output dspmqtrn di esempio mostra che ci sono tre gestori risorse associati al gestore code. Il primo è il gestore risorse `0`, che è il gestore code stesso. Le altre due istanze del gestore risorse sono i database `MQBankDB` e `MQFeeDB` Db2 .

L'esempio mostra solo una singola unità di lavoro in dubbio. Viene emesso un messaggio per tutti e tre i gestori risorse, il che significa che sono stati effettuati aggiornamenti al gestore code e ai database Db2 all'interno dell'unità di lavoro.

Gli aggiornamenti effettuati al gestore code, gestore risorse **0**, sono stati *sottoposti a commit*. Gli aggiornamenti ai database Db2 sono nello stato *preparato* , il che significa che Db2 deve essere diventato non disponibile prima di essere stato chiamato per eseguire il commit degli aggiornamenti ai database *MQBankDB* e *MQFeeDB* .

L'unità di lavoro in dubbio ha un identificativo esterno denominato XID (*id transazione*). Si tratta di una parte di dati forniti a Db2 dal gestore code per identificare la relativa parte dell'unità di lavoro globale.

Risoluzione delle unità di lavoro in sospeso con il comando rsvmqtrn
Le unità di lavoro in sospeso vengono completate quando il gestore code e DB2 vengono risincronizzati.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

L'output mostrato in Figura 16 a pagina 64 mostra una singola unità di lavoro in dubbio in cui la decisione di commit deve ancora essere consegnata ai database DB2 .

Per completare questa unità di lavoro, il gestore code e DB2 devono risincronizzare quando DB2 diventa disponibile. Il gestore code utilizza l'avvio di nuove unità di lavoro come opportunità per riprendere contatto con DB2. In alternativa, è possibile indicare al gestore code di risincronizzare esplicitamente utilizzando il comando **rsvmqtrn** .

Eseguire questa operazione subito dopo che DB2 è stato riavviato, in modo che tutti i blocchi del database associati all'unità di lavoro in dubbio vengano rilasciati il più rapidamente possibile. Utilizzare l'opzione -a, che indica al gestore code di risolvere tutte le unità di lavoro in dubbio. Nel seguente esempio, DB2 è stato riavviato, in modo che il gestore code possa risolvere l'unità di lavoro in dubbio:

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

Risultati ed errori misti
Sebbene il gestore code utilizzi un protocollo di commit a due fasi, ciò non elimina completamente la possibilità che alcune unità di lavoro si completino con risultati misti. Questo è il punto in cui alcuni partecipanti eseguono il commit dei propri aggiornamenti e alcuni ne eseguono il backout.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Le unità di lavoro che si completano con un risultato misto hanno implicazioni gravi perché le risorse condivise che avrebbero dovuto essere aggiornate come una singola unità di lavoro non sono più in uno stato coerente.

I risultati contrastanti sono causati principalmente quando vengono prese decisioni euristiche sulle unità di lavoro invece di consentire al gestore code di risolvere le unità di lavoro in dubbio. Tali decisioni sono al di fuori del controllo del gestore code.

Ogni volta che il gestore code rileva un risultato misto, produce informazioni FFST e documenta l'errore nei log degli errori, con uno dei seguenti due messaggi:

- Se un gestore database esegue il rollback invece di eseguire il commit:

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

- Se un gestore database esegue il commit invece di eseguire il rollback:

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

Ulteriori messaggi identificano i database che sono danneggiati euristicamente. È quindi responsabilità dell'utente ripristinare localmente la congruenza nei database interessati. Si tratta di una procedura complicata in cui è necessario prima isolare l'aggiornamento di cui è stato eseguito erroneamente il commit o il rollback, quindi annullare o ripetere manualmente la modifica del database.

Modifica delle informazioni di configurazione
Dopo che il gestore code è stato avviato correttamente per coordinare le unità di lavoro globali, non modificare le informazioni di configurazione del gestore risorse.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Se è necessario modificare le informazioni di configurazione, è possibile farlo in qualsiasi momento, ma le modifiche non diventano effettive fino a quando il gestore code non viene riavviato.

Se si rimuovono le informazioni di configurazione del gestore risorse per un database, si sta effettivamente eliminando la possibilità per il gestore code di contattare tale gestore database.

Mai modificare l'attributo *Nome* nelle informazioni di configurazione del gestore risorse. Questo attributo identifica in modo univoco tale istanza del gestore database per il gestore code. Se si modifica questo identificativo univoco, il gestore code presume che il database sia stato rimosso e che sia stata aggiunta un'istanza completamente nuova. Il gestore code associa ancora le unità di lavoro in sospeso con il vecchio *Nome*, probabilmente lasciando il database in uno stato in dubbio.

Rimozione delle istanze del gestore database
Se è necessario rimuovere definitivamente un database dalla configurazione, assicurarsi che il database non sia in dubbio prima di riavviare il gestore code.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

I prodotti database forniscono comandi per elencare le transazioni in dubbio. Se sono presenti transazioni in dubbio, prima consentire al gestore code di risincronizzarsi con il database. Eseguire questa operazione avviando il gestore code. È possibile verificare che la risincronizzazione sia stata eseguita utilizzando il comando **rsvmqtrn** o il comando del database per visualizzare le unità di lavoro in dubbio. Una volta completata la risincronizzazione, chiudere il gestore code e rimuovere le informazioni di configurazione del database.

Se non si riesce a osservare questa procedura, il gestore code ricorda ancora tutte le unità di lavoro in dubbio che coinvolgono tale database. Un messaggio di avviso, AMQ7623, viene emesso ogni volta che il gestore code viene riavviato. Se non si configurerà mai più questo database con il gestore code, utilizzare l'opzione -r del comando **rsvmqtrn** per indicare al gestore code di dimenticare la partecipazione del database alle relative transazioni in dubbio. Il gestore code dimentica tali transazioni solo quando le transazioni in dubbio sono state completate con tutti i partecipanti.

In alcuni casi potrebbe essere necessario rimuovere temporaneamente alcune informazioni di configurazione del gestore risorse. Nei sistemi UNIX and Linux , ciò è possibile impostando come commento la stanza in modo che possa essere facilmente reintegrata in un secondo momento. È possibile decidere di eseguire questa operazione se si verificano errori ogni volta che il gestore code contatta un determinato database o gestore database. La rimozione temporanea delle informazioni di configurazione del gestore risorse in questione consente al gestore code di avviare le unità di lavoro globali che coinvolgono tutti i partecipanti. Di seguito è riportato un esempio di stanza `XAResourceManager` commentata:

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

Figura 17. Stanza XAResourceManager commentata su sistemi UNIX and Linux

Sui sistemi Windows , utilizzare WebSphere MQ Explorer per eliminare le informazioni sull'istanza del gestore database. Prestare particolare attenzione a immettere il nome corretto nel campo *Nome* quando viene ripristinato. Se si digita erroneamente il nome, è possibile che si verifichino problemi in dubbio, come descritto in “Modifica delle informazioni di configurazione” a pagina 66.

Registrazione dinamica XA

La specifica XA consente di ridurre il numero di chiamate `xa_*` effettuate da un gestore transazioni a un gestore risorse. Questa ottimizzazione è nota come *registrazione dinamica*.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

La registrazione dinamica è supportata da DB2. Altri database potrebbero supportarlo; consultare la documentazione per il prodotto database per i dettagli.

Perché l'ottimizzazione della registrazione dinamica è utile? Nell'applicazione, alcune unità di lavoro globali potrebbero contenere aggiornamenti alle tabelle del database; altre potrebbero non contenere tali aggiornamenti. Quando non è stato effettuato alcun aggiornamento persistente per le tabelle di un database, non è necessario includere tale database nel protocollo di commit che si verifica durante MQCMIT.

Se il database supporta la registrazione dinamica, l'applicazione richiama `xa_open` durante la prima chiamata MQBEGIN su una connessione WebSphere MQ . Richiama inoltre `xa_close` sulla successiva chiamata MQDISC. Il modello delle successive chiamate XA dipende dal fatto che il database supporti o meno la registrazione dinamica:

Se il database non supporta la registrazione dinamica ...
Ogni unità di lavoro globale coinvolge diverse chiamate di funzioni XA effettuate dal codice WebSphere MQ nella libreria del client del database, indipendentemente dal fatto che sia stato effettuato un aggiornamento permanente alle tabelle di tale database all'interno dell'unità di lavoro. Queste includono:

- `xa_start` e `xa_end` dal processo dell'applicazione. Vengono utilizzati per dichiarare l'inizio e la fine di un'unità di lavoro globale.
- `xa_prepare`, `xa_commite` `xa_rollback` dal processo dell'agent gestore code, amqzlaa0. Questi vengono utilizzati per consegnare il risultato dell'unità di lavoro globale: la decisione di commit o rollback.

Inoltre, il processo dell'agent del gestore code richiama anche `xa_open` durante la prima MQBEGIN.

Se il database supporta la registrazione dinamica ...
Il codice WebSphere MQ effettua solo le chiamate alla funzione XA necessarie. Per un'unità di lavoro globale che **non** ha implicato aggiornamenti permanenti alle risorse del database, non vi sono **chiamate** XA al database. Per un'unità di lavoro globale che **ha** ha coinvolto tali aggiornamenti persistenti, le chiamate devono:

- `xa_end` dal processo di applicazione per dichiarare la fine dell'unità di lavoro globale.

- `xa_prepare`, `xa_commite` `xa_rollback` dal processo dell'agent gestore code, amqzlaa0. Questi vengono utilizzati per consegnare il risultato dell'unità di lavoro globale: la decisione di commit o rollback.

Per il funzionamento della registrazione dinamica, è fondamentale che il database abbia un modo di informare WebSphere MQ quando ha eseguito un aggiornamento persistente che desidera essere incluso nell'unità di lavoro globale corrente. WebSphere MQ fornisce la funzione `ax_reg` per questo scopo.

Il codice client del database che viene eseguito nel tuo processo dell'applicazione trova la funzione `ax_reg` e la richiama, per *registrare dinamicamente* il fatto che ha eseguito un lavoro persistente all'interno dell'unità di lavoro globale corrente. In risposta a questa chiamata `ax_reg` , WebSphere MQ registra che il database ha partecipato. Se questa è la prima chiamata `ax_reg` su questa WebSphere MQ , il processo dell'agent del gestore code richiama `xa_open`.

Il codice client del database effettua questa chiamata `ax_reg` quando è in esecuzione nel processo, ad esempio, durante una chiamata SQL UPDATE o qualsiasi chiamata nell'API client del database è responsabile

Condizioni di errore
Nella registrazione dinamica XA c'è la possibilità di un errore che crea confusione nel gestore code.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Un esempio comune è se si dimentica di impostare correttamente le variabili di ambiente del database prima di avviare il gestore code, le chiamate del gestore code a `xa_open` non riescono. Non è possibile utilizzare unità di lavoro globali.

Per evitare ciò, assicurarsi di aver impostato le relative variabili di ambiente prima di avviare il gestore code. Esaminare la documentazione del prodotto database e i consigli forniti in “Configurazione di Db2” a pagina 51, “Configurazione di Oracle” a pagina 54e “Configurazione Sybase” a pagina 59.

Con tutti i prodotti database, il gestore code richiama `xa_open` una sola volta all'avvio del gestore code, come parte della sessione di recupero (come spiegato in “Considerazioni quando si perde il contatto con il gestore risorse XA” a pagina 62). Questa chiamata `xa_open` non riesce se si impostano le variabili di ambiente del database in modo non corretto, ma non causa l'errore di avvio del gestore code. Questo perché lo stesso codice di errore `xa_open` viene utilizzato dalla libreria del client del database per indicare che il server del database non è disponibile. WebSphere MQ non considera questo come un errore grave, poiché il gestore code deve essere in grado di continuare l'elaborazione dei dati all'esterno delle unità di lavoro globali che coinvolgono tale database.

Le chiamate successive a `xa_open` vengono effettuate dal gestore code durante la prima MQBEGIN su una connessione WebSphere MQ (se non viene utilizzata la registrazione dinamica) o durante una chiamata dal codice client del database alla funzione WebSphere MQfornita `ax_reg` (se viene utilizzata la registrazione dinamica).

La **tempistica** di eventuali condizioni di errore (o, occasionalmente, report FFST) dipende dall'utilizzo della registrazione dinamica:

- Se si sta utilizzando la registrazione dinamica, la chiamata MQBEGIN potrebbe avere esito positivo, ma la chiamata al database SQL UPDATE (o simile) avrà esito negativo.
- Se non si sta utilizzando la registrazione dinamica, la chiamata MQBEGIN avrà esito negativo.

Verificare che le variabili di ambiente siano impostate correttamente nei processi dell'applicazione e del gestore code.

Riepilogo delle chiamate XA
Di seguito è riportato un elenco delle chiamate effettuate alle funzioni XA in una libreria client del database come risultato delle diverse chiamate MQI che controllano le unità di lavoro globali. Questa non è una descrizione completa del protocollo descritto nella specifica XA; viene fornita come una breve panoramica.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Notare che le chiamate `xa_start` e `xa_end` vengono sempre richiamate dal codice WebSphere MQ nel processo dell'applicazione, mentre `xa_prepare`, `xa_commite` `xa_rollback` vengono sempre richiamate dal processo dell'agent gestore code, amqzlaa0.

Le chiamate `xa_open` e `xa_close` visualizzate in questa tabella vengono tutte effettuate dal processo dell'applicazione. Il processo dell'agent del gestore code richiama `xa_open` nelle circostanze descritte in “Condizioni di errore” a pagina 68.

Tabella 7. Riepilogo delle chiamate della funzione XA

Chiamata MQI	**Chiamate XA effettuate con registrazione dinamica**	**Chiamate XA effettuate senza registrazione dinamica**
Primo MQBEGIN	`xa_open`	`xa_open` `xa_start`
MQBEGIN successivo	Nessuna chiamata XA	`xa_start`

MQCMIT (**senza che** `ax_reg` venga chiamato durante l'unità di lavoro globale corrente)	Nessuna chiamata XA	`xa_end`
		`xa_prepare`
		`xa_commit`
		`xa_rollback`

MQCMIT (**con** `ax_reg` richiamato durante l'unità di lavoro globale corrente)	`xa_end`
	`xa_prepare`
	`xa_commit`
	`xa_rollback`
		Non applicabile. Nessuna chiamata viene effettuata a ax_reg in modalità non dinamica.

MQBACK (**senza che** `ax_reg` venga richiamato durante l'unità di lavoro globale corrente)	Nessuna chiamata XA	`xa_end` `xa_rollback`
MQBACK (**con** `ax_reg` richiamato durante l'unità di lavoro globale corrente)	`xa_end` `xa_rollback`	Non applicabile. Nessuna chiamata viene effettuata a ax_reg in modalità non dinamica.
MQDISC, dove MQCMIT o MQBACK è stato richiamato per primo. In caso contrario, l'elaborazione MQCMIT viene eseguita per la prima volta durante MQDISC.	`xa_close`	`xa_close`

Note:

1. Per MQCMIT, `xa_commit` viene richiamato se `xa_prepare` ha esito positivo. Altrimenti, viene richiamato `xa_rollback` .

Scenario 2: Altri software forniscono il coordinamento

Nello scenario 2, un gestore transazioni esterno coordina le unità di lavoro globali, avviandole e eseguendone il commit sotto il controllo dell'API del gestore transazioni. I verbi MQBEGIN, MQCMIT e MQBACK non sono disponibili.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

In questa sezione viene descritto questo scenario, incluso:

- "Coordinamento del punto di sincronizzazione esterno" a pagina 70
- "Utilizzo di CICS" a pagina 72
- "Utilizzo di Microsoft Transaction Server (COM +)" a pagina 77

Il client IBM WebSphere MQ per HP Integrity NonStop Server può utilizzare HP NonStop Transaction Management Facility (TMF) per coordinare le unità di lavoro globali. Per ulteriori informazioni, vedere Utilizzo di HP NonStop TMF.

Coordinamento del punto di sincronizzazione esterno

Un'unità di lavoro globale può anche essere coordinata da un gestore transazioni X/Open XA esterno. Qui il gestore code WebSphere MQ partecipa, ma non coordina, l'unità di lavoro.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Il flusso di controllo in un'unità di lavoro globale coordinata da un gestore transazioni esterno è il seguente:

1. Un'applicazione indica al coordinatore del punto di sincronizzazione esterno (ad esempio, TXSeries) che desidera avviare una transazione.
2. Il coordinatore del punto di sincronizzazione indica ai gestori risorse noti, come WebSphere MQ, la transazione corrente.
3. L'applicazione emette chiamate ai gestori risorse associati alla transazione corrente. Ad esempio, l'applicazione potrebbe emettere chiamate MQGET a WebSphere MQ.
4. L'applicazione emette una richiesta di commit o backout al coordinatore del punto di sincronizzazione esterno.
5. Il coordinatore del punto di sincronizzazione completa la transazione emettendo le chiamate appropriate a ciascun gestore risorse, generalmente utilizzando protocolli di commit a due fasi.

I livelli supportati di coordinatori del punto di sincronizzazione esterno che possono fornire un processo di commit a due fasi per transazioni in cui WebSphere MQ partecipa sono definiti in IBM WebSphere MQ requisiti di sistema dettagliati.

Il resto di questa sezione descrive come abilitare le unità di lavoro esterne.

La struttura dello switch IBM WebSphere MQ XA

Ogni gestore risorse che partecipa a un'unità di lavoro coordinata esternamente deve fornire una struttura di switch XA. Questa struttura definisce sia le capacità del gestore risorse che le funzioni che devono essere richiamate dal coordinatore del punto di sincronizzazione.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

IBM WebSphere MQ fornisce due versioni di questa struttura:

- *MQRMIXASwitch* per la gestione delle risorse XA statiche
- *MQRMIXASwitchDynamic* per la gestione dinamica delle risorse XA

Consultare la documentazione del gestore transazioni per determinare se utilizzare l'interfaccia di gestione delle risorse statica o dinamica. Ovunque un gestore transazioni lo supporti, si consiglia di utilizzare la gestione delle risorse XA dinamica.

Alcuni gestori transazioni a 64 bit considerano il tipo *long* nella specifica XA come 64 bit, mentre altri lo considerano come 32 bit. WebSphere MQ supporta entrambi i modelli:

- Se il gestore transazioni è a 32 bit o il gestore transazioni è a 64 bit ma considera il tipo *long* come a 32 bit, utilizzare il file di caricamento switch elencato in Tabella 8 a pagina 71.
- Se il gestore transazioni è a 64 bit e tratta il tipo *long* come a 64 bit, utilizzare il file di caricamento switch elencato in Tabella 9 a pagina 71.

Un elenco di gestori transazioni a 64 bit noti che considerano il tipo *long* come a 64 bit viene fornito in Tabella 10 a pagina 72. Consultare la documentazione del gestore transazioni se non si è certi del modello utilizzato dal gestore transazioni.

Tabella 8. Nomi file di caricamento switch XA

Piattaforma	**Nome file di caricamento switch (server)**	**Nome file di caricamento switch (client transazionale esteso)**
Finestre	*mqmxa.dll*	*mqcxa.dll*
AIX (senza thread)	*libmqmxa.a*	*libmqcxa.a*
AIX (con thread)	*libmqmxa_r.a*	*libmqcxa_r.a*
HP-UX (senza thread)	*libmqmxa.so*	*libmqcxa.so*
HP-UX (con thread)	*libmqmxa_r.so*	*libmqcxa_r.so*
Linux (senza thread)	*libmqmxa.so*	*libmqcxa.so*
Linux (con thread)	*libmqmxa_r.so*	*libmqcxa_r.so*
Solaris	*libmqmxa.so*	*libmqcxa.so*

Tabella 9. Nomi file di caricamento switch XA a 64 bit alternativi

Piattaforma	**Nome file di caricamento switch (server)**	**Nome file di caricamento switch (client transazionale esteso)**
AIX (senza thread)	*libmqmxa64.a*	*libmqcxa64.a*
AIX (con thread)	*libmqmxa64_r.a*	*libmqcxa64_r.a*
HP-UX (senza thread)	*libmqmxa64.so*	*libmqcxa64.so*
HP-UX (con thread)	*libmqmxa64_r.so*	*libmqcxa64_r.so*
Linux (senza thread)	*libmqmxa64.so*	*libmqcxa64.so*
Linux (con thread)	*libmqmxa64_r.so*	*libmqcxa64_r.so*
Solaris	*libmqmxa64.so*	*libmqcxa64.so*

Tabella 10. Gestori transazioni a 64 bit che richiedono il file di caricamento switch a 64 bit alternativo
Transaction Manager
Tuxedo

Alcuni coordinatori di punti di sincronizzazione esterni (non CICS) richiedono che ciascun gestore risorse che partecipa a un'unità di lavoro fornisca il suo nome nel campo del nome della struttura dello switch XA. Il nome del gestore risorse WebSphere MQ è MQSeries_XA_RMI.

Il coordinatore del punto di sincronizzazione definisce il modo in cui la struttura di switch XA WebSphere MQ si collega ad essa. Le informazioni sul collegamento della struttura switch XA WebSphere MQ con CICS sono fornite in "Utilizzo di CICS" a pagina 72. Per informazioni sul collegamento della struttura dello switch XA WebSphere MQ con altri coordinatori del punto di sincronizzazione conformi a XA, consultare la documentazione fornita con tali prodotti.

Le seguenti considerazioni si applicano all'utilizzo di WebSphere MQ con tutti i coordinatori del punto di sincronizzazione conformi a XA:

- La struttura xa_info inoltrata a qualsiasi chiamata xa_open dal coordinatore del punto di sincronizzazione include il nome di un gestore code WebSphere MQ . Il nome ha lo stesso formato del nome del gestore code passato alla chiamata `MQCONN` . Se il nome inoltrato alla chiamata xa_open è vuoto, viene utilizzato il gestore code predefinito.

  In alternativa, la struttura xa_info può contenere valori per i parametri *TPM* e *AXLIB* . Il parametro *TPM* specifica il gestore transazioni utilizzato. I valori validi sono CICS, TUXEDO e ENCINA. Il parametro *AXLIB* specifica il nome della libreria che contiene le funzioni ax_reg e ax_unreg del gestore transazioni. Per ulteriori informazioni su questi parametri, consultare Configurazione di un client transazionale esteso. Se la struttura xa_info contiene uno di tali parametri, il nome del gestore code viene specificato nel parametro *QMNAME* , a meno che non venga utilizzato il gestore code predefinito.
- Solo un gestore code alla volta può partecipare a una transazione coordinata da un'istanza di un coordinatore del punto di sincronizzazione esterno. Il coordinatore del punto di sincronizzazione è effettivamente connesso al gestore code ed è soggetto alla regola per cui è supportata una sola connessione alla volta.
- Tutte le applicazioni che includono chiamate a un coordinatore del punto di sincronizzazione esterno possono connettersi solo al gestore code che partecipa alla transazione gestita dal coordinatore esterno (poiché sono già effettivamente connesse a tale gestore code). Tuttavia, tali applicazioni devono emettere una chiamata `MQCONN` per ottenere un handle di connessione e una chiamata `MQDISC` prima di uscire.
- Un gestore code con aggiornamenti delle risorse coordinati da un coordinatore del punto di sincronizzazione esterno deve essere avviato prima del coordinatore del punto di sincronizzazione esterno. Allo stesso modo, il coordinatore del punto di sincronizzazione deve terminare prima del gestore code.
- Se il coordinatore del punto di sincronizzazione esterno viene terminato in modo anomalo, arrestare e riavviare il gestore code ***prima di*** riavviare il coordinatore del punto di sincronizzazione per garantire che tutte le operazioni di messaggistica non sottoposte a commit al momento dell'errore vengano risolte correttamente.

Utilizzo di CICS

CICS è uno degli elementi di TXSeries.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Le versioni di TXSeries compatibili con XA (e che utilizzano un processo di commit a due fasi) sono definite all'indirizzo: IBM WebSphere MQ requisiti di sistema dettagliati

WebSphere MQ supporta anche altri gestori transazioni. Consultare IBM WebSphere MQ requisiti di sistema dettagliati per gli elenchi correnti di software supportati.

Requisiti del processo di commit a due fasi
Requisiti del processo di commit a doppia fase quando si utilizza il processo di commit a due fasi CICS con WebSphere MQ. Questi requisiti non si applicano a z/OS.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

tenere presenti i seguenti requisiti:

- WebSphere MQ e CICS devono risiedere sulla stessa macchina fisica.
- WebSphere MQ non supporta CICS su un client WebSphere MQ MQI.
- È necessario avviare il gestore code, con il nome specificato nella sezione di definizione della risorsa XAD, ***prima*** di tentare di avviare CICS. La mancata esecuzione di questa operazione impedirà l'avvio di CICS se è stata aggiunta una stanza di definizione della risorsa XAD per WebSphere MQ alla regione CICS .
- È possibile accedere a un solo gestore code WebSphere MQ alla volta da una singola regione CICS .
- Una transazione CICS deve emettere una richiesta `MQCONN` prima di poter accedere alle risorse WebSphere MQ . La chiamata di `MQCONN` deve specificare il nome del gestore code WebSphere MQ specificato nella voce XAOpen della stanza di definizione della risorsa XAD per la regione CICS . Se questa voce è vuota, la richiesta `MQCONN` deve specificare il gestore code predefinito.
- Una transazione CICS che accede alle risorse WebSphere MQ deve emettere una chiamata `MQDISC` dalla transazione prima di tornare a CICS. L'errore potrebbe indicare che il server delle applicazioni CICS è ancora connesso, lasciando aperte le code. Inoltre, se non si installa un'uscita di terminazione attività (consultare “Uscita terminazione attività di esempio” a pagina 76), il server delle applicazioni CICS potrebbe in seguito terminare in modo anomalo, forse durante una transazione successiva.
- È necessario assicurarsi che l'ID utente CICS (cics) sia un membro del gruppo mqm, in modo che il codice CICS abbia l'autorizzazione a richiamare WebSphere MQ.

  Per le transazioni in esecuzione in un ambiente CICS , il Gestore code adatta i propri metodi di autorizzazione e di determinazione del contesto come segue:

  – Il gestore code interroga l'ID utente con cui CICS esegue la transazione. Questo è l'ID utente controllato da Object Authority Manager e viene utilizzato per le informazioni di contesto.
  – Nel contesto del messaggio, il tipo di applicazione è MQAT_CICS.
  – Il nome dell'applicazione nel contesto viene copiato dal nome della transazione CICS .

Supporto XA generale
XA generale non è supportato su IBM i. Viene fornito un modulo di caricamento switch XA per consentire di collegare CICS con WebSphere MQ su sistemi UNIX and Linux . Inoltre, vengono forniti dei file di codice sorgente di esempio per consentire lo sviluppo degli switch XA per altri messaggi di transazione.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

I nomi dei moduli di caricamento switch forniti sono:

Tabella 11. Codice essenziale per le applicazioni CICS : routine di inizializzazione XA	
C (origine)	**C (exec) - aggiungere uno dei seguenti XAD.Stanza**
amqzscix.c	amqzsc - TXSeries per AIX, Versione 5.1, amqzsc - TXSeries per HP-UX, Versione 5.1 amqzsc - TXSeries per Sun Solaris, Versione 5.1
amqzscin.c	mqmc4swi - TXSeries per Windows, Versione 5.1

Creazione di librerie da utilizzare con TXSeries for Multiplatforms
Utilizzare queste informazioni quando si creano le librerie da utilizzare con TXSeries for Multiplatforms.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

I *file di caricamento degli switch preintegrati* sono librerie condivise (denominate *DLL* sul sistema Windows) che è possibile utilizzare con programmi CICS, che richiedono una transazione di commit a 2 fasi utilizzando il protocollo XA. I nomi di queste librerie preintegrate si trovano nella tabella Codice essenziale per applicazioni CICS : routine di inizializzazione XA. Il codice sorgente di esempio viene fornito anche nelle seguenti directory:

Tabella 12. Directory di installazione su sistemi operativi Windows, UNIX and Linux		
Piattaforma	**Cartella**	**File di origine**
UNIX and Linux	*MQ_INSTALLATION_PATH*/ samp/	amqzscix.c
Windows	*MQ_INSTALLATION_PATH*\Tools \c \ Esempi	amqzscin.c

dove*MQ_INSTALLATION_PATH* è la directory in cui è stato installato IBM WebSphere MQ.

Per creare il file di caricamento switch dall'origine di esempio, seguire le istruzioni appropriate per il proprio sistema operativo:

AIX
Emetti il seguente comando:

```
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
Emetti il seguente comando:

```
/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

Emetti il seguente comando:

```
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
```

Piattaforme Linux

Emetti il seguente comando:

```
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

Eseguire queste operazioni:

1. Utilizzare il comando cl per creare amqzscin.obj compilando almeno le seguenti variabili:

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

2. Creare un file di definizione modulo denominato `mqmc1415.def`, che contiene le seguenti righe:

   ```
   LIBRARY MQMC4SWI
   EXPORTS
   CICS_XA_Init
   ```

3. Utilizzare il comando **lib** per creare un file di esportazione e una libreria di importazione utilizzando almeno la seguente opzione:

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

   Se il comando lib ha esito positivo, viene creato anche un file `mqmc4swi.exp` .

4. Utilizzare il comando di collegamento per creare `mqmc4swi.dll` utilizzando almeno la seguente opzione:

   ```
   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
   ```

Supporto IBM WebSphere MQ XA e Tuxedo

IBM WebSphere MQ in Windows, i sistemiUNIX and Linux possono bloccare indefinitamente le applicazioni XA coordinate da Tuxedo in xa_start.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Ciò può verificarsi solo quando due o più processi coordinati da Tuxedo in un singolo tentativo di transazione globale per accedere a IBM WebSphere MQ utilizzando lo stesso XID (transaction branch ID). Se Tuxedo fornisce a ciascun processo nella transazione globale un XID diverso da utilizzare con IBM WebSphere MQ, ciò non può verificarsi.

Per evitare il problema, configura ogni applicazione in Tuxedo che accede a IBM WebSphere MQ con un singolo ID transazione globale (gtrid), all'interno del suo gruppo di server Tuxedo. I processi nello stesso

gruppo di server utilizzano lo stesso XID durante l'accesso ai gestori risorse per conto di un singolo gtrid e sono pertanto vulnerabili al blocco in xa_start in IBM WebSphere MQ. I processi in gruppi di server differenti utilizzano XID separati quando accedono ai gestori risorse e quindi non devono serializzare il proprio lavoro di transazione in IBM WebSphere MQ.

Abilitazione del processo di commit a due fasi CICS
Per consentire a CICS di utilizzare un processo di commit a due fasi per coordinare transazioni che includono chiamate MQI, aggiungere una voce della stanza di definizione delle risorse XAD CICS alla region CICS . Notare che questo argomento non è applicabile a z/OS.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Di seguito è riportato un esempio di aggiunta di una voce di stanza XAD per WebSphere MQ per Windows, dove <Drive> è l'unità in cui è installato WebSphere MQ (ad esempio, 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>
```

Per i client transazionali estesi, utilizzare il file di caricamento switch mqcc4swi.dll.

Di seguito viene riportato un esempio di aggiunta di una voce della stanza XAD per i sistemi WebSphere MQ for UNIX and Linux , dove *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ :

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

Per i client transazionali estesi, utilizzare il file di caricamento switch amqczsc.

Per informazioni sull'utilizzo del comando **cicsadd** , consultare *CICS Administration Reference*o *CICS Administration Guide* per la propria piattaforma.

Le chiamate a WebSphere MQ possono essere incluse in una transazione CICS e verrà eseguito il commit o il rollback delle risorse WebSphere MQ , come indicato da CICS. Questo supporto non è disponibile per le applicazioni client.

È ***necessario*** emettere un MQCONN dalla transazione CICS per accedere alle risorse WebSphere MQ , seguite da un MQDISC corrispondente all'uscita.

Abilitazione delle uscite utente CICS
Un punto di uscita utente CICS (normalmente indicato come *uscita utente*) è un punto in un modulo di CICS in cui CICS può trasferire il controllo ad un programma scritto dall'utente (un *programma*di uscita utente) e in cui CICS può riprendere il controllo quando il programma di uscita ha terminato il suo lavoro.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Prima di utilizzare un'uscita utente CICS , leggere il manuale *CICS Administration Guide* per la propria piattaforma.

Uscita terminazione attività di esempio
WebSphere MQ fornisce un codice sorgente di esempio per un'uscita di terminazione attività CICS .

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per

andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Il codice sorgente di esempio si trova nelle directory seguenti:

Tabella 13. Uscite di terminazione attività CICS		
Piattaforma	**Cartella**	**File di origine**
Sistemi UNIX and Linux	*MQ_INSTALLATION_PATH*/samp	amqzscgx.c
Finestre	*MQ_INSTALLATION_PATH*\Tools \c \ Esempi	amqzscgn.c

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Le istruzioni di creazione per l'uscita di fine attività di esempio sono contenute nei commenti all'inizio di ciascun file di origine.

Questa uscita viene richiamata da CICS al termine del task normale e anomalo (dopo che è stato eseguito un punto di sincronizzazione). Non è consentito alcun lavoro recuperabile nel programma di uscita.

Queste funzioni vengono utilizzate solo in un contesto WebSphere MQ e CICS in cui la versione CICS supporta l'interfaccia XA. CICS fa riferimento a queste librerie come "programmi" o "uscite utente".

CICS ha un numero di uscite utente e amqzscgx, se utilizzato, è definito e abilitato in CICS come "Uscita utente di terminazione attività (UE014015)", ossia, uscita numero 15.

Quando l'uscita di terminazione del task viene richiamata da CICS, CICS ha già informato WebSphere MQ dello stato di terminazione del task e WebSphere MQ ha intrapreso l'azione appropriata (commit o rollback). L'uscita non fa che emettere un MQDISC per la ripulitura.

Uno scopo dell'installazione e della configurazione del proprio sistema CICS per utilizzare un'uscita di terminazione del task è proteggere il proprio sistema da alcune delle conseguenze del codice dell'applicazione malfunzionante. Ad esempio, se la propria transazione CICS termina in modo anomalo senza prima richiamare MQDISC e non è installata alcuna uscita di terminazione attività, è possibile che si verifichi (entro circa 10 secondi) un successivo errore irreversibile della regione CICS . Ciò è dovuto al fatto che il thread di integrità di WebSphere MQ, che viene eseguito nel processo cicsas, non sarà stato inviato e non avrà il tempo di eseguire la ripulitura e la restituzione. I sintomi potrebbero essere che il processo cicsas termina immediatamente, dopo aver scritto FFST riporta /var/mqm/errors o l'ubicazione equivalente in Windows.

Utilizzo di Microsoft Transaction Server (COM +)

COM + (Microsoft Transaction Server) è progettato per consentire agli utenti di eseguire applicazioni di logica aziendale in un tipico server di livello intermedio.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Per informazioni importanti, consultare Funzioni che possono essere utilizzate solo con l'installazione primaria su Windows .

COM + divide il lavoro in *attività*, che in genere sono brevi parti indipendenti della logica aziendale, come ad esempio *trasferire fondi dal conto A al conto B*. COM + si basa fortemente sull'orientamento degli oggetti e in particolare su COM; un'attività COM + è rappresentata da un oggetto COM (business).

COM + è una parte integrata del sistema operativo. Per utilizzare COM + in Windows 2000 e Windows XP, è necessario Hotfix Q313582 (noto anche come Pacchetto di rollup COM + 19.1).

COM + fornisce tre servizi all'amministratore dell'oggetto di business, eliminando gran parte delle preoccupazioni dal programmatore dell'oggetto di business:

- Gestione transazioni
- Sicurezza
- Pool di risorse

Di solito, si utilizza COM + con il codice front-end che è un client COM per gli oggetti contenuti in COM + e i servizi back-end come un database, con WebSphere MQ bridge tra l'oggetto di business COM + e il back-end.

Il codice front-end può essere un programma autonomo o un ASP (Active Server Page) ospitato da Microsoft Internet Information Server (IIS). Il codice front-end può essere sullo stesso computer di COM + e dei suoi oggetti di business, con connessione tramite COM. In alternativa, il codice front-end può essere su un computer diverso, con connessione tramite DCOM. È possibile utilizzare client diversi per accedere allo stesso business object COM + in situazioni diverse.

Il codice di back-end può trovarsi sullo stesso computer di COM + e dei relativi oggetti di business oppure su un computer differente con connessione tramite uno qualsiasi dei protocolli supportati da WebSphere MQ .

Unità di lavoro globali in scadenza

Il gestore code può essere configurato per far scadere le unità di lavoro globali dopo un intervallo di inattività preconfigurato.

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

Per abilitare questo comportamento, impostare le seguenti variabili di ambiente:

- *AMQ_TRANSACTION_EXPIRY_RESCAN*= < intervallo di riscansione in millisecondi>
- *AMQ_XA_TRANSACTION_SCADENZA*= < intervallo di timeout in millisecondi>

Attenzione: Le variabili di ambiente influiscono solo sulle transazioni che si trovano nello stato *Inattivo* nella tabella 6-4 della specifica XA. Vale a dire, le transazioni che non sono associate su alcun thread dell'applicazione, ma per le quali il software del gestore transazioni esterno non ha ancora richiamato la chiamata della funzione **xa_prepare** .

I gestori transazioni esterni conservano solo un log di transazioni preparate, sottoposte a commit o sottoposte a rollback. Se il gestore transazioni esterno si disattiva per un qualsiasi motivo, al suo ritorno, guida le transazioni preparate, di cui è stato eseguito il commit e di cui è stato eseguito il rollback fino al completamento, ma tutte le transazioni attive che devono ancora essere preparate diventano orfane. Per evitare ciò, impostare *AMQ_XA_TRANSACTION_SCADENZA* per consentire l'intervallo previsto tra un'applicazione che effettua chiamate API transazionali MQI e il completamento della transazione, dopo aver eseguito il lavoro transazionale su altri gestori risorse.

Per garantire una ripulitura tempestiva dopo la scadenza di *AMQ_XA_TRANSACTION_EXPIRY_RESCAN* , impostare il valore *AMQ_TRANSACTION_EXPIRY_RESCAN* su un valore inferiore rispetto all'intervallo *AMQ_XA_TRANSACTION_EXPIRE* , idealmente in modo che la riscansione si verifichi più di una volta all'interno dell'intervallo *AMQ_XA_TRANSACTION_EXPIRE* .

Disposizione unità di ripristino

WebSphere MQ for z/OS fornisce le disposizioni dell'unità di ripristino. Questa funzione consente di configurare se la seconda fase delle transazioni di commit a due fasi può essere gestita, ad esempio, durante il ripristino, quando si è connessi a un altro gestore code all'interno dello stesso QSG (queue - sharing group).

Nota: Questo argomento è disponibile anche in IBM MQ Version 8.0 e versioni successive. Tuttavia, non è possibile passare a una versione successiva utilizzando la casella di elenco "Modifica versione" . Per andare all'argomento in una versione successiva, modificare il numero di versione nella casella URL nel browser.

WebSphere MQ for z/OS V7.0.1 e versioni successive supporta l'unità di disposizione del ripristino.

Disposizione unità di ripristino
La disposizione dell'unità di ripristino è correlata alla connessione di un'applicazione e, successivamente, a tutte le transazioni che avvia. Esistono due possibili disposizioni di unità di recupero.

- Una disposizione di unità di recupero GROUP identifica che un'applicazione transazionale è connessa logicamente al gruppo di condivisione code e non ha un'affinità con alcun gestore code specifico. Qualsiasi transazione di commit a due fasi che viene avviata e che ha completato la phase-1 del processo di commit, ossia che è in dubbio, può essere interrogata e risolta, quando si è connessi a qualsiasi gestore code all'interno di QSG. In uno scenario di recupero, ciò significa che il coordinatore della transazione non deve riconnettersi allo stesso gestore code, che potrebbe non essere disponibile.
- Una disposizione di unità di ripristino QMGR identifica che un'applicazione ha un'affinità diretta con il gestore code a cui è connessa e che anche tutte le transazioni che avvia hanno questa disposizione.

  In uno scenario di recupero, il coordinatore della transazione deve riconnettersi allo stesso gestore code per interrogare e risolvere eventuali transazioni in dubbio, indipendentemente dal fatto che il gestore code appartenga o meno a un gruppo di condivisione code.

Scelta del linguaggio di programmazione da utilizzare

Utilizza queste informazioni per scoprire i linguaggi di programmazione e i framework supportati da IBM WebSphere MQe alcune considerazioni per utilizzarli.

IBM WebSphere MQ fornisce il supporto per i seguenti linguaggi procedurali di programmazione:

- C
- Visual Basic (solo sistemi Windows)
- COBOL

Queste lingue utilizzano l'interfaccia MQI (Message Queue Interface) per accedere ai servizi di accodamento messaggi. Per ulteriori informazioni sul supporto per queste lingue, consultare “Utilizzo dei linguaggi procedurali con WebSphere MQ” a pagina 79.

IBM WebSphere MQ fornisce supporto per:

- .NET
- ActiveX
- C++
- Java
- JMS

Questi linguaggi utilizzano IBM WebSphere MQ Object Model, che fornisce classi che forniscono la stessa funzionalità delle chiamate e delle strutture WebSphere MQ , ma che sono un modo più naturale di programmazione in un ambiente orientato agli oggetti. Alcuni linguaggi che utilizzano IBM WebSphere MQ Object Model forniscono funzioni aggiuntive che non sono disponibili nell'interfaccia MQI (Message Queue Interface). Per ulteriori informazioni sul supporto per queste lingue, consultare “Programmazione orientata agli oggetti con WebSphere MQ” a pagina 80.

Utilizzo dei linguaggi procedurali con WebSphere MQ

Per informazioni dettagliate su come scrivere le applicazioni nella lingua scelta, consultare i seguenti link:

- “Codifica in C” a pagina 83
- “Codifica in Visual Basic” a pagina 87
- “Codifica in COBOL” a pagina 86

Per una panoramica dell'interfaccia di chiamata per le lingue procedurali, consultare Descrizioni delle chiamate. Questo argomento contiene un elenco delle chiamate MQI e ciascuna chiamata mostra come codificare le chiamate in ognuna di queste lingue.

WebSphere MQ fornisce i file di definizione dei dati che consentono di scrivere le applicazioni. Per una descrizione completa, consultare "File di definizione dati IBM WebSphere MQ" a pagina 81.

Se è possibile scegliere la lingua in cui codificare i programmi, considerare la lunghezza massima dei messaggi che i programmi elaboreranno. Se i programmi elaboreranno solo messaggi di lunghezza massima nota, è possibile codificarli in uno qualsiasi dei linguaggi di programmazione supportati. Ma se non si conosce la lunghezza massima dei messaggi che i programmi dovranno elaborare, la lingua scelta dipenderà dalla scrittura di un'applicazione CICS, IMSo batch:

IMS e batch
Codificare i programmi in linguaggio C, PL/I o assembler per utilizzare le funzioni offerte da questi linguaggi per ottenere e rilasciare quantità arbitrarie di memoria. In alternativa, è possibile codificare i programmi in COBOL, ma utilizzare il linguaggio assembler, le sottoroutine PL/I o C per ottenere e rilasciare la memoria.

CICS
Codificare i programmi in qualsiasi lingua supportata da CICS. L'interfaccia EXEC CICS fornisce le chiamate per la gestione della memoria, se necessario.

Programmazione orientata agli oggetti con WebSphere MQ

Alcuni linguaggi e framework di programmazione che utilizzano IBM WebSphere MQ Object Model forniscono funzioni aggiuntive che non sono disponibili nell'interfaccia MQI (message queue interface). Per i dettagli delle classi, dei metodi e delle proprietà forniti da IBM WebSphere MQ Object Model, consultare "Il modello oggetto IBM WebSphere MQ" a pagina 88.

.NET
Consultare Utilizzo di .NET per informazioni sulla codifica dei programmi .NET utilizzando le WebSphere MQ .NET. I Message Service Clients for C/C++ and .NET forniscono un'API (Application Programming Interface) denominata XMS che ha la stessa serie di interfacce di JMS (Java Message Service) API.

C++
IBM WebSphere MQ fornisce classi C+ + equivalenti agli oggetti WebSphere MQ e alcune classi aggiuntive equivalenti ai tipi di dati array. Fornisce un certo numero di funzioni non disponibili tramite MQI. Fare riferimento a Utilizzo di C++ per informazioni sulla codifica dei programmi utilizzando WebSphere MQ Object Model in C + +. Message Service Clients for C/C++ and .NET fornire un'API (application programming interface) denominata XMS che ha la stessa serie di interfacce di Java Message Service (JMS) API.

Java
Consultare Utilizzo di Java per informazioni sulla codificazione dei programmi mediante WebSphere MQ Object Model in Java. Per informazioni sulle differenze tra le classi IBM WebSphere MQ per Java e IBM WebSphere MQ per decidere quali utilizzare, consultare "Utilizzare le classi IBM WebSphere MQ per Java o IBM WebSphere MQ per JMS?" a pagina 90.

JMS
Websphere MQ fornisce anche classi che implementano la specifica JMS (Java Message Service). Per i dettagli delle classi MQ di Websphere per JMS, consultare Utilizzo di Java. Per informazioni sulle differenze tra le classi IBM WebSphere MQ per Java e IBM WebSphere MQ per decidere quale utilizzare, consultare "Utilizzare le classi IBM WebSphere MQ per Java o IBM WebSphere MQ per JMS?" a pagina 90.

I Message Service Clients for C/C++ and .NET forniscono un'API (Application Programming Interface) denominata XMS che ha la stessa serie di interfacce di JMS (Java Message Service) API.

ActiveX
WebSphere MQ ActiveX è comunemente noto come MQAX. MQAX è incluso come parte di WebSphere MQ per Windows. Il supporto per ActiveX è stato stabilizzato al livello WebSphere MQ Versione 6.0 . Per utilizzare le funzioni introdotte in WebSphere MQ successive alla Versione 6.0, utilizzare .NET.

Fare riferimento a Utilizzo dell'interfaccia COM (Component Object Model) (classi di automazione di WebSphere MQ per ActiveX) per informazioni sulla codifica dei programmi utilizzando WebSphere MQ Object Model in ActiveX.

Concetti correlati

Panoramica tecnica

“Sviluppo delle applicazioni” a pagina 7
IBM WebSphere MQ fornisce diversi modi in cui è possibile sviluppare applicazioni per inviare e ricevere messaggi necessari per supportare i processi aziendali. È anche possibile sviluppare applicazioni per gestire i gestori code e le risorse correlate.

“Concetti dello sviluppo di applicazioni” a pagina 8
È possibile utilizzare una scelta di linguaggi procedurali o orientati agli oggetti per scrivere applicazioni IBM WebSphere MQ . Utilizzare i link in questo argomento per informazioni sui concetti di IBM WebSphere MQ che sono utili per sviluppatori di applicazioni.

Riferimenti correlati

Riferimento sviluppo applicazione

File di definizione dati IBM WebSphere MQ

IBM WebSphere MQ fornisce file di definizione dei dati che consentono di scrivere le applicazioni.

I file di definizione dati sono noti anche come:

Lingua	Definizioni di dati
C	Includi file o file di intestazione
Visual Basic	File di modulo (solo versioni a 32 bit)
COBOL	Copia file
Assembler	Macro
PL/I	Includi file

I file di definizione dei dati che consentono di scrivere le uscite del canale sono descritti in WebSphere MQ COPY, header, include e module files.

I file di definizione dei dati che consentono di scrivere le uscite dei servizi installabili sono descritti in “Uscite utente, uscite API e servizi installabili WebSphere MQ” a pagina 376.

Per i file di definizione dati supportati in C + +, consultare Utilizzo di C++.

I nomi dei file di definizione dati hanno il prefisso CMQ e un suffisso determinato dal linguaggio di programmazione:

Suffisso	Lingua
a	Linguaggio assembler
b	Visual Basic
c	C
l	COBOL (senza valori inizializzati)
p	PL/I
v	COBOL (con valori predefiniti impostati)

Libreria di installazione

Il nome **thlqual** è il qualificatore di alto livello della libreria di installazione su z/OS.

Questo argomento introduce i file di definizione dei dati WebSphere MQ , sotto queste intestazioni:

- “File di inclusione linguaggio C” a pagina 82
- “File del modulo Visual Basic” a pagina 82
- “File di copia COBOL” a pagina 82

File di inclusione linguaggio C

I file di inclusione di WebSphere MQ C sono elencati in File di intestazione C. Sono installati nelle seguenti directory o librerie:

Piattaforma	Directory di installazione o libreria
Piattaforme UNIX	*MQ_INSTALLATION_PATH*/inc/
Sistemi Windows	*MQ_INSTALLATION_PATH*\Tools\c\includi

dove *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Nota: Per piattaforme UNIX , i file di inclusione sono simbolicamente collegati in `/usr/include`.

Per ulteriori informazioni sulla struttura delle directory, consultare Pianificazione del supporto del file system .

File del modulo Visual Basic

WebSphere MQ per Windows fornisce quattro file del modulo Visual Basic.

Sono elencati in File del modulo Visual Basic e installati in

```
MQ_INSTALLATION_PATH\Tools\Samples\VB\Include
```

File di copia COBOL

Per COBOL, WebSphere MQ fornisce file di copia separati contenenti le costanti denominate e due file di copia per ognuna delle strutture.

Ci sono due file di copia per ogni struttura perché ognuno è fornito sia con che senza valori iniziali:

- Nella WORKING - STORAGE SECTION di un programma COBOL, utilizzare i file che inizializzano i campi di struttura sui valori predefiniti. Queste strutture sono definite nei file di copia che hanno nomi con suffisso la lettera V (valori).
- Nella LINKAGE SECTION di un programma COBOL, utilizzare le strutture senza valori iniziali. Queste strutture sono definite in file di copia i cui nomi hanno come suffisso la lettera L (collegamento).

I file di copia WebSphere MQ COBOL sono elencati in File COBOL COPY. Sono installati nelle directory seguenti:

Piattaforma	Directory di installazione o libreria
Altre piattaforme UNIX	*MQ_INSTALLATION_PATH*/inc/
Finestre	*MQ_INSTALLATION_PATH*\Tools\cobol\copybook (per Micro Focus COBOL) *MQ_INSTALLATION_PATH*\Tools\cobol\copybook\VAcobol (per IBM VisualAge COBOL)

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Includere nel programma solo quei file necessari. Eseguire questa operazione con una o più istruzioni COPY dopo una dichiarazione level-01 . Ciò significa che è possibile includere più versioni delle strutture in un programma, se necessario. Notare che CMQV è un file di grandi dimensioni.

Di seguito è riportato un esempio di codice COBOL per includere il file di copia CMQMDV:

```
01 MQM-MESSAGE-DESCRIPTOR.
   COPY CMQMDV.
```

Ogni dichiarazione di struttura inizia con un elemento level-01 ; è possibile dichiarare diverse istanze della struttura codificando la dichiarazione level-01 seguita da un'istruzione COPY da copiare nel resto della dichiarazione di struttura. Per fare riferimento all'istanza appropriata, utilizzare la parola chiave IN.

Di seguito è riportato un esempio di codice COBOL per includere due istanze di 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.
```

Allineare le strutture sui limiti a 4 byte. Se si utilizza l'istruzione COPY per includere una struttura che segue un elemento che non è l'elemento level-01 , verificare che la struttura sia un multiplo di 4 byte dall'inizio dell'elemento level-01 . Se non si esegue questa operazione, è possibile ridurre le prestazioni dell'applicazione.

Le strutture sono descritte in Tipi di dati utilizzati in MQI. Le descrizioni dei campi nelle strutture mostrano i nomi dei campi senza prefisso. Nei programmi COBOL, anteporre ai nomi dei campi il nome della struttura seguito da un trattino, come mostrato nelle dichiarazioni COBOL. I campi nei file di copia della struttura hanno come prefisso questo modo.

I nomi campo nelle dichiarazioni nei file di copia della struttura sono in maiuscolo. È possibile utilizzare caratteri minuscoli o maiuscoli. Ad esempio, il campo *StrucId* della struttura MQGMO viene visualizzato come MQGMO - STRUCID nella dichiarazione COBOL e nel file di copia.

Le strutture con suffisso V vengono dichiarate con i valori iniziali per tutti i campi, quindi è necessario impostare solo quei campi in cui il valore richiesto è diverso dal valore iniziale.

Codifica in C

Prendere nota delle informazioni riportate nelle sezioni seguenti quando si codificano i programmi WebSphere MQ in C.

Parametri delle chiamate MQI

I parametri che sono *solo input* e di tipo MQHCONN, MQHOBJ, MQHMSG o MQLONG vengono passati per valore; per tutti gli altri parametri, l' *indirizzo* del parametro viene passato per valore.

Non tutti i parametri che vengono passati per indirizzo devono essere specificati ogni volta che viene richiamata una funzione. Quando un particolare parametro non è richiesto, è possibile specificare un puntatore null come parametro sul richiamo della funzione, al posto dell'indirizzo dei dati del parametro. I parametri per i quali ciò è possibile sono identificati nelle descrizioni delle chiamate.

Non viene restituito alcun parametro come valore della funzione; nella terminologia C, ciò significa che tutte le funzioni restituiscono void.

Gli attributi della funzione sono definiti dalla variabile macro MQENTRY; il valore di questa variabile macro dipende dall'ambiente.

Parametri con tipo di dati non definito

Le funzioni MQGET, MQPUT e MQPUT1 hanno ognuna un parametro *Buffer* che ha un tipo di dati non definito. Questo parametro viene utilizzato per inviare e ricevere i dati del messaggio dell'applicazione.

I parametri di questo tipo vengono mostrati negli esempi C come array di MQBYTE. È possibile dichiarare i parametri in questo modo, ma è più conveniente dichiararli come la struttura che descrive il layout dei dati nel messaggio. Il parametro della funzione viene dichiarato come un puntatore a void e quindi l'indirizzo di qualsiasi dato può essere specificato come parametro nel richiamo della funzione.

Tipi di dati

Tutti i tipi di dati sono definiti con l'istruzione `typedef` .

Per ogni tipo di dati, viene definito anche il tipo di dati puntatore corrispondente. Il nome del tipo di dati del puntatore è il nome del tipo di dati elementari o della struttura preceduto dalla lettera P per indicare un puntatore. Gli attributi del puntatore sono definiti dalla variabile macro MQPOINTER; il valore di questa variabile macro dipende dall'ambiente. Il seguente codice illustra come dichiarare i tipi di dati del puntatore:

```
#define MQPOINTER                  /* depends on environment */
...
typedef MQLONG  MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;     /* pointer to MQMD */
```

Manipolazione di stringhe binarie

Le stringhe di dati binari vengono dichiarate come uno dei tipi di dati MQBYTEn.

Ogni volta che si copiano, confrontano o impostano campi di questo tipo, utilizzare le funzioni C `memcpy`, `memcmpo` `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));
```

Non utilizzare le funzioni di stringa strcpy, strcmp, strncpy o strncmp perché non funzionano correttamente con i dati dichiarati come MQBYTE24.

Manipolazione delle stringhe di caratteri

Quando il gestore code restituisce i dati carattere all'applicazione, il gestore code riempierà sempre i dati carattere con spazi vuoti fino alla lunghezza definita del campo. Il gestore code non restituisce stringhe con terminazione null, ma è possibile utilizzarle nell'input. Pertanto, durante la copia, il confronto o la concatenazione di tali stringhe, utilizzare le funzioni stringa strncpy, strncmp o strncat.

Non utilizzare le funzioni stringa che richiedono che la stringa termini con un valore null (strcpy, strcmp e strcat). Inoltre, non utilizzare la funzione strlen per determinare la lunghezza della stringa; utilizzare invece la funzione sizeof per determinare la lunghezza del campo.

Valori iniziali per le strutture

Il file di inclusione <cmqc.h> definisce varie variabili macro che è possibile utilizzare per fornire valori iniziali per le strutture quando si dichiarano istanze di tali strutture. Queste variabili macro hanno i nomi nel formato MQxxx_DEFAULT, dove MQxxx rappresenta il nome della struttura. Usali come questo:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

Per alcuni campi di caratteri, l'MQI definisce valori particolari che sono validi (ad esempio, per i campi *StrucId* o per il campo *Format* in MQMD). Per ognuno dei valori validi, vengono fornite due variabili macro:

- Una variabile macro definisce il valore come una stringa con una lunghezza, escluso il valore null implicito, che corrisponde esattamente alla lunghezza definita del campo. Ad esempio, il simbolo ¬ rappresenta un carattere vuoto:

  ```
  #define MQMD_STRUC_ID "MD¬¬"
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Utilizzare questo modulo con le funzioni memcpy e memcmp.
- L'altra variabile macro definisce il valore come un array di caratteri; il nome di questa variabile macro è il nome del formato stringa con suffisso _ARRAY. Ad esempio:

  ```
  #define MQMD_STRUC_ID_ARRAY 'M','D','¬','¬'
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Utilizzare questo modulo per inizializzare il campo quando un'istanza della struttura viene dichiarata con valori diversi da quelli forniti dalla variabile macro MQMD_DEFAULT.

Valori iniziali per le strutture dinamiche

Quando è richiesto un numero variabile di istanze di una struttura, le istanze vengono generalmente create nella memoria principale ottenuta dinamicamente utilizzando le funzioni calloc o malloc.

Per inizializzare i campi in tali strutture, si raccomanda la seguente tecnica:

1. Dichiarare un'istanza della struttura utilizzando la variabile macro MQxxx_DEFAULT appropriata per inizializzare la struttura. Questa istanza diventa il *modello* per altre istanze:

   ```
   MQMD ModelMsgDesc = {MQMD_DEFAULT};
                                    /* declare model instance */
   ```

   Codificare le parole chiave statiche o automatiche sulla dichiarazione per fornire la durata statica o dinamica dell'istanza del modello, come richiesto.
2. Utilizzare le funzioni calloc o malloc per ottenere lo storage per un'istanza dinamica della struttura:

   ```
   PMQMD InstancePtr;
   InstancePtr = malloc(sizeof(MQMD));
                                    /* get storage for dynamic instance */
   ```
3. Utilizzare la funzione memcpy per copiare l'istanza del modello nell'istanza dinamica:

   ```
   memcpy(InstancePtr,&ModelMsgDesc,sizeof(MQMD));
                                    /* initialize dynamic instance */
   ```

Utilizza da C++

Per il linguaggio di programmazione C + +, i file di intestazione contengono le seguenti istruzioni aggiuntive incluse solo quando viene utilizzato il compilatore C + +:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Codifica in COBOL

Prendere nota delle informazioni riportate nella seguente sezione quando si codificano i programmi WebSphere MQ in COBOL.

Costanti con nome

I nomi delle costanti vengono visualizzati contenenti il carattere di sottolineatura (_) come parte del nome. In COBOL, è necessario utilizzare il trattino (-) al posto del carattere di sottolineatura. Le costanti che hanno valori stringa di caratteri utilizzano il carattere virgolette singole (') come delimitatore di stringa. Per far sì che il compilatore accetti questo carattere, utilizzare l'opzione APOST del compilatore.

Il file di copia CMQV contiene le dichiarazioni delle costanti denominate come voci level-10 . Per utilizzare le costanti, dichiarare esplicitamente l'elemento level-01 , quindi utilizzare l'istruzione COPY per copiare le dichiarazioni delle costanti:

```
 WORKING-STORAGE SECTION.
 01  MQM-CONSTANTS.
     COPY CMQV.
```

Tuttavia, questo metodo fa sì che le costanti occupino la memoria nel programma anche se non vi si fa riferimento. Se le costanti sono incluse in molti programmi separati all'interno della stessa unità di esecuzione, esisteranno più copie delle costanti; ciò potrebbe comportare l'utilizzo di una quantità significativa di memoria principale. È possibile evitare ciò aggiungendo la clausola GLOBAL alla dichiarazione level-01 :

```
* Declare a global structure to hold the constants
 01  MQM-CONSTANTS GLOBAL.
     COPY CMQV.
```

Questo assegna la memoria solo per *una* serie di costanti all'interno dell'unità di esecuzione; le costanti, tuttavia, possono essere indicate da *qualsiasi* programma all'interno dell'unità di esecuzione, non solo dal programma che contiene la dichiarazione level-01 .

Garantire l'allineamento della struttura

È necessario assicurarsi che le strutture IBM WebSphere MQ trasmesse per l'avvio sulla chiamata MQ siano allineate sui limiti delle parole. Un limite di parole è di 4 byte per i processi a 32 bit, 8 byte per quelli a 64 bit e 16 byte per i processi a 128 bit (IBM i).

Dove possibile, posizionare tutte le strutture IBM WebSphere MQ insieme in modo che siano tutte allineate ai limiti.

Codifica in pTAL

Prendere nota delle informazioni riportate nella seguente sezione quando si codificano programmi IBM WebSphere MQ in pTAL.

HP Integrity NonStop Server

Definizione e inizializzazione di strutture IBM WebSphere MQ

Le definizioni di struttura pTAL per le strutture IBM WebSphere MQ vengono fornite con nomi che terminano con ^DEF. Ad esempio, le seguenti dichiarazioni pTAL vengono codificate per creare una struttura MQMD (IBM WebSphere MQ Message Descriptor) e una struttura MQPMO (IBM WebSphere MQ Put Message Options).

```
STRUCT MYMD(MQMD^DEF);        ! Declare an MQMD structure
STRUCT MYPMO(MQPMO^DEF);      ! Declare an MQPMO structure
```

IBM WebSphere MQ fornisce pTAL DEFINE con nomi che terminano con ^DEFAULT per inizializzare le strutture IBM WebSphere MQ con valori predefiniti. Le seguenti istruzioni pTAL sono codificate per assegnare valori predefiniti alle strutture MQMD e MQPMO dichiarate:

```
MQMD^DEFAULT(MYMD);           ! Assign default values to an MQMD structure
MQPMO^DEFAULT(MYPMO);         ! Assign default values to an MQPMO structure
```

È possibile dichiarare e inizializzare altre strutture IBM WebSphere MQ utilizzando codice simile.

pTAL e CRE

I programmi pTAL non possono inizializzare Common Runtime Environment e pertanto devono essere utilizzati con una routine principale in linguaggio C o COBOL.

Gli esempi pTAL forniti con IBM WebSphere MQ utilizzano una routine mainline in linguaggio C denominata AMQSPTM0.C

Parametri con tipo di dati MQCHAR

Le procedure MQGET, MQPUT e MQPUT1 hanno ognuna un parametro **Buffer** con un tipo di dati MQCHAR .EXT . Questo parametro viene utilizzato per inviare e ricevere i dati del messaggio dell'applicazione.

I parametri di questo ordinamento vengono mostrati negli esempi pTAL come array di stringhe. È possibile dichiarare i parametri in questo modo, ma è più conveniente dichiararli come la struttura che descrive il layout dei dati nel messaggio. Il parametro della procedura è dichiarato come MQCHAR .EXT, ma l'indirizzo di qualsiasi dato può essere specificato come parametro nel richiamo della procedura.

Manipolazione delle stringhe di caratteri

Quando il gestore code restituisce i dati carattere all'applicazione, il gestore code riempierà sempre i dati carattere con spazi vuoti fino alla lunghezza definita del campo. Il gestore code non restituisce stringhe con terminazione null, ma è possibile utilizzarle nell'input.

Codifica in Visual Basic

Prendere nota delle informazioni contenute nella seguente sezione quando si codificano i programmi WebSphere MQ in Visual Basic.

Nota: All'esterno dell'ambiente .NET, il supporto per Visual Basic (VB) in WebSphere MQ è stato stabilizzato al livello V6.0 . La maggior parte delle nuove funzioni aggiunte a WebSphere MQ 7.0 o versioni successive non è disponibile per le applicazioni VB. Se si sta programmando in VB.NET, utilizzare le classi WebSphere MQ .NET. Per ulteriori informazioni, consultare Utilizzo di .NET.

Visual Basic è supportato solo su Windows.

Per evitare la conversione non intenzionale dei dati binari tra Visual Basic e WebSphere MQ, utilizzare una definizione di MQBYTE invece di MQSTRING. CMQB.BAS definisce diversi nuovi tipi MQBYTE che sono equivalenti a una definizione di byte C e li utilizza all'interno delle strutture WebSphere MQ . Ad esempio, per la struttura MQMD (message descriptor), MsgId (message identifier) è definito come MQBYTE24.

Visual Basic non dispone di un tipo di dati puntatore, per cui i riferimenti ad altre strutture dati WebSphere MQ sono per offset piuttosto che per puntatore. Dichiarare una struttura composta dalle due strutture componenti e specificare la struttura composta nella chiamata. WebSphere Il supporto MQ per Visual Basic fornisce una chiamata MQCONNXAny per rendere ciò possibile e consentire alle applicazioni client di specificare le proprietà del canale su una connessione client. Accetta una struttura non tipizzata (MQCNOCD) al posto della tipica struttura MQCNO.

La struttura MQCNOCD è una struttura composta da un MQCNO seguito da un MQCD. Questa struttura viene dichiarata nel file di intestazione delle uscite CMQXB. Utilizzare la routine MQCNOCD_DEFAULTS per inizializzare una struttura MQCNOCD. Viene fornito un esempio di chiamate MQCONNX (amqscnxb.vbp).

MQCONNXAny ha gli stessi parametri di MQCONNX, ad eccezione del fatto che il parametro *ConnectOpts* è dichiarato come qualsiasi tipo di dati anziché di tipo MQCNO. Ciò consente alla funzione di accettare la struttura MQCNO o MQCNOCD. Questa funzione viene dichiarata nel file di intestazione principale CMQB.

Il modello oggetto IBM WebSphere MQ

IBM WebSphere MQ Object Model è composto da classi, metodi e proprietà. Utilizzare queste informazioni per informazioni su ciascuno di tali concetti.

Il modello oggetto IBM WebSphere MQ è composto da:

- *Classi* che rappresentano concetti familiari di WebSphere MQ come gestori code, code e messaggi.
- *Metodi* su ciascuna classe corrispondente a chiamate MQI.
- *Proprietà* su ciascuna classe corrispondente agli attributi degli oggetti WebSphere MQ .

Quando si crea un'applicazione WebSphere MQ utilizzando WebSphere MQ Object Model, si creano le istanze di queste classi nel programma. Un'istanza di una classe nella programmazione orientata agli oggetti viene denominata *oggetto*. Quando un oggetto viene creato, si interagisce con l'oggetto esaminando o impostando i valori delle proprietà dell'oggetto (l'equivalente dell'emissione di una chiamata MQINQ o MQSET) e effettuando chiamate di metodo sull'oggetto (l'equivalente dell'emissione di altre chiamate MQI).

Questi argomenti descrivono in dettaglio ciascuno dei modelli oggetto WebSphere MQ :

- “Classi” a pagina 88
- “riferimenti a oggetti” a pagina 89
- “Codici di ritorno” a pagina 89

Classi

Il modello oggetto WebSphere MQ fornisce la seguente serie di classi di base.

L'effettiva implementazione del modello varia leggermente tra i diversi ambienti orientati agli oggetti supportati.

MQQueueManager
Un oggetto della classe MQQueueManager rappresenta una connessione a un gestore code. Dispone di metodi per Connect (), Disconnect (), Commit () e Backout () (l'equivalente di MQCONN o MQCONNX, MQDISC, MQCMIT e MQBACK). Dispone di proprietà corrispondenti agli attributi di un gestore code. L'accesso a una proprietà dell'attributo del gestore code si connette implicitamente al gestore code, se non è già connesso. L'eliminazione di un oggetto MQQueueManager si disconnette implicitamente dal gestore code.

MQQUEUE
Un oggetto della classe MQQueue rappresenta una coda. Dispone di metodi per inserire () e richiamare () i messaggi da e verso la coda (l'equivalente di MQPUT e MQGET). Ha proprietà corrispondenti agli attributi di una coda. L'accesso a una proprietà dell'attributo della coda o l'emissione di una chiamata al metodo Put () o Get () apre implicitamente la coda (l'equivalente di MQOPEN). L'eliminazione di un oggetto MQQueue chiude implicitamente la coda (l'equivalente di MQCLOSE).

MQArgomento
Un oggetto della classe MQTopic rappresenta un argomento. Dispone di metodi per inserire () (pubblicare) e Get () (ricevere o sottoscrivere) i messaggi da e verso l'argomento (l'equivalente di MQPUT e MQGET). Ha proprietà corrispondenti agli attributi di un argomento. È possibile accedere a un oggetto MQTopic solo per la pubblicazione o la sottoscrizione, non contemporaneamente. Quando viene utilizzato per ricevere i messaggi, l'oggetto MQTopic può essere creato con una sottoscrizione non gestita o gestita e come sottoscrittore durevole o non durevole - vengono forniti più costruttori sovraccaricati per questi diversi scenari.

Messaggio MQT
Un oggetto della classe MQMessage rappresenta un messaggio da inserire in una coda o da ottenere da una coda. Contiene un buffer e incapsula sia i dati dell'applicazione che MQMD. Dispone di proprietà corrispondenti ai campi MQMD e ai metodi che consentono di scrivere e leggere i dati utente di diversi tipi (ad esempio, stringhe, numeri interi lunghi, numeri interi brevi, byte singoli) nel e dal buffer.

Opzioni MQPutMessage
Un oggetto della classe Opzioni MQPutMessagerappresenta la struttura MQPMO. Ha proprietà corrispondenti ai campi MQPMO.

Opzioni MQGetMessage
Un oggetto della classe Opzioni MQGetMessagerappresenta la struttura MQGMO. Ha proprietà corrispondenti ai campi MQGMO.

Processo MQ
Un oggetto della classe MQProcess rappresenta una definizione di processo (utilizzata con il trigger). Dispone di proprietà che rappresentano gli attributi di una definizione di processo.

MQDistributionList
Un oggetto della classe MQDistributionList rappresenta un elenco di distribuzione (utilizzato per inviare più messaggi con un singolo MQPUT). Contiene un elenco di oggetti MQDistributionListItem.

MQDistributionListElemento
Un oggetto della classe di voci MQDistributionListrappresenta una singola destinazione dell'elenco di distribuzione. Comprende le strutture MQOR, MQRR e MQPMR e dispone di proprietà corrispondenti ai campi di tali strutture.

riferimenti a oggetti

In un programma WebSphere MQ che utilizza MQI, WebSphere MQ restituisce gli handle di connessione e gli handle di oggetto al programma.

Questi handle devono essere passati come parametri nelle successive chiamate WebSphere MQ . Con WebSphere MQ Object Model, questi handle sono nascosti dal programma di applicazione. Invece, la creazione di un oggetto da una classe determina la restituzione di un riferimento oggetto al programma applicativo. È questo riferimento oggetto che viene utilizzato quando si effettuano chiamate di metodo e accessi di proprietà rispetto all'oggetto.

Codici di ritorno

L'emissione di una chiamata di metodo o l'impostazione di un valore di proprietà comporta l'impostazione di codici di ritorno.

Questi codici di ritorno sono un codice di completamento e un codice motivo e sono essi stessi proprietà dell'oggetto. I valori del codice di completamento e del codice motivo sono gli stessi definiti per MQI, con alcuni valori aggiuntivi specifici dell'ambiente orientato agli oggetti.

Utilizzare le classi IBM WebSphere MQ per Java o IBM WebSphere MQ per JMS?

Un'applicazione Java può utilizzare le classi IBM WebSphere MQ per Java o IBM WebSphere MQ per JMS per accedere alle risorse IBM WebSphere MQ . Ogni approccio ha i suoi vantaggi.

Le classi IBM WebSphere MQ per Java incapsulano l'interfaccia MQI (Message Queue Interface), l'API IBM WebSphere MQ nativa, e utilizzano lo stesso modello oggetto di altre interfacce orientate agli oggetti, mentre le classi IBM WebSphere MQ per Java Message Service implementano le interfacce JMS (Java Message Service) di Sun.

Se si ha familiarità con IBM WebSphere MQ in ambienti diversi da Java, utilizzando linguaggi procedurali o orientati agli oggetti, è possibile trasferire le proprie conoscenze esistenti all'ambiente Java utilizzando le classi IBM WebSphere MQ per Java. È anche possibile sfruttare l'intera gamma di funzioni di IBM WebSphere MQ, non tutte disponibili nelle classi IBM WebSphere MQ per JMS.

Se non si ha familiarità con IBM WebSphere MQo si ha già esperienza JMS, potrebbe essere più semplice utilizzare l'API JMS familiare per accedere alle risorse IBM WebSphere MQ , utilizzando le classi IBM WebSphere MQ per JMS. JMS è anche parte integrante della piattaforma Java Platform, Enterprise Edition (Java EE). Le applicazioni Java EE possono utilizzare MDB (message - driven bean) per elaborare i messaggi in modo asincrono e gli MDB possono elaborare solo i messaggi JMS. JMS è anche il meccanismo standard per Java EE per interagire con i sistemi di messaggistica asincrona come IBM WebSphere MQ. Ogni server delle applicazioni che è conforme a Java EE deve includere un provider JMS, pertanto è possibile utilizzare JMS per comunicare tra diversi server delle applicazioni oppure è possibile trasferire un'applicazione da un provider JMS a un altro senza alcuna modifica all'applicazione.

Progettazione di applicazioni IBM WebSphere MQ

Una volta stabilito il modo in cui le applicazioni possono sfruttare le piattaforme e gli ambienti disponibili, è necessario decidere come utilizzare le funzioni offerte da WebSphere MQ.

Quando si progetta un'applicazione IBM WebSphere MQ , considerare le seguenti domande e opzioni:

Tipo di applicazione

Qual è lo scopo della tua applicazione? Consultare i seguenti link per informazioni sui diversi tipi di applicazione che è possibile sviluppare:

- Server
- Client
- Pubblicazione/sottoscrizione
- Servizi web
- Uscite utente, uscite API e servizi installabili

Inoltre, è anche possibile scrivere le applicazioni per automatizzare la gestione di IBM WebSphere MQ. Per ulteriori informazioni, consultare Introduzione a WebSphere MQ Administration Interface (MQAI) e Automating administration tasks .

Linguaggio programmazione

IBM WebSphere MQ supporta diversi linguaggi di programmazione procedurali e orientati agli oggetti per la scrittura di applicazioni. Per ulteriori informazioni, consultare “Scelta del linguaggio di programmazione da utilizzare” a pagina 79.

Applicazioni per più di una piattaforma

L'applicazione verrà eseguita su più di una piattaforma? Hai una strategia per passare a una piattaforma diversa da quella che usi oggi? Se la risposta a una di queste domande è sì, assicurarsi di codificare i programmi per l'indipendenza della piattaforma.

Se si utilizza C, codificare nello standard ANSI C. Utilizzare una funzione della libreria C standard piuttosto che una funzione specifica della piattaforma equivalente, anche se la funzione specifica della piattaforma è più veloce o più efficiente. L'eccezione è quando l'efficienza nel codice è fondamentale, quando è necessario codificare per entrambe le situazioni utilizzando #ifdef. Ad esempio:

```
#ifdef _AIX
     AIX specific code
#else
     generic code
#endif
```

Tipi di code
Si desidera creare una coda ogni volta che ne è necessaria una o si desidera utilizzare code già impostate? Si desidera eliminare una coda quando si è terminato di utilizzarla o verrà utilizzata di nuovo? Si desidera utilizzare le code alias per l'indipendenza dell'applicazione? Per vedere quali tipi di code sono supportati, fare riferimento a Code .

Usando i cluster del gestore code
È possibile trarre vantaggio dall'amministrazione del sistema semplificata e da una maggiore disponibilità, scalabilità e bilanciamento del carico di lavoro possibili quando si utilizzano i cluster. Per ulteriori informazioni, consultare Cluster di gestori code .

Tipi di messaggi
È possibile utilizzare i datagrammi per i messaggi semplici, ma richiedere i messaggi (per i quali si prevedono risposte) per altre situazioni. È possibile assegnare diverse priorità ad alcuni messaggi. Per ulteriori informazioni sulla progettazione dei messaggi, consultare “Progettazione dei messaggi” a pagina 93.

Utilizzo della messaggistica di pubblicazione / sottoscrizione o point - to - point
Utilizzando la messaggistica di pubblicazione / sottoscrizione, un'applicazione di invio invia le informazioni che desidera condividere in un messaggio IBM WebSphere MQ ad una destinazione standard gestita dalla sottoscrizione IBM WebSphere MQ publish? e consente a IBM WebSphere MQ di gestire la distribuzione di tali informazioni. L'applicazione di destinazione non deve sapere nulla sulla fonte delle informazioni che riceve, registra solo un interesse per uno o più argomenti e riceve tali informazioni quando sono disponibili. Per ulteriori informazioni sulla messaggistica di pubblicazione / sottoscrizione, consultare Introduzione alla messaggistica di pubblicazione / sottoscrizione IBM WebSphere MQ .

Utilizzando la messaggistica point - to - point, un'applicazione di invio invia un messaggio a una coda specifica, da dove sa che un'applicazione di ricezione lo richiamerà. Un'applicazione ricevente riceve i messaggi da una coda specifica e agisce sul contenuto. Un'applicazione spesso funziona sia come mittente che come destinatario, inviando una query a un'altra applicazione e ricevendo una risposta.

Controllo dei programmi IBM WebSphere MQ
È possibile che si desideri avviare alcuni programmi automaticamente o far sì che i programmi attendano l'arrivo di un determinato messaggio su una coda (utilizzando la funzione IBM WebSphere MQ *triggering* , consultare “Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger” a pagina 331). In alternativa, è possibile avviare un'altra istanza di un'applicazione quando i messaggi su una coda non vengono elaborati abbastanza velocemente (utilizzando la funzione IBM WebSphere MQ Eventi di strumentazione come descritto in Eventi di strumentazione).

Esecuzione della tua applicazione su un client IBM WebSphere MQ
L'MQI completa è supportato nell'ambiente del client e ciò consente a quasi tutte le applicazioni IBM WebSphere MQ di essere ricollegate per essere eseguite su un client IBM WebSphere MQ MQI. Collegare l'applicazione sul client IBM WebSphere MQ MQI alla libreria MQIC, piuttosto che alla libreria MQI.

Nota: Un'applicazione in esecuzione su un client IBM WebSphere MQ può connettersi a più di un gestore code contemporaneamente oppure utilizzare un nome gestore code con un asterisco (*) su una chiamata MQCONN o MQCONNX. Modificare l'applicazione se si desidera creare un collegamento alle librerie del gestore code invece che alle librerie del client, in quanto questa funzione non sarà disponibile.

Per ulteriori informazioni, consultare “Esecuzione di applicazioni nell'ambiente client IBM WebSphere MQ MQI” a pagina 362.

Prestazioni applicazioni
Le decisioni di progettazione possono influenzare le prestazioni dell'applicazione, per suggerimenti per migliorare le prestazioni delle applicazioni IBM WebSphere MQ , consultare “Progettazione e prestazioni delle applicazioni” a pagina 94.

Tecniche IBM WebSphere MQ avanzate
Per applicazioni più avanzate, è possibile utilizzare alcune tecniche IBM WebSphere MQ avanzate, come la correlazione delle risposte e la generazione e l'invio di informazioni di contesto IBM WebSphere MQ . Per ulteriori informazioni, consultare “Tecniche IBM WebSphere MQ avanzate” a pagina 95.

Proteggere i dati e mantenerne l'integrità
È possibile utilizzare le informazioni di contesto passate con un messaggio per verificare che il messaggio sia stato inviato da un'origine accettabile. È possibile utilizzare le funzioni di sincronizzazione fornite da IBM WebSphere MQ o dal sistema operativo per garantire che i dati rimangano congruenti con altre risorse (consultare “Commit e backout delle unità di lavoro” a pagina 325 per ulteriori dettagli). È possibile utilizzare la funzione *persistenza* dei messaggi IBM WebSphere MQ per assicurare la consegna di messaggi importanti.

Test delle applicazioni IBM WebSphere MQ

L'ambiente di sviluppo dell'applicazione per programmi IBM WebSphere MQ non è diverso da quello per qualsiasi altra applicazione, quindi è possibile utilizzare gli stessi strumenti di sviluppo e le funzioni di traccia IBM WebSphere MQ .

Gestione di eccezioni ed errori
È necessario considerare come elaborare i messaggi che non possono essere consegnati e come risolvere le situazioni di errore riportate dal gestore code. Per alcuni report, è necessario impostare le opzioni di report su MQPUT.

Concetti correlati
IBM WebSphere MQ
“Concetti dello sviluppo di applicazioni” a pagina 8
È possibile utilizzare una scelta di linguaggi procedurali o orientati agli oggetti per scrivere applicazioni IBM WebSphere MQ . Utilizzare i link in questo argomento per informazioni sui concetti di IBM WebSphere MQ che sono utili per sviluppatori di applicazioni.

“Scrittura di un'applicazione di accodamento” a pagina 195
Utilizzare queste informazioni per informazioni sulla scrittura delle applicazioni di accodamento, la connessione e la disconnessione da un gestore code, la pubblicazione / sottoscrizione e l'apertura e la chiusura di oggetti.

“Scrittura delle applicazioni client” a pagina 354
Informazioni necessarie per scrivere le applicazioni client su WebSphere MQ.

“Usando .Net” a pagina 562
WebSphere Le classi di MQ per .NET consentono a un programma scritto nel framework di programmazione .NET di connettersi a WebSphere MQ come client WebSphere MQ MQI o di connettersi direttamente a un WebSphere MQ .

“Usando C++” a pagina 630
WebSphere MQ fornisce classi C++ equivalenti agli oggetti di WebSphere MQ e alcune classi aggiuntive equivalenti ai tipi di dati array. Fornisce un certo numero di funzioni non disponibili tramite MQI.

“Utilizzo delle classi WebSphere MQ per JMS” a pagina 717
WebSphere MQ classes for Java Message Service (WebSphere MQ classes for JMS) è il fornitore JMS fornito con WebSphere MQ. Oltre ad implementare le interfacce definite nel pacchetto javax.jms , WebSphere MQ classes per JMS fornisce due serie di estensioni all'API JMS.

“Utilizzo di classi WebSphere MQ per Java” a pagina 654

Le classi WebSphere MQ per Java consentono di utilizzare WebSphere MQ in un ambiente Java. Un'applicazione Java può utilizzare le classi WebSphere MQ per Java o WebSphere MQ per JMS per accedere alle risorse WebSphere MQ .

"Utilizzo di Component Object Model Interface (WebSphere MQ Automation Classes for ActiveX)" a pagina 1035
Le WebSphere MQ Automation Classes for ActiveX (MQAX) sono componenti ActiveX che forniscono classi che è possibile utilizzare nella propria applicazione per accedere a WebSphere MQ.

Progettazione dei messaggi

Considerare gli aspetti forniti in queste informazioni per semplificare la progettazione dei messaggi.

Creare un messaggio quando si utilizza una chiamata MQI per inserire il messaggio su una coda. Come input per la chiamata, si forniscono alcune informazioni di controllo in un *descrittore di messaggi* (MQMD) e i dati che si desidera inviare ad un altro programma. Ma nella fase di progettazione, è necessario considerare quanto segue, perché influenzano il modo in cui si creano i propri messaggi:

Tipo di messaggio da utilizzare
Si sta progettando una semplice applicazione in cui è possibile inviare un messaggio, quindi non intraprendere ulteriori azioni? O stai chiedendo una risposta a una domanda? Se si sta facendo una domanda, è possibile includere nel descrittore del messaggio il nome della coda su cui si desidera ricevere la risposta.

Si desidera che i messaggi di richiesta e risposta siano sincroni? Ciò implica che si imposta un periodo di timeout per la risposta per rispondere alla richiesta, e se non si riceve la risposta entro tale periodo, viene trattato come un errore.

Oppure preferisci lavorare in modo asincrono, in modo che i tuoi processi non dipendano dal verificarsi di eventi specifici, come i segnali di temporizzazione comuni?

Un'altra considerazione è se hai tutti i tuoi messaggi all'interno di un'unità di lavoro.

Assegnazione di priorità differenti ai messaggi
È possibile assegnare un valore di priorità a ciascun messaggio e definire la coda in modo che conservi i messaggi in ordine di priorità. Se si esegue questa operazione, quando un altro programma richiama un messaggio dalla coda, riceve sempre il messaggio con la priorità più alta. Se la coda non conserva i messaggi in ordine di priorità, un programma che richiama i messaggi dalla coda li richiamerà nell'ordine in cui sono stati aggiunti alla coda.

I programmi possono anche selezionare un messaggio utilizzando l'identificativo assegnato dal gestore code quando il messaggio è stato inserito nella coda. In alternativa, è possibile generare i propri identificativi per ciascuno dei messaggi.

Effetto del riavvio del gestore code sui messaggi
Il gestore code conserva tutti i messaggi persistenti, ripristinandoli quando necessario dai file di log di WebSphere MQ , quando vengono riavviati. I messaggi non persistenti e le code dinamiche temporanee non vengono conservati. I messaggi che non si desidera eliminare devono essere definiti come persistenti quando vengono creati. Quando si scrive un'applicazione per i sistemi WebSphere MQ per Windows o WebSphere MQ su UNIX and Linux , accertarsi di conoscere il modo in cui il sistema è stato impostato rispetto all'allocazione del file di log per ridurre il rischio di progettare un'applicazione che verrà eseguita ai limiti del file di log.

Fornire informazioni personali al destinatario dei messaggi
Di solito, il gestore code imposta l'ID utente, ma le applicazioni debitamente autorizzate possono anche impostare questo campo, in modo che sia possibile includere il proprio ID utente e altre informazioni che il programma di ricezione può utilizzare per scopi di account o di sicurezza.

Quantità di code di ricezione
Se un messaggio potrebbe dover essere inserito su diverse code, è possibile utilizzare un elenco di distribuzioneo pubblicare su un argomento.

Progettazione e prestazioni delle applicazioni

Esistono diversi modi in cui la scarsa progettazione del programma può influire sulle prestazioni. Questi possono essere difficili da rilevare perché il programma può sembrare di eseguire bene se stesso, ma influenzano le prestazioni di altre attività. Diversi problemi specifici dei programmi che effettuano chiamate WebSphere MQ sono illustrati in questo argomento.

Ecco alcune idee per aiutarti a progettare applicazioni efficienti:

- Progetta la tua applicazione in modo che l'elaborazione proceda in parallelo con il tempo di pensiero di un utente:
  - Visualizzare un pannello e consentire all'utente di iniziare a digitare mentre l'applicazione è ancora in fase di inizializzazione.
  - Ottenere i dati necessari in parallelo da server differenti.
- Mantenere aperte le connessioni e le code se si intende riutilizzarle invece di aprirle e chiuderle ripetutamente, collegarle e disconnetterle.
- Tuttavia, un'applicazione server che sta inserendo solo un messaggio deve utilizzare MQPUT1.
- I gestori code sono ottimizzati per i messaggi di dimensione compresa tra 4 KB e 100 KB. I messaggi molto grandi sono inefficienti; è probabilmente meglio inviare 100 messaggi da 1 MB ciascuno piuttosto che un singolo messaggio da 100 MB. Anche i messaggi molto piccoli sono inefficienti. Il gestore code esegue la stessa quantità di lavoro per un messaggio a byte singolo come per un messaggio da 4 KB.
- Mantenere i messaggi all'interno di un'unità di lavoro, in modo che possano essere sottoposti a commit o a backout contemporaneamente.
- Utilizzare l'opzione non persistente per i messaggi che non devono essere ripristinabili.
- Se è necessario inviare un messaggio a un numero di code di destinazione, utilizzare un elenco di distribuzione.

Effetto della lunghezza del messaggio

La quantità di dati in un messaggio può influire sulle prestazioni dell'applicazione che elabora il messaggio. Per ottenere le migliori prestazioni dalla tua applicazione, invia solo i dati essenziali in un messaggio. Ad esempio, in una richiesta di addebito di un conto bancario, le uniche informazioni che potrebbero dover essere inoltrate dal client all'applicazione server sono il numero di conto e l'importo dell'addebito.

Effetto della persistenza del messaggio

I messaggi persistenti vengono generalmente registrati. La registrazione dei messaggi riduce le prestazioni della tua applicazione, quindi utilizza i messaggi persistenti solo per i dati essenziali. Se i dati in un messaggio possono essere eliminati se il gestore code si arresta o ha esito negativo, utilizzare un messaggio non persistente.

Ricerca di un particolare messaggio

La chiamata MQGET di solito richiama il primo messaggio da una coda. Se si utilizzano gli identificativi di correlazione e messaggio (*MsgId* e *CorrelId*) nel descrittore del messaggio per specificare un particolare messaggio, il gestore code deve ricercare la coda fino a quando non trova quel messaggio. L'utilizzo della chiamata MQGET in questo modo influisce sulle prestazioni dell'applicazione.

Code che contengono messaggi di lunghezza diversa

Se l'applicazione non può utilizzare messaggi di lunghezza fissa, ingrandire e ridurre i buffer in modo dinamico per adattarli alla dimensione tipica del messaggio. Se l'applicazione emette una chiamata MQGET che non riesce perché il buffer è troppo piccolo, viene restituita la dimensione dei dati del messaggio. Aggiungere codice alla propria applicazione in modo che il buffer venga ridimensionato di conseguenza e la chiamata MQGET venga reimmessa.

Nota: se non si imposta esplicitamente l'attributo *MaxMsgLength* , il valore predefinito è 4 MB, che potrebbe essere molto inefficiente se viene utilizzato per influenzare la dimensione del buffer dell'applicazione.

Frequenza dei punti di sincronizzazione

I programmi che emettono un numero molto elevato di chiamate MQPUT o MQGET all'interno del punto di sincronizzazione, senza eseguirne il commit, possono causare problemi di prestazioni. Le code interessate possono riempirsi di messaggi attualmente inaccessibili, mentre altre attività potrebbero essere in attesa di ricevere tali messaggi. Ciò ha implicazioni in termini di memoria, e in termini di thread collegati con le attività che stanno tentando di ottenere messaggi.

Utilizzo della chiamata MQPUT1

Utilizzare la chiamata MQPUT1 solo se si dispone di un singolo messaggio da inserire in una coda. Se si desidera inserire più di un messaggio, utilizzare la chiamata MQOPEN, seguita da una serie di chiamate MQPUT e da una singola chiamata MQCLOSE.

Numero di thread in uso

Per WebSphere MQ per Windows, un'applicazione potrebbe richiedere un numero elevato di thread. A ciascun processo del gestore code viene assegnato un numero massimo consentito di thread dell'applicazione.

Le applicazioni potrebbero utilizzare troppi thread. Considerare se l'applicazione prende in considerazione questa possibilità e se intraprende azioni per arrestare o notificare questo tipo di ricorrenza.

Tecniche IBM WebSphere MQ avanzate

Per una semplice applicazione IBM WebSphere MQ , è necessario stabilire quali oggetti WebSphere MQ utilizzare nell'applicazione e quali tipi di messaggi si desidera utilizzare. Per un'applicazione più avanzata, è possibile utilizzare alcune delle tecniche introdotte nelle seguenti sezioni.

In attesa di messaggi

Un programma che serve una coda può attendere i messaggi per:

- Attesa fino all'arrivo di un messaggio o fino alla scadenza di un intervallo di tempo specificato (consultare “In attesa di messaggi” a pagina 269).
- Stabilire un'uscita di callback da guidare quando arriva un messaggio; consultare “Utilizzo asincrono dei messaggi IBM WebSphere MQ” a pagina 34.
- Esecuzione di chiamate periodiche sulla coda per verificare se un messaggio è arrivato (*polling*). Ciò non è in genere consigliabile perché può avere implicazioni sulle prestazioni.

Correlazione delle risposte

Nelle applicazioni WebSphere MQ , quando un programma riceve un messaggio che lo richiede per eseguire alcune operazioni, generalmente il programma invia uno o più messaggi di risposta al richiedente.

Per aiutare il richiedente ad associare queste risposte alla sua richiesta originale, un'applicazione può impostare un *identificativo di correlazione* nel descrittore di ogni messaggio. I programmi quindi copiano l'identificativo del messaggio di richiesta nel campo identificativo di correlazione dei relativi messaggi di risposta.

Impostazione e utilizzo delle informazioni di contesto

Le *informazioni di contesto* sono utilizzate per associare i messaggi all'utente che li ha generati e per identificare l'applicazione che ha generato il messaggio. Tali informazioni sono utili per la sicurezza, la contabilità, il controllo e la determinazione dei problemi.

Quando si crea un messaggio, è possibile specificare un'opzione che richiede che il gestore code associ le informazioni di contesto predefinite al proprio messaggio.

Per ulteriori informazioni sull'utilizzo e l'impostazione delle informazioni di contesto, consultare “Contesto messaggio” a pagina 39.

Avvio automatico dei programmi WebSphere MQ

Utilizzare WebSphere MQ *trigger* per avviare automaticamente un programma quando i messaggi arrivano su una coda.

È possibile impostare condizioni di trigger su una coda in modo che un programma inizi ad elaborare tale coda:

- Ogni volta che un messaggio arriva sulla coda
- Quando arriva il primo messaggio sulla coda
- Quando il numero di messaggi sulla coda raggiunge un numero predefinito

Per ulteriori informazioni sull'attivazione, consultare “Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger” a pagina 331. L'attivazione è solo un modo per avviare un programma automaticamente. Ad esempio, è possibile avviare un programma automaticamente su un timer utilizzando funzioni nonWebSphere MQ .

WebSphere MQ può definire oggetti di servizio per avviare i programmi WebSphere MQ all'avvio del gestore code; consultare Oggetti di servizio.

Generazione di report WebSphere MQ

È possibile richiedere i seguenti report all'interno di un'applicazione:

- Report di eccezioni
- Report di scadenza
- Report COA (Confirm - on - arrival)
- Report COD (Confirm - on - delivery)
- Report PAN (positive action notification)
- Report NAN (negative action notification)

Tali regole sono descritte in “Messaggi di report” a pagina 11.

Cluster e affinità di messaggi

Prima di iniziare ad utilizzare i cluster con più definizioni per la stessa coda, esaminare le applicazioni per verificare se vi sono dei cluster che richiedono uno scambio di messaggi correlati.

All'interno di un cluster, un messaggio può essere instradato a qualsiasi gestore code che ospita un'istanza della coda appropriata. Pertanto, la logica delle applicazioni con affinità di messaggi può essere alterata.

Ad esempio, si potrebbero avere due applicazioni che si basano su una serie di messaggi che scorrono tra di loro sotto forma di domande e risposte. Potrebbe essere importante che tutte le domande vengano inviate allo stesso gestore code e che tutte le risposte vengano inviate nuovamente all'altro gestore code. In questa situazione, è importante che la routine di gestione del carico di lavoro non invii i messaggi ad alcun gestore code che ospita un'istanza della coda appropriata.

Laddove possibile, rimuovere le affinità. La rimozione delle affinità dei messaggi migliora la disponibilità e scalabilità delle applicazioni.

Per ulteriori informazioni, consultare Gestione delle affinità dei messaggi .

Programmi di WebSphere MQ di esempio

Utilizzare questa raccolta di argomenti per informazioni sui programmi WebSphere MQ di esempio su piattaforme differenti.

- “Programmi di esempio per piattaforme distribuite” a pagina 97

Concetti correlati

“Concetti dello sviluppo di applicazioni” a pagina 8
È possibile utilizzare una scelta di linguaggi procedurali o orientati agli oggetti per scrivere applicazioni IBM WebSphere MQ . Utilizzare i link in questo argomento per informazioni sui concetti di IBM WebSphere MQ che sono utili per sviluppatori di applicazioni.

“Scelta del linguaggio di programmazione da utilizzare” a pagina 79
Utilizza queste informazioni per scoprire i linguaggi di programmazione e i framework supportati da IBM WebSphere MQe alcune considerazioni per utilizzarli.

“Progettazione di applicazioni IBM WebSphere MQ” a pagina 90
Una volta stabilito il modo in cui le applicazioni possono sfruttare le piattaforme e gli ambienti disponibili, è necessario decidere come utilizzare le funzioni offerte da WebSphere MQ.

“Scrittura di un'applicazione di accodamento” a pagina 195
Utilizzare queste informazioni per informazioni sulla scrittura delle applicazioni di accodamento, la connessione e la disconnessione da un gestore code, la pubblicazione / sottoscrizione e l'apertura e la chiusura di oggetti.

“Scrittura delle applicazioni client” a pagina 354
Informazioni necessarie per scrivere le applicazioni client su WebSphere MQ.

“Utilizzo dei servizi Web in WebSphere MQ” a pagina 948
È possibile sviluppare applicazioni IBM WebSphere MQ per servizi Web utilizzando il trasporto IBM WebSphere MQ per SOAP o il bridge IBM WebSphere MQ per HTTP.

“Scrittura di applicazioni di pubblicazione / sottoscrizione” a pagina 279
Iniziare a scrivere applicazioni WebSphere MQ di pubblicazione / sottoscrizione.

“Creazione di un'applicazione IBM WebSphere MQ” a pagina 431
Utilizzare queste informazioni per informazioni sulla generazione di un'applicazione IBM WebSphere MQ su piattaforme differenti.

“Gestione degli errori del programma” a pagina 550
Queste informazioni spiegano gli errori associati alle chiamate MQI delle applicazioni quando effettua una chiamata o quando il suo messaggio viene consegnato alla destinazione finale.

Programmi di esempio per piattaforme distribuite

Questo argomento descrive i programmi di esempio forniti con IBM WebSphere MQ, scritti in C e COBOL. Gli esempi mostrano gli utilizzi tipici di MQI (Message Queue Interface).

Gli esempi non sono destinati a dimostrare tecniche di programmazione generali, quindi viene omesso il controllo degli errori che si potrebbe voler includere in un programma di produzione. Tuttavia, questi esempi sono adatti per essere utilizzati come base per i propri programmi di accodamento messaggi.

Il codice sorgente per tutti gli esempi viene fornito con il prodotto; questa sorgente include commenti che spiegano le tecniche di accodamento dei messaggi dimostrate nei programmi.

Programmi di esempio C++: consultare Utilizzo di C++ per una descrizione dei programmi di esempio disponibili in C + +.

I nomi degli esempi iniziano con il prefisso amq. Il quarto carattere indica il linguaggio di programmazione e, se necessario, il compilatore.

s	linguaggio C
0	Linguaggio COBOL su compilatori IBM e Micro Focus
i	Linguaggio COBOL solo su compilatori IBM
m	Linguaggio COBOL solo su compilatori Micro Focus

L'ottavo carattere dell'eseguibile indica se l'esempio viene eseguito in modalità di collegamento locale o in modalità client. Se non è presente un ottavo carattere, l'esempio viene eseguito in modalità bind locale. Se l'ottavo carattere è 'c', l'esempio viene eseguito in modalità client. Per impostare il gestore code per accettare le connessioni client, consultare “Preparazione ed esecuzione dei programmi di esempio” a pagina 110 per i dettagli.

Utilizzare i seguenti collegamenti per ulteriori informazioni sui programmi di esempio:

- “Funzioni dimostrate nei programmi di esempio” a pagina 98
- “Programmi di esempio di pubblicazione / sottoscrizione” a pagina 136
- “Programmi di esempio Put” a pagina 141
- “Programma di esempio Elenco di distribuzione” a pagina 128
- “Programmi di esempio Sfoglia” a pagina 117
- “Il programma di esempio Browser” a pagina 118
- “I programmi di esempio Get” a pagina 130
- “Programmi di esempio del messaggio di riferimento” a pagina 142
- “I programmi di esempio Richiesta” a pagina 148
- “I programmi di esempio Inquire” a pagina 134
- “Il programma di esempio Inquire Properties of a Message Handle” a pagina 135
- “I programmi di esempio Set” a pagina 152
- “I programmi di esempio Echo” a pagina 129
- “Il programma di esempio Conversione dati” a pagina 120
- “I programmi di esempio Trigger” a pagina 156
- “Il programma di esempio Put asincrono” a pagina 116
- “Esempi di coordinamento database” a pagina 121
- “L'esempio di transazione CICS” a pagina 119
- “Esempi TUXEDO” a pagina 157
- “Esempio di gestore code di messaggi non instradabili” a pagina 128
- “Il programma di esempio Connect” a pagina 119
- “Il programma di esempio di uscita API” a pagina 114
- “Utilizzo dell'uscita di sicurezza SSPI su sistemi Windows” a pagina 170
- “Esecuzione degli esempi utilizzando le code remote” a pagina 171
- “Programma di esempio Monitoraggio coda cluster (AMQSCLM)” a pagina 171
- “Programma di esempio per Connection Endpoint Lookup (CEPL)” a pagina 180

Funzioni dimostrate nei programmi di esempio

Una raccolta di tabelle che mostra le tecniche dimostrate dai programmi di esempio WebSphere MQ .

Tutti gli esempi aprono e chiudono code utilizzando le chiamate MQOPEN e MQCLOSE, quindi queste tecniche non sono elencate separatamente nelle tabelle. Vedere l'intestazione che include la piattaforma a cui si è interessati.

Esempi per sistemi UNIX and Linux

Questo argomento mostra le tecniche dimostrate dai programmi di esempio per WebSphere MQ su sistemi UNIX and Linux .

Consultare “Preparazione ed esecuzione di programmi di esempio su sistemi UNIX” a pagina 112 per informazioni sull'ubicazione in cui sono memorizzati i programmi di esempio per WebSphere MQ su sistemi UNIX e Linux .

Tabella 14 a pagina 99 La tabella elenca i file di origine C e COBOL forniti e se è incluso un server o un client eseguibile.

Tabella 14. WebSphere MQ su programmi di esempio UNIX and Linux che dimostrano l'utilizzo di MQI (C e COBOL)

Tecnica	**C (origine) (“1” a pagina 101)**	**COBOL (origine) (“2” a pagina 101)**	**Server (eseguibile C)**	**Client (eseguibile C) (“3” a pagina 101)**
Utilizzo dell'interfaccia di pubblicazione / sottoscrizione	amqspuba amqssuba amqssbxa	nessun campione	amqspub amqssub amqssbx	nessun campione
Inserimento di messaggi utilizzando la chiamata MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Inserimento di un singolo messaggio utilizzando la chiamata MQPUT1	amqsinqa amqsecha	amqminqx amqmechx amqiinqx amqiechx	amqsinq amqsech	amqsechc
Inserimento di messaggi in un elenco di distribuzione (“4” a pagina 101)	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Risposta a un messaggio di richiesta	amqsinqa	amqminqx amqiinqx	amqsinq	nessun campione
Ricezione messaggi (nessuna attesa)	amqsgbr0	amq0gbr0	amqsgbr	nessun campione
Ricezione dei messaggi (attendere con un limite di tempo)	amqsget0	amq0get0	amqsget	amqsgetc
Ricezione messaggi (attesa illimitata)	amqstrg0	nessun campione	amqstrg	amqstrgc
Ricezione di messaggi (con conversione dati)	amqsecha	nessun campione	amqsech	nessun campione
Inserimento di messaggi di riferimento in una coda (“4” a pagina 101)	amqsprma	nessun campione	amqsprm	amqsprmc
Richiamo dei messaggi di riferimento da una coda (“4” a pagina 101)	amqsgrma	nessun campione	amqsgrm	amqsgrmc
Uscita del canale dei messaggi di riferimento (“4” a pagina 101)	amqsqrma amqsxrma	nessun campione	amqsxrm	nessun campione
Ricerca dei primi 20 caratteri di un messaggio	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Visualizzazione dei messaggi completi	amqsbcg0	nessun campione	amqsbcg	amqsbcgc
Utilizzo di una coda di input condivisa	amqsinqa	amqminqx amqiinqx	amqsinq	amqsinqc

Tabella 14. WebSphere MQ su programmi di esempio UNIX and Linux che dimostrano l'utilizzo di MQI (C e COBOL) (Continua)

Tecnica	**C (origine) (“1” a pagina 101)**	**COBOL (origine) (“2” a pagina 101)**	**Server (eseguibile C)**	**Client (eseguibile C) (“3” a pagina 101)**
Utilizzo di una coda di immissione esclusiva	amqstrg0	amq0req0	amqstrg	amqstrgc
Utilizzo della chiamata MQINQ	amqsinqa	amqminqx amqiinqx	amqsinq	nessun campione
Utilizzo della chiamata MQSET	amqsseta	amqmsetx amqisetx	amqsset	amqssetc
Utilizzo di una coda di risposta	amqsreq0	amq0req0	amqsreq	amqsreqc
Richiesta di eccezioni ai messaggi	amqsreq0	amq0req0	amqsreq	nessun campione
Accettazione di un messaggio troncato	amqsgbr0	amq0gbr0	amqsgbr	nessun campione
Utilizzo di un nome coda risolto	amqsgbr0	amq0gbr0	amqsgbr	nessun campione
Attivazione di un processo	amqstrg0	nessun campione	amqstrg	amqstrgc
Utilizzo della conversione dati	(“5” a pagina 101)	nessun campione	nessun campione	nessun campione
WebSphere MQ (che coordina i gestori database conformi a XA) che accedono a un singolo database utilizzando SQL	amqsxas0.sqc DB2 amqsxas0.ec Informix	amq0xas0.sqb	nessun campione	nessun campione
WebSphere MQ (che coordina i gestori database compatibili con XA) che accedono a due database utilizzando SQL	amqsxag0.c amqsxab0.sqc amqsxaf0.sqc	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	nessun campione	nessun campione
Transazione CICS (“6” a pagina 101)	amqscic0.ccs	nessun campione	amqscic0	nessun campione
transazione Encina (“4” a pagina 101)	amqsxae0	nessun campione	amqsxae0	nessun campione
Transazione TUXEDO per inserire messaggi (“7” a pagina 101)	amqstxpx	nessun campione	nessun campione	nessun campione
Transazione TUXEDO per richiamare i messaggi (“7” a pagina 101)	amqstxgx	nessun campione	nessun campione	nessun campione
Server per TUXEDO (“7” a pagina 101)	amqstxsx	nessun campione	nessun campione	nessun campione
gestore coda di messaggi non instradabili (DLQ, dead-letter queue)	Directory `./tools/c/Samples/dlq` (“8” a pagina 101)	nessun campione	amqsdlq	nessun campione

Tabella 14. WebSphere MQ su programmi di esempio UNIX and Linux che dimostrano l'utilizzo di MQI (C e COBOL) (Continua)

Tecnica	C (origine) (“1” a pagina 101)	COBOL (origine) (“2” a pagina 101)	Server (eseguibile C)	Client (eseguibile C) (“3” a pagina 101)
Da un client MQI, inserimento di un messaggio	nessun campione	nessun campione	nessun campione	amqsputc
Da un client MQI, ricezione di un messaggio	nessun campione	nessun campione	nessun campione	amqsgetc
Connessione al gestore code mediante MQCONNX	amqscnxc	nessun campione	nessun campione	amqscnxc
Utilizzo delle uscite API	amqsaxe0	nessun campione	amqsaxe	nessun campione
Uscita bilanciamento carico di lavoro cluster	amqswlm0	nessun campione	amqswlm	nessun campione
Inserimento dei messaggi in modo asincrono e richiamo dello stato utilizzando la chiamata MQSTAT	amqsapt0	nessun campione	amqsapt	amqsaptc
Client riconnettibili	amqsphac amqsghac amqsmhac	nessun campione	non applicabile	amqsphac amqsghac amqsmhac
Utilizzo dei destinatari dei messaggi per l'utilizzo asincrono dei messaggi da più code	amqscbf0	nessun campione	amqscbf	amqscbfc
Specifica delle informazioni di connessione SSL/TLS su MQCONNX	amqssslc	nessun campione	non applicabile	amqssslc

Note:

1. La versione eseguibile degli esempi del client WebSphere MQ MQI condivide la stessa origine degli esempi eseguiti in un ambiente server.
2. Compilare i programmi che iniziano con 'amqm' con il compilatore Micro Focus COBOL, quelli che iniziano con 'amqi' con il compilatore IBM COBOL e quelli che iniziano con 'amq0'.
3. Le versioni eseguibili degli esempi client WebSphere MQ MQI non sono disponibili su WebSphere MQ per HP-UX.
4. Supportato su WebSphere MQ per AIX, WebSphere MQ per HP-UXe WebSphere MQ solo per Solaris.
5. Su WebSphere MQ per AIX, WebSphere MQ per HP-UXe WebSphere MQ per Solaris, questo programma è denominato `amqsvfc0.c`
6. CICS è supportato solo da WebSphere MQ per AIX e WebSphere MQ per HP-UX .
7. TUXEDO non è supportato da WebSphere MQ per Linux su System p.
8. L'origine per il gestore code di messaggi non recapitabili è costituita da diversi file ed è fornita in una directory separata.

Informazioni dettagliate sul supporto per i sistemi UNIX and Linux sono disponibili nella pagina dei requisiti di sistemi WebSphere MQ all'indirizzo Requisiti di sistema per IBM WebSphere MQ.

Esempi per client IBM WebSphere MQ per HP Integrity NonStop Server

Questo argomento mostra le tecniche dimostrate dai programmi di esempio per il client IBM WebSphere MQ sui sistemi HP Integrity NonStop Server .

Tabella 15 a pagina 102La tabella elenca i programmi di esempio di origine C, COBOL e pTAL forniti.

Tabella 15. IBM WebSphere MQ su programmi di esempio HP Integrity NonStop Server che dimostrano l'utilizzo di C, COBOL e pTAL

Tecnica	**C**				**COBOL**		**pTAL**	
	OSS (Origine)	OSS (eseguibile)	Guardiano (origine)	Guardiano (eseguibile)	OSS (Origine)	Guardiano (origine)	OSS (Origine)	Guardiano (origine)
Utilizzo dell'interfaccia di pubblicazione / sottoscrizione	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
Inserimento di messaggi utilizzando la chiamata MQPUT	amqsput0.c	amqsputc	MQSPUTC	AMQSPUTC	amq0put0.cbl	MQSPUTL	amqtput0.tal	MQSPUTT
Inserimento di un singolo messaggio utilizzando la chiamata MQPUT1	amqsecha.c	amqsechc	MQSECHC	AMQSECHC			amqtech0.tal	MQSECHT
Inserimento di messaggi in un elenco di distribuzione	amqsptl0.c	amqsptlc	MQSPTLC	AMQSPTLC	amq0ptl0.cbl	MQSPTLL		
Risposta a un messaggio di richiesta	amqsinqa.c	amqsinqc	MQSINQC	AMQSINQC				
Ricezione messaggi (nessuna attesa)	amqsgbr0.c	amqsgbrc	MQSGBRC	BRCAMQSGC	amq0gbr0.cbl	MQSGBRL		

Tabella 15. IBM WebSphere MQ su programmi di esempio HP Integrity NonStop Server che dimostrano l'utilizzo di C, COBOL e pTAL (Continua)

Tecnica	**C**				**COBOL**		**pTAL**	
Ricezione dei messaggi (attendere con un limite di tempo)	amqsget0.c	amqsgetc	MQSGETC	GETC AMQS	amq0get0.cbl	MQSGETL	amqtget0.tal	MQSGETT
Ricezione messaggi (attesa illimitata)	amqstrg0.c	amqstrgc	MQSTRGC	AMQSTRGC				
Ricezione di messaggi (con conversione dati)	amqsecha.c	amqsechc	MQSECHC	AMQSECHC				
Inserimento di messaggi di riferimento in una coda	amqsprma.c	amqsprmc	MMQSPRMC	MMQSPRMC				
Richiamo dei messaggi di riferimento da una coda	amqsgrma.c	amqsgrmc	MMQSGRMC	GRMAMQS				
Uscita canale messaggi di riferimento	amqsqrma.c amqsxrma.c		MQSQRMC MQSXRMC					
Ricerca dei primi 20 caratteri di un messaggio	amqsgbr0.c	amqsgbrc	MQSGBRC	BRCAMQSGC	amq0gbr0.cbl	MQSGBRL		

Tabella 15. IBM WebSphere MQ su programmi di esempio HP Integrity NonStop Server che dimostrano l'utilizzo di C, COBOL e pTAL (Continua)

Tecnica	C				COBOL		pTAL	
Visualizzazione dei messaggi completi	amqsbcg0.c	amqsbcgc	MQSBCGC	AMQSBCGC				
Utilizzo di una coda di input condivisa	amqsinqa.c	amqsinqc	MQSINQC	MQSINQC				
Utilizzo di una coda di immissione esclusiva	amqstrg0.c	amqstrgc	MQSTRGC	AMQSTRGC				
Utilizzo della chiamata MQINQ	amqsinqa.c	amqsinqc	MQSINQC	AMQSINQC				
Utilizzo della chiamata MQSET	amqsseta.c	amqssetc	MQSSETC	AMQSSEC				
Utilizzo di una coda di risposta	amqsreq0.c	amqsreqc	MQSREQC	AMQSREQC	amq0req0.cbl	MQSREQL		
Richiesta di eccezioni ai messaggi	amqsreq0.c	amqsreqc	MQSREQC	AMQSREQC	amq0req0.cbl	MQSREQL		
Accettazione di un messaggio troncato	amqsgbr0.c	amqsgbrc	MQSGBRC	BRCAMQSGC	amq0gbr0.cbl	MQSGBRL		
Utilizzo di un nome coda risolto	amqsgbr0.c	amqsgbrc	MQSGBRC	BRCAMQSGC	amq0gbr0.cbl	MQSGBRL		

Tabella 15. IBM WebSphere MQ su programmi di esempio HP Integrity NonStop Server che dimostrano l'utilizzo di C, COBOL e pTAL (Continua)

Tecnica	**C**				**COBOL**		**pTAL**	
Attivazione di un processo	amqstrg0.c	amqstrgc	MQSTRGC	AMQSTRGC				
Utilizzo della conversione dati	amqsvfc0.c							
Gestore code di messaggi non recapitabili (1)	Directory ./samp/dlq							
Connessione a un gestore code mediante MQCONNX	amqscnxc.c	amqscnxc	MQSCNXC					
Utilizzo delle uscite API	amqsaxe0.c amqsaem0.c							
Uscita bilanciamento carico di lavoro cluster	amqswlm0.c		LMC MQSWL					
Monitor coda cluster	amqsclma.c							
Inserimento dei messaggi in modo asincrono e richiamo dello stato utilizzando la chiamata MQSTAT	amqsapt0.c	amqsaptc	TMQSAPC	TMQSAPC				

Tabella 15. IBM WebSphere MQ su programmi di esempio HP Integrity NonStop Server che dimostrano l'utilizzo di C, COBOL e pTAL (Continua)

Tecnica	**C**				**COBOL**		**pTAL**	
Client riconnettibili	amqsghac.c amqsmhac.c amqsphac.c	amqsghac amqsmhac amqsphac	MQSGHAC MQSMHAC MQSPHAC MQSFHAC	AMQSGHAC AMQSMHAC AMQSPHAC AMQSFHAC				
Utilizzo dei destinatari dei messaggi per l'utilizzo asincrono dei messaggi da più code	amqscbf0.c	amqscbfc						
Specifica delle informazioni di connessione SSL/TLS su MQCONNX	amqssslc.c	amqssslc	SSLC MQS	SSLC AMQS				
Traccia attività	amqsact0.c	amqsactc	CACTMQS	AMQSACTC				
Proprietà dei messaggi	amqsiqma.c amqsstma.c	amqsiqmc amqsstmc	MQSIQMC MQSSTMC	AMQSIQMC AMQSSTMC				
Server dei comandi	amqsstop.c		MQSSTOC					
Registra eventi	amqslog0.c	amqslogc	LOGMQS	LOGAMQS				
Contabilità	amqsmon0.c	amqsmonc	MQSMONC	AMQSMONC				

Tabella 15. IBM WebSphere MQ su programmi di esempio HP Integrity NonStop Server che dimostrano l'utilizzo di C, COBOL e pTAL (Continua)

Tecnica	C				COBOL		pTAL	
Interfaccia di amministrazione	amqsaicq.c amqsaiem.c amqsailq.c							
Un esempio di funzione principale del linguaggio C per richiamare pTAL			MMQSPTMC					

Note:

1. L'origine per il gestore code di messaggi non recapitabili è costituita da diversi file ed è fornita in una directory separata.
2. Per informazioni sullo sviluppo di applicazioni per il proprio client IBM WebSphere MQ sulla piattaforma HP Integrity NonStop Server , consultare:
   - "Creazione della tua applicazione su HP Integrity NonStop Server" a pagina 437
     - "Preparazione dei programmi C in HP Integrity NonStop Server" a pagina 439
     - "Preparazione dei programmi COBOL" a pagina 440
     - "Preparazione dei programmi pTAL" a pagina 442

Esempi per IBM WebSphere MQ per Finestre

Questo argomento mostra le tecniche dimostrate dai programmi di esempio per IBM WebSphere MQ per Finestre.

Tabella 16 a pagina 107La tabella elenca i file di origine C e COBOL forniti e se è incluso un server o un client eseguibile.

Tabella 16. Programmi di esempio IBM WebSphere MQ per Finestre che dimostrano l'utilizzo di MQI (C e COBOL)

Tecnica	C (origine)	COBOL (origine)	Server (eseguibile C)	Client (eseguibile C)
Utilizzo dell'interfaccia di pubblicazione / sottoscrizione	amqspuba amqssuba amqssbxa	nessun campione	amqspub amqssub amqssbx	nessun campione
Inserimento di messaggi utilizzando la chiamata MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Inserimento di un singolo messaggio utilizzando la chiamata MQPUT1	amqsinqa amqsecha	amqminq2 amqmech2 amqiinq2 amqiech2	amqsinq amqsech	amqsinqc amqsechc

Tabella 16. Programmi di esempio IBM WebSphere MQ per Finestre che dimostrano l'utilizzo di MQI (C e COBOL) (Continua)

Tecnica	C (origine)	COBOL (origine)	Server (eseguibile C)	Client (eseguibile C)
Inserimento di messaggi in un elenco di distribuzione	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Risposta a un messaggio di richiesta	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Ricezione messaggi (nessuna attesa)	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Ricezione dei messaggi (attendere con un limite di tempo)	amqsget0	amq0get0	amqsget	amqsgetc
Ricezione messaggi (attesa illimitata)	amqstrg0	nessun campione	amqstrg	amqstrgc
Ricezione di messaggi (con conversione dati)	amqsecha	nessun campione	amqsech	amqsechc
Inserimento di messaggi di riferimento in una coda	amqsprma	nessun campione	amqsprm	amqsprmc
Richiamo dei messaggi di riferimento da una coda	amqsgrma	nessun campione	amqsgrm	amqsgrmc
Uscita canale messaggi di riferimento	amqsqrma amqsxrma	nessun campione	amqsxrm	nessun campione
Ricerca dei primi 20 caratteri di un messaggio	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Visualizzazione dei messaggi completi	amqsbcg0	nessun campione	amqsbcg	amqsbcgc
Utilizzo di una coda di input condivisa	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Utilizzo di una coda di immissione esclusiva	amqstrg0	amq0req0	amqstrg	amqstrgc
Utilizzo della chiamata MQINQ	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Utilizzo della chiamata MQSET	amqsseta	amqmset2 amqiset2	amqsset	amqssetc
Utilizzo della chiamata MQINQMP	amqsiqma	nessun campione	nessun campione	nessun campione
Utilizzo di una coda di risposta	amqsreq0	amq0req0	amqsreq	amqsreqc
Richiesta di eccezioni ai messaggi	amqsreq0	amq0req0	amqsreq	amqsreqc
Accettazione di un messaggio troncato	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Utilizzo di un nome coda risolto	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Attivazione di un processo	amqstrg0	nessun campione	amqstrg	amqstrgc
Utilizzo della conversione dati	amqsvfc0	nessun campione	nessun campione	nessun campione

Tabella 16. Programmi di esempio IBM WebSphere MQ per Finestre che dimostrano l'utilizzo di MQI (C e COBOL) (Continua)

Tecnica	**C (origine)**	**COBOL (origine)**	**Server (eseguibile C)**	**Client (eseguibile C)**
WebSphere MQ (che coordina i gestori database conformi a XA) che accedono a un singolo database utilizzando SQL	amqsxas0.sqc DB2 amqsxas0.ec Informix	amq0xas0.sqb	nessun campione	nessun campione
WebSphere MQ (che coordina i gestori database compatibili con XA) che accedono a due database utilizzando SQL	amqsxag0.c amqsxab0.sqc DB2 amqsxaf0.sqc DB2	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	nessun campione	nessun campione
Transazione TUXEDO per inserire messaggi	amqstxpx	nessun campione	nessun campione	nessun campione
Transazione TUXEDO per richiamare i messaggi	amqstxgx	nessun campione	nessun campione	nessun campione
Server per TUXEDO	amqstxsx	nessun campione	nessun campione	nessun campione
gestore coda di messaggi non instradabili (DLQ, dead-letter queue)	Directory `./tools/c/Samples/dlq` (“1” a pagina 110)	nessun campione	amqsdlq	nessun campione
Da un client WebSphere MQ MQI, inserimento di un messaggio	nessun campione	nessun campione	nessun campione	amqsputc
Da un client WebSphere MQ MQI, ottenendo un messaggio	nessun campione	nessun campione	nessun campione	amqsgetc
Connessione al gestore code mediante MQCONNX	amqscnxc	nessun campione	nessun campione	amqscnxc
Utilizzo delle uscite API	amqsaxe0	nessun campione	amqsaxe	nessun campione
Bilanciamento carico di lavoro cluster	amqswlm0	nessun campione	amqswlm	nessun campione
Routine di sicurezza SSPI	amqsspin	nessun campione	amqrspin.dll	amqrspin.dll
Inserimento dei messaggi in modo asincrono e richiamo dello stato utilizzando la chiamata MQSTAT	amqsapt0	nessun campione	amqsapt	amqsaptc
Client riconnettibili	amqsphac amqsghac amqsmhac	nessun campione	Non applicabile	amqsphac amqsghac amqsmhac
Utilizzo dei destinatari dei messaggi per l'utilizzo asincrono dei messaggi da più code	amqscbf0	nessun campione	amqscbf	amqscbfc

Tabella 16. Programmi di esempio IBM WebSphere MQ per Finestre che dimostrano l'utilizzo di MQI (C e COBOL) (Continua)

Tecnica	C (origine)	COBOL (origine)	Server (eseguibile C)	Client (eseguibile C)
Specifica delle informazioni di connessione SSL/TLS su MQCONNX	amqssslc	nessun campione	non applicabile	amqssslc

Note:

1. L'origine per il gestore code di messaggi non recapitabili è costituita da diversi file ed è fornita in una directory separata.

Esempi di Visual Basic per IBM WebSphere MQ per Finestre

Questo argomento mostra le tecniche dimostrate dai programmi di esempio di Visual Basic per IBM WebSphere MQ per Windows.

La Tabella 17 a pagina 110 mostra le tecniche dimostrate dai programmi di esempio IBM WebSphere MQ per Windows .

Un progetto può contenere diversi file. Quando si apre un progetto in Visual Basic, gli altri file vengono caricati automaticamente. Non viene fornito alcun programma eseguibile.

Tutti i progetti di esempio, eccetto mqtrivc.vbp, sono configurati per funzionare con il server IBM WebSphere MQ . Per informazioni su come modificare i progetti di esempio per utilizzare i client IBM WebSphere MQ , vedere “Preparazione dei programmi Visual Basic in Windows” a pagina 465.

Tabella 17. Programmi di esempio di IBM WebSphere MQ per Windows che dimostrano l'utilizzo di MQI (Visual Basic)

Tecnica	Nome file di progetto
Inserimento di messaggi utilizzando la chiamata MQPUT	amqsputb.vbp
Richiamo dei messaggi utilizzando la chiamata MQGET	amqsgetb.vbp
Esplorazione di una coda utilizzando la chiamata MQGET	amqsbcgb.vbp
Esempio MQGET e MQPUT semplice (client)	mqtrivc.vbp
Esempio MQGET e MQPUT semplice (server)	mqtrivs.vbp
Inserimento e ottenimento di stringhe e strutture definite dall'utente utilizzando MQPUT e MQGET	strings.vbp
Utilizzo di strutture PCF per avviare e arrestare un canale	pcfsamp.vbp
Creazione di una coda utilizzando MQAI	amqsaicq.vbp
Elenco delle code di un gestore code utilizzando MQAI	amqsailq.vbp
Monitoraggio degli eventi utilizzando MQAI	amqsaiem.vbp

Preparazione ed esecuzione dei programmi di esempio

Configurare il gestore code per accettare in modo sicuro le richieste di connessioni in entrata dalle applicazioni in esecuzione in modalità client.

Prima di iniziare

Verificare che il gestore code esista già e che sia stato avviato. Determinare se i record di autenticazione di canale sono già abilitati come segue:

```
DISPLAY QMGR CHLAUTH
```

Questa attività prevede che i record di autenticazione di canale siano abilitati. Se si tratta di un gestore code utilizzato da altri utenti e applicazioni, la modifica di questa impostazione influirà su tutti gli altri utenti e applicazioni. Se il gestore code non utilizza i record di autenticazione di canale, il passo “4” a pagina 111 può essere sostituito con un metodo di autenticazione alternativo (ad esempio, un'uscita di sicurezza) che imposta MCAUSER su *non - privileged - user - id* che si otterrà nel passo “1” a pagina 111.

È necessario conoscere il nome del canale che l'applicazione prevede di utilizzare in modo che l'applicazione possa utilizzare il canale. È inoltre necessario conoscere quali oggetti, ad esempio code o argomenti, l'applicazione prevede di utilizzare in modo che l'applicazione possa utilizzarli.

Informazioni su questa attività

Questa attività crea un ID utente non privilegiato da utilizzare per un'applicazione client che si connette al gestore code. L'accesso viene concesso all'applicazione client solo per essere in grado di utilizzare il canale di cui ha bisogno e la coda di cui ha bisogno utilizzando questo ID utente.

Procedura

1. Ottenere un ID utente sul sistema su cui è in esecuzione il gestore code. Per questa attività questo ID utente non deve essere un utente di gestione privilegiato. Questo ID utente sarà l'autorizzazione con cui la connessione client verrà eseguita sul gestore code.
2. Avviare un programma listener con i seguenti comandi dove:

   qmgr è il nome del gestore code
   nnnn è il numero di porta scelto

   a) Per sistemi UNIX e Windows :

   ```
   runmqlsr -t tcp -m qmgr -p nnnn
   ```

3. Se l'applicazione utilizza il SISTEMA SYSTEM.DEF.SVRCONN quindi questo canale è già definito. Se l'applicazione utilizza un altro canale, crearlo immettendo il seguente comando MQSC:

   ```
   DEFINE CHANNEL('channel-name') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Channel for use by sample programs')
   ```

   channel - name è il nome del tuo canale.
4. Creare una regola di autenticazione di canale consentendo solo all'indirizzo IP del sistema client di utilizzare il canale immettendo il comando MQSC:

   ```
   SET CHLAUTH('channel-name') TYPE(ADDRESSMAP) ADDRESS('client-machine-IP-address') +
   MCAUSER('non-privileged-user-id')
   ```

   channel - name è il nome del tuo canale.
   client - machine - IP - address è l'indirizzo IP del sistema client.
   Se l'applicazione client di esempio è in esecuzione sulla stessa macchina del gestore code, utilizzare l'indirizzo IP '127.0.0.1' se l'applicazione si connetterà utilizzando 'localhost'. Se diverse macchine client si collegheranno, è possibile utilizzare un modello o un intervallo invece di un singolo indirizzo IP. Consultare Indirizzi IP generici per i dettagli.
   non - privileged - user - id è l'ID utente ottenuto nel passo “1” a pagina 111
5. Se l'applicazione utilizza il SISTEMA SYSTEM.DEFAULT.LOCAL.QUEUE questa coda è già definita. Se l'applicazione utilizza un'altra coda, crearla emettendo il comando MQSC:

   ```
   DEFINE QLOCAL('queue-name') DESCR('Queue for use by sample programs')
   ```

   queue - name è il nome della coda.
6. Concedere l'accesso per connettersi e interrogare il gestore code:

   a) Per sistemi UNIX eWindows immettono i comandi MQSC:

```
SET AUTHREC OBJTYPE(QMGR) PRINCIPAL('non-privileged-user-id') +
AUTHADD(CONNECT, INQ)
```

non - privileged - user - id è l'ID utente ottenuto nel passo “1” a pagina 111

7. Se l'applicazione è un'applicazione point - to - point, ovvero utilizza le code, concede l'accesso per consentire l'interrogazione e l'inserimento e il richiamo dei messaggi utilizzando la coda dall'ID utente da utilizzare, immettendo i comandi MQSC:
   a) Per sistemi UNIX e Windows immettono i comandi MQSC:

```
SET AUTHREC PROFILE('queue-name') OBJTYPE(QUEUE) +
PRINCIPAL('non-privileged-user-id') AUTHADD(PUT, GET, INQ, BROWSE)
```

queue - name è il nome della coda.
non - privileged - user - id è l'ID utente ottenuto nel passo “1” a pagina 111

8. Se l'applicazione è un'applicazione di pubblicazione / sottoscrizione, ossia utilizza gli argomenti, concedere l'accesso per consentire la pubblicazione e la sottoscrizione utilizzando l'argomento dall'ID utente da utilizzare, immettendo i comandi MQSC:
   a) Per sistemi UNIX e Windows immettono i comandi MQSC:

```
SET AUTHREC PROFILE('SYSTEM.BASE.TOPIC') OBJTYPE(TOPIC) +
PRINCIPAL('non-privileged-user-id') AUTHADD(PUB, SUB)
```

non - privileged - user - id è l'ID utente ottenuto nel passo “1” a pagina 111
Ciò fornirà a *non - privileged - user - id* l'accesso a qualsiasi argomento nella struttura ad albero degli argomenti, in alternativa, è possibile definire un oggetto argomento utilizzando **DEFINE TOPIC** e concedere gli accessi solo alla parte della struttura ad albero degli argomenti a cui fa riferimento tale oggetto argomento. Per i dettagli, consultare Controllo dell'accesso utente agli argomenti .

Operazioni successive

L'applicazione client può ora connettersi al gestore code e inserire o richiamare i messaggi utilizzando la coda.

Attività correlate

Concessione dell'accesso a un oggetto WebSphere MQ su sistemi UNIX o Linux e Windows

Riferimenti correlati

SET CHLAUTH
Definire il canale
DEFINE QLOCAL
SET AUTHREC

Preparazione ed esecuzione di programmi di esempio su sistemi UNIX

Tabella 18. Dove individuare gli esempi per WebSphere MQ su sistemi UNIX and Linux

Contenuto	**Cartella**
file di origine	*MQ_INSTALLATION_PATH*/samp
file di origine del gestore code di messaggi non instradabili	*MQ_INSTALLATION_PATH*/samp/dlq
file eseguibili	*MQ_INSTALLATION_PATH*/samp/bin
MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .	

I file di esempio di WebSphere MQ su sistemi UNIX and Linux si trovano nelle directory elencate in Tabella 18 a pagina 112 se i valori predefiniti sono stati utilizzati al momento dell'installazione. Per eseguire gli esempi, utilizzare le versioni eseguibili fornite o compilare le versioni di origine come qualsiasi altra

applicazione, utilizzando un compilatore ANSI. Per informazioni su come eseguire questa operazione, consultare “Esecuzione dei programmi di esempio” a pagina 113.

Preparazione ed esecuzione di programmi di esempio su sistemi Windows

Tabella 19. Dove trovare gli esempi per WebSphere MQ per Finestre

Contenuto	**Cartella**
Codice sorgente C	*MQ_INSTALLATION_PATH*\Tools\C\Esempi
Codice di origine per l'esempio di gestore di lettere non recapitate	*MQ_INSTALLATION_PATH*\Tools\C\Esempi \DLQ
Codice sorgente COBOL	*MQ_INSTALLATION_PATH*\Tools\Cobol \ Esempi
File eseguibili C [1]	*MQ_INSTALLATION_PATH*\ Tools\C\Samples \ Bin (versioni a 32 bit) *MQ_INSTALLATION_PATH*\ Tools\C\Samples\Bin64 (versioni a 64 bit)
File MQSC di esempio	*MQ_INSTALLATION_PATH*\Tools\MQSC\Esempi
Codice sorgente Visual Basic	*MQ_INSTALLATION_PATH*\Tools\VB\SampVB6
Esempi .NET	*MQ_INSTALLATION_PATH*\Tools\dotnet \ Esempi

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Nota:

1. Sono disponibili versioni a 64 - bit di alcuni esempi di file eseguibili C.

I file di esempio di WebSphere MQ per Windows si trovano nelle directory elencate in Tabella 19 a pagina 113 se i valori predefiniti sono stati utilizzati al momento dell'installazione; l'unità di installazione ha il valore predefinito < c:>. Per eseguire gli esempi, utilizzare le versioni eseguibili fornite o compilare le versioni di origine come si farebbe con qualsiasi altra applicazione WebSphere MQ per Windows . Per informazioni su come svolgere questa procedura, consultare “Esecuzione dei programmi di esempio” a pagina 113.

Esecuzione dei programmi di esempio

Considerare l'utilizzo di questo argomento quando si eseguono programmi di esempio su piattaforme differenti.

Prima di poter eseguire uno qualsiasi dei programmi di esempio, creare un gestore code e impostare le definizioni predefinite. Ciò è spiegato in Amministrazione.

Su Windows, UNIX e piattaforme Linux

Gli esempi necessitano di una serie di code da gestire. Utilizzare le proprie code oppure eseguire il file MQSC di esempio amqscos0.tst per creare una serie.

Per eseguire questa operazione sui sistemi UNIX and Linux , immettere:

- `runmqsc QManagerName <amqscos0.tst >/tmp/sampobj.out`

Controllare il file `sampobj.out` per assicurarsi che non vi siano errori.

Per eseguire questa operazione sui sistemi Windows , immettere:

- `runmqsc QManagerName  <amqscos0.tst > sampobj.out`

Controllare il file `sampobj.out` per assicurarsi che non vi siano errori. Questo file è nella directory corrente.

È ora possibile eseguire applicazioni di esempio. Immettere il nome dell'applicazione di esempio seguito da qualsiasi parametro, ad esempio:

- `amqsput myqueue qmanagername`

dove `myqueue` è il nome della coda in cui verranno inseriti i messaggi e `qmanagername` è il gestore code proprietario di `myqueue`.

Consultare la descrizione dei singoli esempi per informazioni sui parametri previsti da ciascuno di essi.

Lunghezza del nome della coda

Per i programmi di esempio COBOL, quando si passano i nomi delle code come parametri, è necessario fornire 48 caratteri, se necessario con spazi vuoti. Qualsiasi elemento diverso da 48 caratteri causa il malfunzionamento del programma con codice di errore 2085.

Esempi Inquire, Set ed Echo

Per gli esempi Inquire, Set ed Echo, le definizioni di esempio attivano le versioni C di questi esempi.

Se si desiderano le versioni COBOL è necessario modificare le definizioni di processo:

- SYSTEM.SAMPLE.INQPROCESS
- SYSTEM.SAMPLE.SETPROCESS
- SYSTEM.SAMPLE.ECHOPROCESS

Su sistemi Windows, UNIX and Linux eseguire questa operazione modificando il file `amqscos0.tst` e modificando i nomi dei file eseguibili C con i nomi dei file eseguibili COBOL prima di utilizzare il comando **`runmqsc`** , come mostrato in precedenza.

Il programma di esempio di uscita API

L'uscita API di esempio genera una traccia MQI per un file specificato dall'utente con un prefisso definito nella variabile di ambiente MQAPI_TRACE.

Per ulteriori informazioni sulle uscite API, consultare “Scrittura e compilazione delle uscite API” a pagina 390.

Sorgente
amqsaxe0.c

Binario
amqsaxe

Configurazione per l'uscita di esempio

1. Aggiungere quanto segue al file qm.ini .

   Piattaforme diverse da Windows

   ```
   ApiExitLocal:
      Sequence=100
      Function=EntryPoint
      Module=MQ_INSTALLATION_PATH/samp/bin/amqsaxe
      Name=SampleApiExit
   ```

   dove *MQ_INSTALLATION_PATH* rappresenta la directory in cui è installato IBM WebSphere MQ .

   Windows

   ```
   ApiExitLocal:
      Sequence=100
      Function=EntryPoint
      Module=MQ_INSTALLATION_PATH\Tools\c\Samples\bin\amqsaxe
      Name=SampleApiExit
   ```

dove *MQ_INSTALLATION_PATH* rappresenta la directory in cui è installato IBM WebSphere MQ .

2. Imposta la variabile di ambiente

```
MQAPI_TRACE_LOGFILE=/tmp/MqiTrace
```

3. Esegui la tua applicazione.

   I file di emissione vengono creati nella directory /tmp con nomi come: `MqiTrace.<pid>.<tid>.log`

Il programma di esempio di consumo asincrono

Il programma di esempio amqscbf dimostra l'utilizzo di MQCB e MQCTL per utilizzare i messaggi da più code in modo asincrono.

amqscbf viene fornito come codice sorgente C e un client binario e un server eseguibile sulle piattaforme Windows, UNIX and Linux .

Il programma viene avviato dalla riga comandi e utilizza i seguenti parametri facoltativi:

```
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
```

Fornire più di un nome coda per leggere i messaggi da più code (l'esempio supporta un massimo di dieci code).

Nota: *Tipo di riconnessione* è valido solo per i programmi client.

Esempio

L'esempio mostra amqscbf eseguito come un programma server che legge un messaggio da QL1 e viene quindi arrestato.

Utilizzare WebSphere MQ Explorer per inserire un messaggio di prova su QL1. Arrestare il programma premendo Invio.

```
C:\>amqscbf QL1
Sample AMQSCBF0 start

Press enter to end
Message Call (9 Bytes) :
Message 1

Sample AMQSCBF0 end
```

Cosa dimostra amqscbf

L'esempio mostra come leggere i messaggi da più code in ordine di arrivo. Ciò richiederebbe molto più codice utilizzando MQGET sincrono. In caso di utilizzo asincrono, non è richiesto alcun polling e la gestione del thread e della memoria viene eseguita da WebSphere MQ. Un esempio "reale" dovrebbe occuparsi degli errori; nell'esempio gli errori vengono scritti nella console.

Il codice di esempio contiene la seguente procedura:

1. Definire la singola funzione di callback di consumo del messaggio,

```
void MessageConsumer(MQHCONN      hConn,
                     MQMD    * pMsgDesc,
                     MQGMO   * pGetMsgOpts,
                     MQBYTE  * Buffer,
                     MQCBC   * pContext)
{ ... }
```

2. Connettersi al gestore code,

```
MQCONNX(QMName,&cno,&Hcon,&CompCode,&CReason);
```

3. Aprire le code di immissione e associarle alla funzione di callback `MessageConsumer` ,

```
MQOPEN(Hcon,&od,O_options,&Hobj,&OpenCode,&Reason);
cbd.CallbackFunction = MessageConsumer;
MQCB(Hcon,MQOP_REGISTER,&cbd,Hobj,&md,&gmo,&CompCode,&Reason);
```

   `cbd.CallbackFunction` non deve essere impostato per ciascuna coda; è un campo di sola immissione. Ma è possibile associare una funzione di callback differente a ciascuna coda.

4. Avviare l'utilizzo dei messaggi,

```
MQCTL(Hcon,MQOP_START,&ctlo,&CompCode,&Reason);
```

5. Attendere che l'utente abbia premuto Invio e quindi interrompere l'utilizzo dei messaggi,

```
MQCTL(Hcon,MQOP_STOP,&ctlo,&CompCode,&Reason);
```

6. Infine, disconnettersi dal gestore code,

```
MQDISC(&Hcon,&CompCode,&Reason);
```

Il programma di esempio Put asincrono

Informazioni sull'esecuzione dell'esempio amqsapt e sulla progettazione del programma di esempio Asynchronous Put.

Il programma di esempio di inserimento asincrono inserisce i messaggi su una coda utilizzando la chiamata MQPUT asincrona e quindi richiama le informazioni sullo stato utilizzando la chiamata MQSTAT. Consultare “Funzioni dimostrate nei programmi di esempio” a pagina 98 per il nome di questo programma su piattaforme differenti.

Esecuzione dell'esempio amqsapt

Questo programma impiega fino a 6 parametri:

1. Il nome della coda di destinazione (obbligatorio)
2. Il nome del gestore code (facoltativo)
3. Opzioni di apertura (facoltativo)
4. Opzioni di chiusura (facoltativo)
5. Il nome del gestore code di destinazione (facoltativo)
6. Il nome della coda dinamica (facoltativo)

Se non viene specificato un gestore code, amqsapt si connette al gestore code predefinito.

Progettazione del programma di esempio Put asincrono

Il programma utilizza la chiamata MQOPEN con le opzioni di output fornite o con le opzioni MQOO_OUTPUT e MQOO_FAIL_IF_QUIESCING per aprire la coda di destinazione per l'inserimento dei messaggi.

Se non riesce ad aprire la coda, il programma emette un messaggio di errore contenente il codice motivo restituito dalla chiamata MQOPEN. Per semplificare il programma, in questa e nelle successive chiamate MQI, il programma utilizza i valori predefiniti per molte opzioni.

Per ogni riga di input, il programma legge il testo in un buffer e utilizza la chiamata MQPUT con MQPMO_ASYNC_RESPONSE per creare un messaggio datagramma contenente il testo di tale riga e inserirlo in maniera asincrona nella coda di destinazione. Il programma continua fino a quando non

raggiunge la fine dell'input o la chiamata MQPUT ha esito negativo. Se il programma raggiunge la fine dell'input, chiude la coda utilizzando la chiamata MQCLOSE.

Il programma, quindi, emette la chiamata MQSTAT, restituendo una struttura MQSTS e visualizza i messaggi contenenti il numero di messaggi immessi correttamente, il numero di messaggi immessi con un'avvertenza e il numero di errori.

Programmi di esempio Sfoglia

I programmi di esempio Sfoglia consultano i messaggi su una coda utilizzando la chiamata MQGET.

Consultare "Funzioni dimostrate nei programmi di esempio" a pagina 98 per i nomi di questi programmi.

Progettazione del programma di esempio Sfoglia

Il programma apre la coda di destinazione utilizzando la chiamata MQOPEN con l'opzione MQOO_BROWSE. Se non riesce ad aprire la coda, il programma emette un messaggio di errore contenente il codice motivo restituito dalla chiamata MQOPEN.

Per ogni messaggio sulla coda, il programma utilizza la chiamata MQGET per copiare il messaggio dalla coda, quindi visualizza i dati contenuti nel messaggio. La chiamata MQGET utilizza queste opzioni:

MQGMO_BROWSE_SUCCESSIVO
Dopo la chiamata MQOPEN, il cursore di esplorazione viene posizionato logicamente prima del primo messaggio nella coda, quindi questa opzione determina la restituzione del ***primo*** messaggio quando la chiamata viene effettuata per la prima volta.

MQGMO_NO_WAIT
Il programma non attende se non ci sono messaggi nella coda.

MQGMO_ACCEPT_TRUNCATED_MSG
La chiamata MQGET specifica un buffer di dimensione fissa. Se un messaggio è più lungo di questo buffer, il programma visualizza il messaggio troncato, insieme ad un'avvertenza che il messaggio è stato troncato.

Il programma dimostra come è necessario cancellare i dati dei campi *MsgId* e *CorrelId* della struttura MQMD dopo ogni chiamata MQGET, poiché la chiamata imposta questi campi sui valori contenuti nel messaggio richiamato. La cancellazione di questi campi significa che le chiamate MQGET successive richiamano i messaggi nell'ordine in cui sono conservati nella coda.

Il programma continua fino alla fine della coda; la chiamata MQGET restituisce il codice motivo MQRC_NO_MSG_AVAILABLE e il programma visualizza un messaggio di avviso. Se la chiamata MQGET non riesce, il programma visualizza un messaggio di errore che contiene il codice di errore.

Il programma chiude quindi la coda utilizzando la chiamata MQCLOSE.

Sistemi UNIX, Linux e Windows

Utilizzare questo argomento per informazioni su Sfoglia programmi di esempio su sistemi UNIX, Linux e Windows .

La versione C del programma richiede 2 parametri

1. Il nome della coda di origine (necessario)
2. Il nome del gestore code (facoltativo)

Se un gestore code non è specificato, si connette a quello predefinito. Ad esempio, immettere uno dei seguenti:

- `amqsgbr myqueue qmanagername`
- `amqsgbrc myqueue qmanagername`
- `amq0gbr0 myqueue`

dove `myqueue` è il nome della coda da cui verranno visualizzati i messaggi e `qmanagername` è il gestore code proprietario di `myqueue`.

Se si omette qmanagername, quando si esegue l'esempio C, si presuppone che il gestore code predefinito sia proprietario della coda.

La versione COBOL non ha alcun parametro. Si connette al gestore code predefinito e quando viene eseguito viene richiesto:

```
Please enter the name of the target queue
```

Vengono visualizzati solo i primi 50 caratteri di ciascun messaggio, seguiti da - - - truncated in questo caso.

Il programma di esempio Browser

Il programma di esempio Browser legge e scrive sia il descrittore del messaggio che i campi del contenuto del messaggio di tutti i messaggi su una coda.

Il programma di esempio è scritto come un programma di utilità, non solo per dimostrare una tecnica. Consultare "Funzioni dimostrate nei programmi di esempio" a pagina 98 per i nomi di questi programmi.

Questo programma utilizza i seguenti parametri:

1. Il nome della coda di origine
2. Il nome del gestore code
3. Un parametro facoltativo per le proprietà.

I primi due parametri di immissione per questo programma sono obbligatori. Ad esempio, avviare il programma in uno dei seguenti modi:

- amqsbcg myqueue qmanagername
- amqsbcgc myqueue qmanagername

dove myqueue è il nome della coda in cui verranno esaminati i messaggi e qmanagername è il gestore code proprietario di myqueue.

Legge ogni messaggio dalla coda e scrive quanto segue in stdout:

- Campi del descrittore del messaggio formattato
- Dati del messaggio (dump in formato esadecimale e, dove possibile, in formato carattere)

I valori consentiti per il parametro di proprietà sono:

Valore	Funzionamento
0	Comportamento predefinito, come per V6. Le proprietà che vengono distribuite all'applicazione dipendono dall'attributo della coda *PropertyControl* da cui viene richiamato il messaggio.

1	Un handle del messaggio viene creato e utilizzato con MQGET. Le proprietà del messaggio, eccetto quelle contenute nel descrittore del messaggio (o estensione), vengono visualizzate in modo simile al descrittore del messaggio. Ad esempio:
	`****Message properties****`
	`  <property name> : <property value>`
	Oppure, se non sono disponibili proprietà:
	`****Message properties****`
	`  None`
	I valori numerici vengono visualizzati utilizzando printf, i valori stringa vengono racchiusi tra virgolette singole e le stringhe di byte vengono racchiuse tra virgolette singole e X, come per il descrittore del messaggio.

Valore	Funzionamento
2	Viene specificato MQGMO_NO_PROPERTIES, in modo che vengano restituite solo le proprietà del descrizione del messaggio.
3	Viene specificato MQGMO_PROPERTIES_FORCE_MQRFH2 , in modo che tutte le proprietà vengano restituite nei dati del messaggio.
4	Viene specificato MQGMO_PROPERTIES_COMPATIBILITY, in modo che tutte le proprietà possano essere restituite a seconda che sia inclusa una proprietà della versione 6, altrimenti le proprietà vengono eliminate.

Il programma è limitato alla stampa dei primi 65535 caratteri del messaggio e non riesce con il motivo *messaggio troncato* se viene letto un messaggio più lungo.

Consultare Amministrazione per un esempio dell'output di questo programma di utilità.

L'esempio di transazione CICS

Viene fornito un programma di transazione CICS di esempio, denominato amqscic0.ccs per il codice sorgente e amqscic0 per la versione eseguibile. È possibile creare transazioni utilizzando le funzionalità CICS standard.

Consultare “Creazione di un'applicazione IBM WebSphere MQ” a pagina 431 per i dettagli sui comandi necessari per la propria piattaforma.

La transazione legge i messaggi dalla coda di trasmissione SYSTEM.SAMPLE.CICS.WORKQUEUE sul gestore code predefinito e le colloca nella coda locale, il nome del quale è contenuto nell'intestazione di trasmissione del messaggio. Eventuali errori vengono inviati alla coda SYSTEM.SAMPLE.CICS.DLQ.

Nota: È possibile utilizzare uno script MQSC di esempio amqscic0.tst per creare queste code e code di input di esempio.

Il programma di esempio Connect

Il programma di esempio Connect consente di esplorare la chiamata MQCONNX e le sue opzioni da un client. L'esempio si connette al gestore code utilizzando la chiamata MQCONNX, richiede il nome del gestore code utilizzando la chiamata MQINQ e lo visualizza. Inoltre, vengono fornite informazioni sull'esecuzione dell'esempio amqscnxc.

Nota: Il programma di esempio Connect è un esempio client. È possibile compilarlo ed eseguirlo su un server, ma la funzione è significativa solo su un client e vengono forniti solo i file eseguibili dal client.

Esecuzione dell'esempio amqscnxc

La sintassi della riga comandi del programma di esempio Connect è:

```
amqscnxc [-x ConnName [-c SvrconnChannelName]] [QMgrName]
```

I parametri sono facoltativi e il loro ordine non è importante ad eccezione di QMgrName, che, se specificato, deve essere l'ultimo. I parametri sono:

ConnName
Il nome della connessione TCP/IP del gestore code del server

Nome SvrconnChannel
Il nome del canale di connessione server

QMgrName
Il nome del gestore code di destinazione

Se non si specifica il nome connessione TCP/IP, MQCONNX viene emesso con *ClientConnPtr* impostato su NULL. Se si specifica il nome connessione TCP/IP ma non il canale di connessione server (l'inverso

non è consentito), l'esempio utilizza il nome SYSTEM.DEF.SVRCONN. Se non si specifica il gestore code di destinazione, l'esempio si connette al gestore code in ascolto sul nome della connessione TCP/IP fornito.

Nota: Se si immette un punto interrogativo come unico parametro o se si immettono parametri non corretti, si riceve un messaggio che spiega come utilizzare il programma.

Se si esegue l'esempio senza alcuna opzione della riga comandi, il contenuto della variabile di ambiente MQSERVER viene utilizzato per determinare le informazioni di connessione. (In questo esempio MQSERVER è impostata su SYSTEM.DEF.SVRCONN/TCP/machine.site.company.com.) Viene visualizzato il seguente output:

```
Sample AMQSCNXC start
Connecting to the default queue manager
with no client connection information specified.
Connection established to queue manager machine

Sample AMQSCNXC end
```

Se si esegue l'esempio e si fornisce un nome di connessione TCP/IP e un nome di canale di connessione server, ma nessun nome di gestore code di destinazione, come riportato di seguito:

```
amqscnxc -x machine.site.company.com -c SYSTEM.ADMIN.SVRCONN
```

viene utilizzato il nome gestore code predefinito e viene visualizzato un output simile al seguente:

```
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
```

Se si esegue l'esempio e si fornisce un nome connessione TCP/IP e un nome gestore code di destinazione, come segue:

```
amqscnxc -x machine.site.company.com MACHINE
```

viene visualizzato un output simile al seguente:

```
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
```

Il programma di esempio Conversione dati

Il programma di esempio di conversione dati è una struttura di una routine di uscita di conversione dati. Scopri la progettazione del campione di conversione dati.

Consultare “Funzioni dimostrate nei programmi di esempio” a pagina 98 per i nomi di questi programmi.

Progettazione del campione di conversione dati

Ogni routine di uscita di conversione dati converte un singolo formato di messaggio denominato. Questa struttura è intesa come un wrapper per i frammenti di codice generati dal programma di utilità di generazione dell'uscita di conversione dati.

Il programma di utilità produce un frammento di codice per ogni struttura di dati; molte di queste strutture costituiscono un formato, quindi diversi frammenti di codice vengono aggiunti a questa struttura per produrre una routine per eseguire la conversione dei dati dell'intero formato.

Il programma verifica quindi se la conversione è riuscita o meno e restituisce i valori richiesti al chiamante.

Esempi di coordinamento database

Vengono forniti due esempi che dimostrano come WebSphere MQ può coordinare gli aggiornamenti WebSphere MQ e gli aggiornamenti del database all'interno della stessa unità di lavoro.

Questi esempi sono:

1. AMQSXAS0 (in C) o AMQ0XAS0 (in COBOL), che aggiorna un singolo database all'interno di una unità di lavoro WebSphere MQ .
2. AMQSXAG0 (in C) o AMQ0XAG0 (in COBOL), AMQSXAB0 (in C) o AMQ0XAB0 (in COBOL) e AMQSXAF0 (in C) o AMQ0XAF0 (in COBOL), che insieme aggiornano due database all'interno di un'unità di lavoro WebSphere MQ , mostrando come è possibile accedere a più database. Questi esempi vengono forniti per mostrare l'utilizzo della chiamata MQBEGIN, le chiamate SQL miste e WebSphere MQ e dove e quando connettersi a un database.

Figura 18 a pagina 121 mostra il modo in cui gli esempi forniti vengono utilizzati per aggiornare i database:

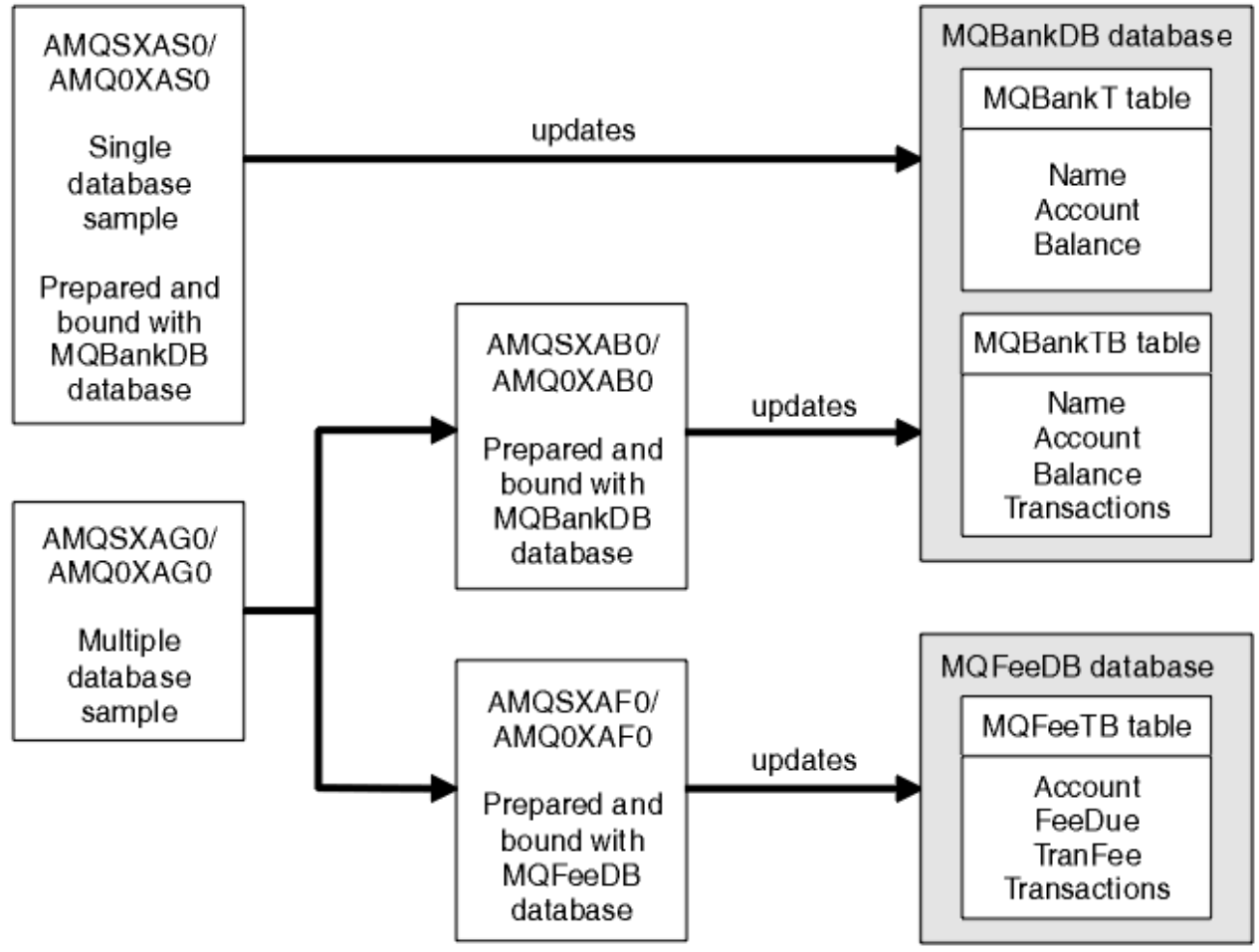


Figura 18. Gli esempi di coordinamento del database

I programmi leggono un messaggio da una coda (nel punto di sincronizzazione), quindi, utilizzando le informazioni nel messaggio, ottengono le informazioni rilevanti dal database e le aggiornano. Il nuovo stato del database viene quindi stampato.

La logica del programma è la seguente:

1. Utilizzare il nome della coda di input dall'argomento del programma
2. Connettersi al gestore code predefinito (o, facoltativamente, al nome fornito in C) utilizzando MQCONN
3. Aprire una coda (utilizzando MQOPEN) per l'input mentre non vi sono errori
4. Avviare un'unità di lavoro utilizzando MQBEGIN
5. Richiama il successivo messaggio (utilizzando MQGET) dalla coda nel punto di sincronizzazione
6. Ottieni informazioni dai database
7. Aggiorna informazioni dai database
8. Esegui il commit delle modifiche utilizzando MQCMIT
9. Stampare le informazioni aggiornate (nessun messaggio disponibile conta come errore e il loop termina)
10. Chiudere la coda utilizzando MQCLOSE
11. Disconnetti dalla coda utilizzando MQDISC

I cursori SQL vengono utilizzati negli esempi, in modo che le letture dai database (ossia, più istanze) vengano bloccate mentre un messaggio viene elaborato, consentendo l'esecuzione simultanea di più istanze di questi programmi. I cursori sono esplicitamente aperti, ma implicitamente chiusi dalla chiamata MQCMIT.

Il singolo esempio di database (AMQSXAS0 o AMQ0XAS0) non dispone di istruzioni SQL CONNECT e la connessione al database viene effettuata implicitamente da WebSphere MQ con la chiamata MQBEGIN. L'esempio di database multiplo (AMQSXAG0 o AMQ0XAG0, AMQSXAB0 o AMQ0XAB0e AMQSXAF0 o AMQ0XAF0) ha istruzioni SQL CONNECT, poiché alcuni prodotti database consentono una sola connessione attiva. Se questo non è il caso del prodotto database o se si sta accedendo a un singolo database in più prodotti database, è possibile rimuovere le istruzioni SQL CONNECT.

Gli esempi vengono preparati con il prodotto database IBM DB2 , quindi potrebbe essere necessario modificarli per lavorare con altri prodotti database.

Il controllo errori SQL utilizza le routine in UTIL.C e CHECKERR.CBL fornito da DB2. Questi devono essere compilati o sostituiti prima della compilazione e del collegamento.

Nota: Se si utilizza l'origine Micro Focus COBOL CHECKERR.MFC per il controllo degli errori SQL, è necessario modificare l'ID programma in maiuscolo, ossia CHECKERR, affinché AMQ0XAS0 sia collegato correttamente.

Creazione di database e tabelle

Creare i database e le tabelle prima di compilare gli esempi.

Per creare i database, utilizzare il metodo usuale per il prodotto database, ad esempio:

```
DB2 CREATE DB MQBankDB
DB2 CREATE DB MQFeeDB
```

Creare le tabelle utilizzando le seguenti istruzioni SQL:

In 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));
```

In 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.
```

Immettere i dati nelle tabelle utilizzando le istruzioni SQL nel modo seguente:

```
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);
.
```

Nota: Per COBOL, utilizzare le stesse istruzioni SQL ma aggiungere END_EXEC alla fine di ogni riga.

Precompilazione, compilazione e collegamento degli esempi

Informazioni sulla precompilazione, la compilazione e il collegamento di esempi in C e COBOL.

Precompilare i file .SQC (in C) e .SQB (in COBOL) e collegarli al database appropriato per produrre i file .C o .CBL. A tale scopo, utilizzare il metodo tipico per il prodotto database.

Precompilazione in 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
```

Precompilazione in 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
```

Compilazione e collegamento

I seguenti comandi di esempio utilizzano i simboli *<DB2TOP>* e *MQ_INSTALLATION_PATH*. *<DB2TOP>* rappresenta la directory di installazione per il prodotto DB2 . *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ .

- Su AIX, il percorso della directory è:

  ```
  /usr/lpp/db2_05_00
  ```

- In HP-UX e Solaris, il percorso della directory è:

  ```
  /opt/IBMdb2/V5.0
  ```

- Sui sistemi Windows , il percorso della directory dipende dal percorso scelto durante l'installazione del prodotto. Se si scelgono le impostazioni predefinite, il percorso è:

  ```
  c:\sqllib
  ```

Nota: Prima di immettere il comando di collegamento sui sistemi Windows , assicurarsi che la variabile di ambiente LIB contenga i percorsi delle librerie DB2 e WebSphere MQ .

Copiare i seguenti file in una directory temporanea:

- Il file `amqsxag0.c` dall'installazione WebSphere MQ

  Nota: Questo file può essere trovato nelle seguenti directory:

  – Su sistemi UNIX and Linux :

  ```
  MQ_INSTALLATION_PATH/samp/xatm
  ```

  – Su sistemi Windows :

  ```
  MQ_INSTALLATION_PATH\tools\c\samples\xatm
  ```

- I file `.c` ottenuti precompilando i file di origine `.sqc` , `amqsxas0.sqc, amqsxaf0.sqc` e `amqsxab0.sqc`
- I file `util.c` e `util.h` dall'installazione di DB2 .

  Nota: Questi file si trovano nella directory:

```
<DB2TOP>/samples/c
```

Creare i file di oggetto per ogni file `.c` utilizzando il seguente comando del compilatore per la piattaforma che si sta utilizzando:

- 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
```

- Sistemi Windows

```
cl /c /IMQ_INSTALLATION_PATH\tools\c\include /I
<DB2TOP>\include
 <FILENAME>.c
```

Creare il file eseguibile `amqsxag0` utilizzando il seguente comando di collegamento per la piattaforma che si sta utilizzando:

- 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 Revisione 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
```

- Sistemi Windows

```
link util.obj amqsxaf0.obj amqsxab0.obj amqsxag0.obj mqm.lib db2api.lib
 /out:amqsxag0.exe
```

Creare il file eseguibile `amqsxas0` utilizzando i seguenti comandi di compilazione e collegamento per la piattaforma che si sta utilizzando:

- AIX

```
xlc_r -H512 -T512 -L<DB2TOP>/lib -ldb2
 -LMQ_INSTALLATION_PATH/lib -lmqm util.o amqsxas0.o -o amqsxas0
```

- HP-UX Revisione 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
```

- Sistemi Windows

```
link util.obj amqsxas0.obj mqm.lib db2api.lib /out:amqsxas0.exe
```

Ulteriori informazioni

Se si sta utilizzando AIX o HP-UX e si desidera accedere a Oracle, utilizzare il compilatore xlc_r e il collegamento a libmqm_r.a.

Esecuzione degli esempi

Utilizzare queste informazioni per informazioni su come configurare il gestore code prima di eseguire gli esempi di coordinamento del database su C e COBOL.

Prima di eseguire gli esempi, configurare il gestore code con il prodotto database che si sta utilizzando. Per informazioni su come svolgere questa procedura, consultare "Scenario 1: il gestore code esegue il coordinamento" a pagina 43.

I seguenti titoli forniscono informazioni su come eseguire gli esempi in C e COBOL:

- "Esempi C" a pagina 126
- "Esempi COBOL" a pagina 127

Esempi C

I messaggi devono essere nel seguente formato per essere letti da una coda:

```
UPDATE Balance change=nnn WHERE Account=nnn
```

AMQSPUT può essere utilizzato per inserire i messaggi nella coda.

Gli esempi di coordinamento del database richiedono due parametri:

1. Nome coda (obbligatorio)
2. Nome gestore code (facoltativo)

Supponendo che sia stato creato e configurato un gestore code per il singolo database di esempio denominato singDBQM, con una coda denominata singDBQ, si incrementa l'account di Mr Fred Bloggs di 50 come segue:

```
  AMQSPUT singDBQ singDBQM
```

Quindi, immettere il seguente messaggio:

```
  UPDATE Balance change=50 WHERE Account=1
```

È possibile inserire più messaggi nella coda.

```
  AMQSXAS0 singDBQ singDBQM
```

Viene quindi stampato lo stato aggiornato del conto di Fred Bloggs.

Supponendo di aver creato e configurato un gestore code per l'esempio a più database denominato multDBQM, con una coda denominata multDBQ, si decrementa l'account di Mary Brown di 75 come segue:

```
AMQSPUT multDBQ multDBQM
```

Quindi, immettere il seguente messaggio:

```
UPDATE Balance change=-75 WHERE Account=3
```

È possibile inserire più messaggi nella coda.

```
AMQSXAG0 multDBQ multDBQM
```

Viene quindi stampato lo stato aggiornato del conto di Mary Brown.

Esempi COBOL

I messaggi devono essere nel seguente formato per essere letti da una coda:

```
UPDATE Balance change=snnnnnnnn WHERE Account=nnnnnnnn
```

Per semplicità, il `Balance change` deve essere un numero di otto caratteri con segno e il `Account` deve essere un numero di otto caratteri.

L'esempio AMQSPUT può essere utilizzato per inserire i messaggi nella coda.

Gli esempi non utilizzano parametri e utilizzano il gestore code predefinito. Può essere configurato per eseguire solo uno degli esempi alla volta. Supponendo che sia stato configurato il gestore code predefinito per il singolo esempio di database, con una coda denominata singDBQ, si incrementa l'account di Fred Bloggs di 50 nel modo seguente:

```
AMQSPUT singDBQ
```

Quindi, immettere il seguente messaggio:

```
UPDATE Balance change=+00000050 WHERE Account=00000001
```

È possibile inserire più messaggi nella coda:

```
AMQ0XAS0
```

Immettere il nome della coda:

```
singDBQ
```

Viene quindi stampato lo stato aggiornato del conto di Fred Bloggs.

Supponendo che sia stato configurato il gestore code predefinito per l'esempio di più database, con una coda denominata multDBQ, si decrementa l'account di Mary Brown di 75 come segue:

```
AMQSPUT multDBQ
```

Quindi, immettere il seguente messaggio:

```
UPDATE Balance change=-00000075 WHERE Account=00000003
```

È possibile inserire più messaggi nella coda:

```
AMQ0XAG0
```

Immettere il nome della coda:

```
multDBQ
```

Viene quindi stampato lo stato aggiornato del conto di Mary Brown.

Esempio di gestore code di messaggi non instradabili

Viene fornito un gestore code di messaggi non instradabili di esempio, il nome della versione eseguibile è amqsdlq. Se si desidera un gestore code di messaggi non instradabili diverso da RUNMQDLQ, l'origine dell'esempio è disponibile per essere utilizzata come base.

L'esempio è simile al gestore di lettere non recapitate fornito all'interno del prodotto, ma la traccia e la notifica degli errori sono differenti. Sono disponibili due variabili di ambiente:

TRACCIA_ODQ
Impostare su YES o yes per attivare la traccia

ODQ_MSG
Impostare sul nome del file contenente i messaggi di errore e informativi. Il file fornito è denominato amqsdlq.msg.

È necessario rendere note queste variabili all'ambiente utilizzando i comandi **export** o **set** , a seconda della propria piattaforma; la funzione di traccia è disattivata utilizzando il comando **unset** .

È possibile modificare il file dei messaggi di errore, amqsdlq.msg, per adattarlo ai propri requisiti. L'esempio inserisce i messaggi in stdout, ***non*** nel file di log degli errori WebSphere MQ .

La Amministrazione o la *Guida alla gestione del sistema* per la propria piattaforma spiega come funziona il gestore dei messaggi non recapitabili e come viene eseguito.

Programma di esempio Elenco di distribuzione

L'esempio Elenco di distribuzione amqsptl0 fornisce un esempio di inserimento di un messaggio in più code di messaggi. Si basa sull'esempio MQPUT amqsput0.

Esecuzione dell'esempio Elenco di distribuzioni, amqsptl0

L'esempio Elenco di distribuzione viene eseguito in modo simile agli esempi Put.

Prende i parametri seguenti:

- I nomi delle code
- I nomi dei gestori code

Questi valori vengono immessi come coppie. Ad esempio:

```
amqsptl0 queue1 qmanagername1 queue2 qmanagername2
```

Le code vengono aperte utilizzando MQOPEN e i messaggi vengono inseriti nelle code utilizzando MQPUT. I codici di errore vengono restituiti se uno dei nomi della coda o del gestore code non viene riconosciuto.

Ricordarsi di definire i canali tra i gestori code in modo che i messaggi possano fluire tra di essi. Il programma di esempio non lo fa per voi.

Progettazione dell'esempio Elenco di distribuzione

I record di inserimento messaggi (MQPMR) specificano gli attributi dei messaggi per ciascuna destinazione. L'esempio fornisce valori per *MsgId* e *CorrelId*, e questi sovrascrivono i valori specificati nella struttura MQMD.

Il campo *PutMsgRecFields* nella struttura MQPMO indica quali campi sono presenti nei MQPMR:

```
MQLONG PutMsgRecFields=MQPMRF_MSG_ID + MQPMRF_CORREL_ID;
```

Successivamente, l'esempio assegna i record di risposta e i record oggetto. I record oggetto (MQORs) richiedono almeno una coppia di nomi e un numero pari di nomi, ovvero *ObjectName* e *ObjectQMgrName*.

La fase successiva prevede la connessione ai gestori code utilizzando MQCONN. L'esempio tenta di connettersi al gestore code associato alla prima coda in MQOR; se non riesce, passa attraverso i record dell'oggetto a turno. Si viene informati se non è possibile connettersi ad alcun gestore code e il programma si chiude.

Le code di destinazione vengono aperte tramite MQOPEN e il messaggio viene inserito in queste code mediante MQPUT. Eventuali problemi ed errori vengono riportati nei record di risposta (MQRR).

Infine, le code di destinazione vengono chiuse utilizzando MQCLOSE e il programma si disconnette dal gestore code utilizzando MQDISC. Gli stessi record di risposta vengono utilizzati per ogni chiamata che indica *CompCode* e *Reason*.

I programmi di esempio Echo

I programmi di esempio Echo riecheggiano un messaggio da una coda messaggi alla coda di risposta.

Consultare “Funzioni dimostrate nei programmi di esempio” a pagina 98 per i nomi di questi programmi.

I programmi devono essere eseguiti come programmi attivati.

Su sistemi UNIX, Linux e Windows , il loro unico input è una struttura MQTMC2 (messaggio trigger) contenente il nome di una coda di destinazione e il gestore code. La versione COBOL utilizza il gestore code predefinito.

Una volta impostata correttamente la definizione, avviare prima AMQSERV4 in un lavoro, quindi AMQSREQ4 in un'altro. È possibile utilizzare AMQSTRG4 invece di AMQSERV4, ma i potenziali ritardi di inoltro dei lavori potrebbero rendere meno semplice seguire ciò che sta accadendo.

Utilizzare i programmi di esempio Richiesta per inviare messaggi alla coda SYSTEM.SAMPLE.ECHO. I programmi di esempio Echo inviano un messaggio di risposta contenente i dati nel messaggio di richiesta alla coda di risposta specificata nel messaggio di richiesta.

Progettazione dei programmi di esempio Echo

Il programma apre la coda denominata nella struttura del messaggio trigger che è stata passata quando è stata avviata. (Per chiarezza, chiameremo questa *coda di richiesta*.) Il programma utilizza la chiamata MQOPEN per aprire questa coda per l'input condiviso.

Il programma utilizza la chiamata MQGET per rimuovere i messaggi da questa coda. Questa chiamata utilizza l'opzione MQGMO_ACCEPT_TRUNCATED_MSG, MQGMO_CONVERT e MQGMO_WAIT, con intervallo di attesa di 5 secondi. Il programma verifica il descrittore di ciascun messaggio per verificare se si tratta di un messaggio di richiesta; in caso contrario, il programma elimina il messaggio e visualizza un messaggio di avvertenza.

Per ogni riga di input, il programma legge il testo in un buffer e utilizza la chiamata MQPUT1 per inserire un messaggio di richiesta, contenente il testo di quella riga, nella coda di risposta.

Se la chiamata MQGET ha esito negativo, il programma inserisce un messaggio di report nella coda di risposta, impostando il campo *Feedback* del descrittore del messaggio sul codice di errore restituito da MQGET.

Quando non ci sono messaggi rimanenti sulla coda di richiesta, il programma chiude tale coda e si disconnette dal gestore code.

I programmi di esempio Get

I programmi di esempio Get ricevono i messaggi da una coda utilizzando la chiamata MQGET.

Consultare "Funzioni dimostrate nei programmi di esempio" a pagina 98 per i nomi di questi programmi.

Progettazione del programma di esempio Get

Il programma apre la coda di destinazione utilizzando la chiamata MQOPEN con l'opzione MQOO_INPUT_AS_Q_DEF. Se non è in grado di aprire la coda, il programma visualizza un messaggio di errore contenente il codice motivo restituito dalla chiamata MQOPEN.

Per ogni messaggio sulla coda, il programma utilizza la chiamata MQGET per rimuovere il messaggio dalla coda, quindi visualizza i dati contenuti nel messaggio. La chiamata MQGET utilizza l'opzione MQGMO_WAIT, specificando un *WaitInterval* di 15 secondi, in modo che il programma attenda per questo periodo se non è presente alcun messaggio nella coda. Se non arriva alcun messaggio prima della scadenza di questo intervallo, la chiamata ha esito negativo e restituisce il codice di errore MQRC_NO_MSG_AVAILABLE.

Il programma dimostra in che modo è necessario cancellare i campi *MsgId* e *CorrelId* della struttura MQMD dopo ogni chiamata MQGET poiché la chiamata imposta questi campi sui valori contenuti nel messaggio richiamato. La cancellazione di questi campi significa che le chiamate MQGET successive richiamano i messaggi nell'ordine in cui sono conservati nella coda.

La chiamata MQGET specifica un buffer di dimensione fissa. Se un messaggio è più lungo di questo buffer, la chiamata non riesce e il programma si arresta.

Il programma continua fino a quando la chiamata MQGET non restituisce il codice motivo MQRC_NO_MSG_AVAILABLE o la chiamata MQGET non riesce. Se la chiamata ha esito negativo, il programma visualizza un messaggio di errore che contiene il codice di errore.

Il programma chiude quindi la coda utilizzando la chiamata MQCLOSE.

Esecuzione degli esempi amqsget e amqsgetc

Ognuno di questi programmi ha due parametri:

1. Il nome della coda di origine (obbligatorio)
2. Il nome del gestore code (facoltativo)

Se un gestore code non è specificato, amqsget si connette al gestore code predefinito e amqsgetc si connette al gestore code identificato da una variabile di ambiente o dal file di definizione del canale client.

Per eseguire questi programmi, immettere uno dei seguenti:

- `amqsget myqueue qmanagername`
- `amqsgetc myqueue qmanagername`

dove `myqueue` è il nome della coda da cui il programma riceve i messaggi e `qmanagername` è il gestore code proprietario di `myqueue`.

Se si omette il `qmanagername`, i programmi assumono il valore predefinito o, nel caso del client MQI, il gestore code identificato da una variabile di ambiente o dal file di definizione del canale client.

Programmi di esempio HA (High Availability)

I programmi di esempio ad alta disponibilità **amqsghac**, **amqsphac**e **amqsmhac** utilizzano la riconnessione client automatizzata per dimostrare il recupero in caso di errore di un gestore code. **amqsfhac** verifica che un gestore code che utilizza la memoria di rete mantenga l'integrità dei dati in seguito a un errore.

I programmi **amqsghac**, **amqsphac**e **amqsmhac** vengono avviati dalla riga comandi e possono essere utilizzati in combinazione per dimostrare la riconnessione dopo l'errore di un'istanza di gestore code a più istanze.

In alternativa, è anche possibile utilizzare gli esempi **amqsghac**, **amqsphac**e **amqsmhac** per dimostrare la riconnessione del client ai gestori code a istanza singola, generalmente configurati in un gruppo di gestori code.

Per semplificare l'esempio, in modo che sia facile da configurare, vengono mostrati i programmi di esempio che si riconnettono a un gestore code a istanza singola avviato, arrestato e quindi riavviato di nuovo; consultare “Imposta e controlla il gestore code” a pagina 133.

Utilizzare **amqsfhac** in parallelo con **amqmfsck** per verificare l'integrità del file system. Per ulteriori informazioni, consultare **amqmfsck** (controllo file system) e Verifica del funzionamento del file system condiviso .

amqsphac *queueName* **[***qMgrNome***]**

- **amqsphac** è un'applicazione IBM WebSphere MQ MQI client . Inserisce una sequenza di messaggi in una coda con un ritardo di due secondi tra ciascun messaggio e visualizza gli eventi inviati al relativo gestore eventi.
- Nessun punto di sincronizzazione viene utilizzato per inserire i messaggi nella coda.
- La riconnessione può essere effettuata a qualsiasi gestore code nello stesso gruppo di gestori code.

amqsghac *queueName* **[***qMgrNome***]**

- **amqsghac** è un'applicazione IBM WebSphere MQ MQI client . Richiama i messaggi da una coda e visualizza gli eventi inviati al gestore eventi.
- Nessun punto di sincronizzazione viene utilizzato per richiamare i messaggi dalla coda.
- La riconnessione può essere effettuata a qualsiasi gestore code nello stesso gruppo di gestori code.

amqsmhac -s *sourceQueueNome* **-t** *targetQueueNome* **[-m** *qMgrNome***] [-w** *waitInterval***]**

- **amqsmhac** è un'applicazione IBM WebSphere MQ MQI client . Copia i messaggi da una coda a un'altra con un intervallo di attesa predefinito di 15 minuti dopo l'ultimo messaggio ricevuto prima del termine del programma.
- I messaggi vengono copiati nel punto di sincronizzazione.
- La riconnessione può essere effettuata solo allo stesso gestore code.

amqsfhac *QueueManagerNome QueueName SideQueueName InTransactionCount RepeatCount* **(0|1|2)**

- **amqsfhac** è un'applicazione IBM WebSphere MQ MQI client . Verifica che un gestore code a più istanze IBM WebSphere MQ che utilizza la memoria di rete, come un NAS o un file system cluster, mantenga l'integrità dei dati. Seguire i passi per eseguire **amqsfhac** in Verifica del funzionamento del file system condiviso .
- Utilizza l'opzione MQCNO_RECONNECT_Q_MGR durante la connessione a *QueueManagerNome*. Si riconnette automaticamente quando si verifica il failover del gestore code.
- Inserisce *InTransactionCount***RepeatCount* messaggi persistenti in *QueueName* durante il quale si verifica il failover del gestore code per un numero qualsiasi di volte. **amqsfhac** si riconnette al gestore code ogni volta e continua. Il test è per assicurarsi che nessun messaggio venga perso.
- *InTransactionInTransaction* i messaggi vengono inseriti all'interno di ciascuna transazione. La transazione viene ripetuta *RepeatCount* il numero di volte. Se si verifica un errore in una transazione,

amqsfhac esegue il rollback e reinoltra la transazione quando **amqsfhac** si riconnette al gestore code.

- Inserisce inoltre i messaggi in *SideQueueNome*. Utilizza *SideQueuecoda laterale* per verificare se è stato eseguito correttamente il commit o il rollback di tutti i messaggi da *QueueName* . Se rileva un'incongruenza, scrive un messaggio di errore.
- Variare la quantità di traccia di output da **amqsfhac** impostando l'ultimo parametro su `(0|1|2)`.

  0
  Output minimo.

  1
  Output intermedio.

  2
  La maggior parte dell'output.

Configurazione di una connessione client

È necessario configurare un canale di connessione client e server per eseguire gli esempi. La procedura di verifica client spiega come impostare un ambiente di test client. Consultare Verifica dell'installazione di un client.

In alternativa, utilizzare la configurazione fornita nel seguente esempio.

Esempio di utilizzo di amqsghac, amqsphace amqsmhac

L'esempio illustra i client ricollegabili che utilizzano un gestore code a istanza singola.

I messaggi vengono inseriti nella coda SOURCE da **amqsphac**, trasferiti a TARGET da **amqsmhac**e richiamati da TARGET da **amqsghac**; consultare Figura 19 a pagina 132.

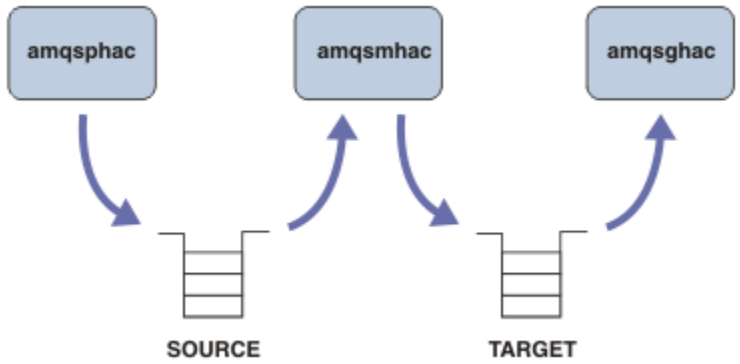


Figura 19. Esempi di client ricollegabili

Seguire questa procedura per eseguire gli esempi.

1. Creare un file `hasamples.tst` contenente i comandi:

```
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. Immettere i seguenti comandi in una richiesta comandi:

   a. `crtmqm QM1`

   b. `strmqm QM1`

c. `runmqsc QM1 < hasamples.tst`

3. Impostare il valore della variabile d'ambiente **MQCHLLIB** sul percorso del file di definizione del canale client AMQCLCHL.TAB , ad esempio `SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc`.
4. Aprire tre nuove finestre con **MQCHLLIB** impostato; ad esempio in Windows, immettere **start** tre volte al prompt dei comandi precedente avviando ogni programma in una delle finestre. Consultare il passo “5” a pagina 134 in “Imposta e controlla il gestore code” a pagina 133.)
5. Immettere il comando `endmqm -r -p QM1` per arrestare il gestore code e consentire ai client di riconnettersi.
6. Immettere il comando `strmqm QM1` per riavviare il gestore code.

I seguenti esempi mostrano i risultati dell'esecuzione degli esempi **amqsghac**, **amqsphac**e **amqsmhac** su Windows .

Imposta e controlla il gestore code

1. Creare il gestore code.

```
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.
```

   Ricordare la directory dei dati per impostare la variabile **MQCHLLIB** in un secondo momento.

2. Avviare il gestore code.

```
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. Creare le code e i canali, modificare la porta del listener e avviare il listener e il canale.

```
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. Rendere nota la tabella del canale client ai client.

   Utilizzare la directory di dati restituita dal comando **crtmqm** nel passo “1” a pagina 133e aggiungere la directory `@ipcc` per impostare la variabile **MQCHLLIB** .

```
C:\>SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc
```

5. Avviare i programmi di esempio nelle altre finestre

```
C:\>start amqsphac SOURCE QM1
C:\>start amqsmhac -s SOURCE -t TARGET -m QM1
C:\>start amqsghac TARGET QM1
```

6. Terminare il gestore code e riavviarlo.

```
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>
```

Attività correlate

Verifica del funzionamento del file system condiviso

Riferimenti correlati

amqmfsck (controllo file system)

I programmi di esempio Inquire

I programmi di esempio di interrogazione analizzano alcuni attributi di una coda che utilizza la chiamata MQINQ.

Consultare “Funzioni dimostrate nei programmi di esempio” a pagina 98 per i nomi di questi programmi.

Questi programmi sono progettati per essere eseguiti come programmi attivati, quindi il loro unico input è una struttura MQTMC2 (messaggio trigger) per sistemi IBM i, Windows, UNIX and Linux . Questa struttura contiene il nome di una coda di destinazione con attributi su cui eseguire l'interrogazione. La versione C utilizza anche il nome del gestore code. La versione COBOL utilizza il gestore code predefinito.

Affinché il processo di attivazione funzioni, assicurarsi che il programma di esempio Inquire che si desidera utilizzare sia attivato dai messaggi che arrivano sulla coda SYSTEM.SAMPLE.INQ. A tale scopo, specificare il nome del programma di esempio di interrogazione che si desidera utilizzare nel campo *ApplicId* della definizione del processo SYSTEM.SAMPLE.INQPROCESS. La coda di esempio ha un tipo di trigger FIRST; se ci sono già dei messaggi nella coda prima di eseguire l'esempio di richiesta, l'esempio inquire non viene attivato dai messaggi inviati.

Una volta impostata correttamente la definizione:

- Per sistemi UNIX, Linux e Windows , avviare il programma **runmqtrm** in una sessione, quindi avviare il programma `amqsreq` in un altro.

Utilizzare i programmi di esempio Richiesta per inviare messaggi di richiesta, ciascuno contenente solo un nome coda, alla coda SYSTEM.SAMPLE.INQ. Per ogni messaggio di richiesta, i programmi di esempio Inquire inviano un messaggio di risposta contenente informazioni sulla coda specificata nel messaggio di richiesta. Le risposte vengono inviate alla coda di risposta specificata nel messaggio di richiesta.

Progettazione del programma di esempio Inquire

Il programma apre la coda denominata nella struttura del messaggio trigger che è stata passata quando è stata avviata. (Per chiarezza, chiameremo questa *coda di richiesta*.) Il programma utilizza la chiamata MQOPEN per aprire questa coda per l'input condiviso.

Il programma utilizza la chiamata MQGET per rimuovere i messaggi da questa coda. Questa chiamata utilizza le opzioni MQGMO_ACCEPT_TRUNCATED_MSG e MQGMO_WAIT, con un intervallo di attesa di 5 secondi. Il programma verifica il descrittore di ciascun messaggio per verificare se si tratta di un messaggio di richiesta; in caso contrario, il programma elimina il messaggio e visualizza un messaggio di avvertenza.

Per ciascun messaggio di richiesta rimosso dalla coda di richieste, il programma legge il nome della coda (che chiameremo *coda di destinazione*) contenuta nei dati e apre tale coda utilizzando la chiamata MQOPEN con l'opzione MQOO_INQ. Il programma utilizza quindi la chiamata MQINQ per richiedere informazioni sui valori degli attributi *InhibitGet*, *CurrentQDepth*e *OpenInputCount* della coda di destinazione.

Se la chiamata MQINQ ha esito negativo, il programma utilizza la chiamata MQPUT1 per inserire un messaggio di risposta nella coda di risposta. Questo messaggio contiene i valori dei tre attributi.

Se la chiamata MQOPEN o MQINQ ha esito negativo, il programma utilizza la chiamata MQPUT1 per inserire un messaggio di report nella coda di risposta. Nel campo *Feedback* del descrittore del messaggio di questo messaggio di report è presente il codice motivo restituito dalla chiamata MQOPEN o MQINQ, a seconda di quale non è riuscito.

Dopo la chiamata MQINQ, il programma chiude la coda di destinazione utilizzando la chiamata MQCLOSE.

Quando non ci sono messaggi rimanenti sulla coda di richiesta, il programma chiude tale coda e si disconnette dal gestore code.

Il programma di esempio Inquire Properties of a Message Handle

AMQSIQMA è un programma C di esempio per analizzare le proprietà di un handle del messaggio da una coda messaggi ed è un esempio dell'utilizzo della chiamata API MQINQMP.

Questo esempio crea un handle del messaggio e lo inserisce nel campo MsgHandle della struttura MQGMO. L'esempio ottiene quindi un messaggio e interroga e stampa tutte le proprietà con cui è stato popolato l'handle del messaggio.

```
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
```

Programmi di esempio di pubblicazione / sottoscrizione

I programmi di esempio di pubblicazione / sottoscrizione dimostrano l'utilizzo delle funzioni di pubblicazione e sottoscrizione in WebSphere MQ.

Esistono tre programmi di esempio in linguaggio C che illustrano come programmare l'interfaccia di pubblicazione / sottoscrizione di WebSphere MQ . Ci sono alcuni esempi C che utilizzano interfacce più vecchie e ci sono esempi Java. Gli esempi Java utilizzano l'interfaccia di pubblicazione / sottoscrizione WebSphere MQ in com.ibm.mq.jar e l'interfaccia di pubblicazione / sottoscrizione JMS in com.ibm.mqjms. Gli esempi JMS non sono trattati in questo argomento.

C

Trovare l'esempio del publisher `amqspub` nella cartella degli esempi C . Eseguirlo con qualsiasi nome di argomento che si desidera come primo parametro, seguito da un nome di gestore code facoltativo. Ad esempio, `amqspub mytopic QM3` . Esiste anche una versione client denominata `amqspubc`. Se si sceglie di eseguire la versione del client, consultare prima “Preparazione ed esecuzione dei programmi di esempio” a pagina 110 per i dettagli.

Il publisher si connette al gestore code predefinito e risponde con l'output, `target topic is mytopic` . Ogni riga immessa in questa finestra da questo momento in poi viene pubblicata in `mytopic`.

Aprire un'altra finestra di comandi nella stessa directory ed eseguire il programma sottoscrittore (subscriber), `amqssub`, fornendogli lo stesso nome argomento e un nome gestore code facoltativo. Ad esempio, `amqssub mytopic QM3`.

Il sottoscrittore risponde con l'output, `Calling MQGET : 30 seconds wait time`. Da ora in poi, le righe immessa nel publisher vengono visualizzate nell'output del sottoscrittore.

Avviare un altro sottoscrittore (subscriber) in un'altra finestra di comandi e controllare che entrambi i sottoscrittori ricevano le pubblicazioni.

Per la documentazione completa dei parametri, incluse le opzioni di impostazione, fare riferimento al codice sorgente di esempio. I valori per il campo delle opzioni del sottoscrittore sono descritti nel seguente argomento: Opzioni (MQLONG).

Esiste un altro esempio di sottoscrittore (subscriber) `amqssbx`, che offre ulteriori opzioni di sottoscrizione come switch della riga comandi.

Immettere `amqssbx -d mysub -t mytopic -k` per richiamare il sottoscrittore utilizzando sottoscrizioni durevoli che vengono conservate dopo la chiusura del sottoscrittore.

Verificare la sottoscrizione pubblicando un altro elemento utilizzando il publisher. Attendere per 30 secondi la chiusura del sottoscrittore. Pubblicare altri elementi nello stesso argomento. Riavviare il sottoscrittore. L'ultimo elemento pubblicato mentre il sottoscrittore non era in esecuzione viene visualizzato dal sottoscrittore immediatamente riavviato.

Legacy C

Esiste una ulteriore serie di esempi C che mostrano i comandi accodati. Alcuni di questi esempi sono stati originariamente forniti come parte del supporto MQ0C . Le funzioni dimostrate negli esempi sono completamente supportate, per motivi di compatibilità.

Si sconsiglia di utilizzare l'interfaccia comandi in coda. È molto più complessa dell'API di pubblicazione / sottoscrizione e non vi è alcun motivo funzionale convincente per programmare comandi accodati complessi. Tuttavia, si potrebbe trovare l'approccio accodato più adatto, forse perché si sta già utilizzando

l'interfaccia o perché l'ambiente di programmazione semplifica la creazione di un messaggio complesso e la chiamata di un MQPUT generico, piuttosto che la creazione di chiamate differenti a MQSUB.

Gli esempi aggiuntivi si trovano nella sottodirectory pubsub nella cartella `samples` .

In Tabella 20 a pagina 137sono elencati sei tipi di esempi.

Tabella 20. Categorie di programmi di esempio C di pubblicazione / sottoscrizione legacy

Categoria	Programmi	Commenti
RFH1	`amqssr1a.c` `amqspr1a.c`	Esempio di pubblicazione / sottoscrizione semplice creato utilizzando i messaggi in formato RFH1 .
RFH2	`amqssr2a.c` `amqspr2a.c`	Esempio di pubblicazione / sottoscrizione semplice creato utilizzando i messaggi in formato RFH2 .
Esempi MQAI	`amqsppca.c` `amqsspca.c`	Esempio di pubblicazione / sottoscrizione semplice creato utilizzando i comandi PCF e l'interfaccia comandi MQAI.
MA0C Servizio risultati utilizzando RFH1	`amqsgama.c` `amqsresa.c`	Servizio dei risultati creato utilizzando intestazioni RFH1 1. Richiede le code definite in `amqsgama.tst` e `amqsresa.tst` 2. `amqsresa` deve essere avviato prima `amqsgama`
MA0C Servizio risultati utilizzando RFH2	`amqsgr2a.c` `amqsrr2a.c`	Servizio dei risultati creato utilizzando intestazioni RFH2 1. Richiede le code definite in `amqsgama.tst` e `amqsresa.tst` 2. `amqsresa` deve essere avviato prima `amqsgama`
Esempio di pubblicazione / sottoscrizione dell'uscita di instradamento	`amqspsra.c`	Dimostra come modificare la destinazione della coda o del gestore code per un messaggio di pubblicazione / sottoscrizione in un'uscita di instradamento.

Java

L'esempio Java `MQPubSubApiSample.java` combina publisher e sottoscrittori in un singolo programma. I relativi file di origine e di classe compilati si trovano nella cartella degli esempi `wmqjava` .

Se si sceglie di eseguire in modalità client, consultare prima “Preparazione ed esecuzione dei programmi di esempio” a pagina 110 per i dettagli.

Eseguire l'esempio dalla riga comandi utilizzando il comando Java, se si dispone di un ambiente Java configurato. È anche possibile eseguire l'esempio dallo spazio di lavoro WebSphere MQ Explorer Eclipse che ha un workbench di programmazione Java già configurato.

Potrebbe essere necessario modificare alcune delle proprietà del programma di esempio per eseguirlo. A tale scopo, fornire i parametri alla JVM o modificare l'origine.

Le istruzioni in “Esecuzione dell'esempio Java MQPubSubApiSample” a pagina 138 mostrano come eseguire l'esempio dallo spazio di lavoro Eclipse .

Esecuzione dell'esempio Java MQPubSubApiSample
Come eseguire MQPubSubApiSample utilizzando Java Development Tools dalla piattaforma Eclipse .

Prima di iniziare
Aprire il workbench Eclipse . Creare una nuova directory dello spazio di lavoro e selezionarla. Chiudere la finestra di benvenuto.

Seguire la procedura descritta in “Preparazione ed esecuzione dei programmi di esempio” a pagina 110 prima di eseguire come client.

Informazioni su questa attività
Il programma di esempio di pubblicazione / sottoscrizione Java è un programma Java client WebSphere MQ . L'esempio viene eseguito senza modifiche utilizzando un gestore code predefinito in ascolto sulla porta 1414. L'attività descrive questo caso semplice e indica in termini generali come fornire parametri e modificare l'esempio per adattarlo alle diverse configurazioni di WebSphere MQ . L'esempio è illustrato in esecuzione su Windows. I percorsi dei file differiranno su altre piattaforme.

Procedura
1. Importa i programmi di esempio Java
   a) Nel workbench, fare clic su **Finestra** > **Apri prospettiva** > **Altro** > **Java** e selezionare **OK**.
   b) Passare alla vista **Esplora package** .
   c) Fare clic con il tasto destro del mouse nello spazio vuoto nella vista **Esplora package** . Fare clic su **Nuovo progetto** > **Java**.
   d) Nel campo **Project name** immettere `MQ Java Samples`. Fare clic su **Avanti**.
   e) Nel pannello **Java Settings** , passare alla scheda **Librerie** .
   f) Fare clic su **Aggiungi JAR esterni**.
   g) Passare a *MQ_INSTALLATION_PATH*`\java\lib` dove *MQ_INSTALLATION_PATH* è la cartella di installazione di WebSphere MQ e selezionare `com.ibm.mq.jar` e `com.ibm.mq.jmqi.jar`
   h) Fare clic su **Apri** > **Fine.**
   i) Fare clic con il pulsante destro del mouse su `src` nella vista **Esplora package** .
   j) Selezionare **Importa ...** > **Generale** > **File System** > **Avanti** > **Sfoglia**... e passare al percorso *MQ_INSTALLATION_PATH*`\tools\wmqjava\samples` dove *MQ_INSTALLATION_PATH* è la directory di installazione di WebSphere MQ .
   k) Nel pannello **Importa** , Figura 20 a pagina 139, fare clic su `samples` (non selezionare la check box).
   l) Selezionare `MQPubSubApiSample.java`. Il campo **Into folder** deve contenere `MQ Java Samples/src`. Fare clic su **Fine**.

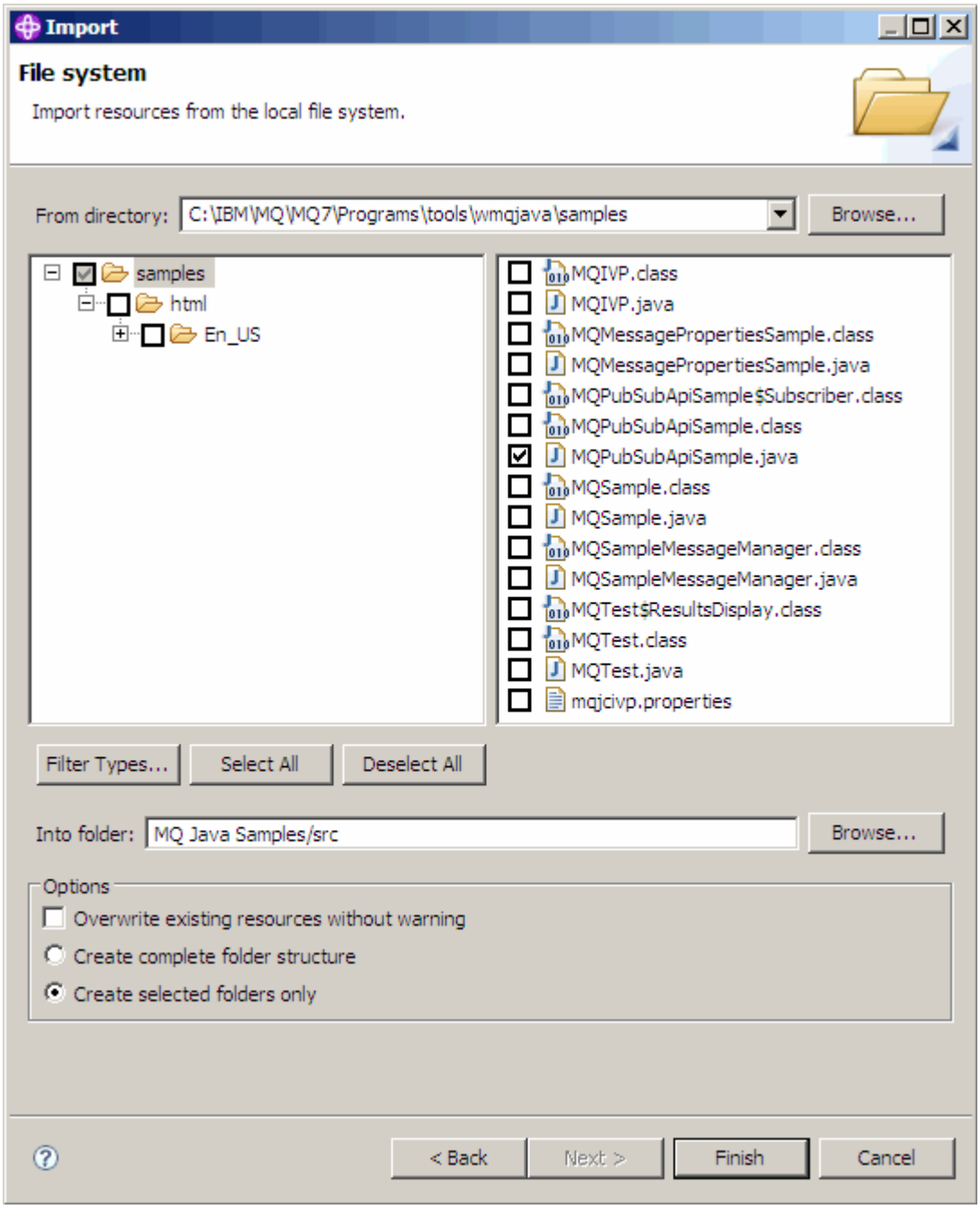


Figura 20. Importazione file system

2. Eseguire il programma di esempio di pubblicazione / sottoscrizione.

   Esistono due modi per eseguire il programma, a seconda che sia necessario modificare i parametri predefiniti.

   - La prima scelta esegue il programma senza apportare alcuna modifica:
     - Nel menu principale dello spazio di lavoro, espandere la cartella `src` . Fare clic con il tasto destro del mouse su **MQPubSubApiSample.javaRun - as** > **1. Applicazione Java**
   - La seconda scelta esegue il programma con parametri o con codice sorgente modificato per il proprio ambiente:
     - Aprire `MQPubSubApiSample.java` e studiare il costruttore `MQPubSubApiSample` .

– Modificare gli attributi del programma.

  Questi attributi sono modificabili utilizzando lo switch JVM -D o fornendo un valore predefinito per la proprietà di sistema modificando il codice di origine.

  - topicObject
  - queueManagerName
  - subscriberCount

  Questi attributi sono modificabili solo modificando il codice sorgente nel costruttore.

  - nome host
  - porta
  - canale

  Per impostare le proprietà di sistema, codificare un valore predefinito nell'accessor, ad esempio:

```
queueManagerName = System.getProperty("com.ibm.mq.pubSubSample.queueManagerName",
"QM3");
```

  Oppure fornire il parametro alla JVM utilizzando l'opzione `-D` , come mostrato nei seguenti passi:

a. Copiare il nome completo di System.Property che si desidera impostare, ad esempio: `com.ibm.mq.pubSubSample.queueManagerName`.
b. Nello spazio di lavoro, fare clic con il tasto destro del mouse su **Esegui** > **Apri finestra Esegui**. Fare doppio clic su Java Application in **Crea, gestisci ed esegui applicazioni** e fare clic sulla scheda **(x) = Argomenti** .
c. Nel riquadro **Argomenti VM:** , immettere `-D` e incollare il nome System.property , `com.ibm.mq.pubSubSample.queueManagerName`, seguito da =QM3. Fare clic su **Applica** > **Esegui**.
d. Aggiungere ulteriori argomenti come un elenco separato da virgole o come righe aggiuntive nel riquadro, senza separatori di virgole.

   Ad esempio:`-Dcom.ibm.mq.pubSubSample.queueManagerName=QM3, -Dcom.ibm.mq.pubSubSample.subscriberCount=6`.

Il programma di esempio Pubblica uscita

AMQSPSE0 è un programma C di esempio di un'uscita per intercettare una pubblicazione prima di consegnarla a un sottoscrittore. L'uscita può quindi, ad esempio, modificare le intestazioni del messaggio, il payload o la destinazione oppure impedire che il messaggio venga pubblicato in un sottoscrittore.

Per eseguire l'esempio, eseguire le seguenti attività:

1. Configurare il gestore code:
   - Sui sistemi UNIX and Linux aggiungere una sezione come questa al file `qm.ini` :

```
PublishSubscribe:
           PublishExitPath=<Module>
           PublishExitFunction=EntryPoint
```

     dove il modulo è *MQ_INSTALLATION_PATH*`/samp/bin/amqspse` . *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ . In Windows impostare gli attributi equivalenti nel registro.
2. Assicurarsi che il modulo sia accessibile a WebSphere MQ.
3. Riavviare il gestore code per selezionare la configurazione.
4. Nel processo dell'applicazione da tracciare, descrivere dove devono essere scritti i file di traccia. Ad esempio:

- Sui sistemi UNIX and Linux , assicurarsi che la directory`/var/mqm/trace` esista e esportare la seguente variabile di ambiente:

```
export MQPSE_TRACE_LOGFILE=/var/mqm/trace/PubTrace
```

- Su Windows, verificare che la directory `C:\temp` esista e impostare la seguente variabile di ambiente:

```
set MQPSE_TRACE_LOGFILE=C:\temp\PubTrace
```

Programmi di esempio Put

I programmi di esempio Put inserano i messaggi su una coda utilizzando la chiamata MQPUT.

Consultare “Funzioni dimostrate nei programmi di esempio” a pagina 98 per i nomi di questi programmi.

Progettazione del programma di esempio Put

Il programma utilizza la chiamata MQOPEN con l'opzione MQOO_OUTPUT per aprire la coda di destinazione per inserire i messaggi.

Se non riesce ad aprire la coda, il programma emette un messaggio di errore contenente il codice motivo restituito dalla chiamata MQOPEN. Per semplificare il programma, in questa e nelle successive chiamate MQI, il programma utilizza i valori predefiniti per molte opzioni.

Per ogni riga di input, il programma legge il testo in un buffer e utilizza la chiamata MQPUT per creare un messaggio datagramma contenente il testo di tale riga. Il programma continua fino a quando non raggiunge la fine dell'input o la chiamata MQPUT ha esito negativo. Se il programma raggiunge la fine dell'input, chiude la coda utilizzando la chiamata MQCLOSE.

Esecuzione dei programmi di esempio Put

Esecuzione degli esempi amqsput e amqsputc

Ognuno di questi programmi ha 2 parametri:

1. Il nome della coda di destinazione (obbligatorio)
2. Il nome del gestore code (facoltativo)

Se non viene specificato un gestore code, amqsput si connette al gestore code predefinito e amqsputc si connette al gestore code identificato da una variabile di ambiente o dal file di definizione del canale del client. Per eseguire questi programmi, immettere uno dei seguenti:

- `amqsput myqueue qmanagername`
- `amqsputc myqueue qmanagername`

dove `myqueue` è il nome della coda in cui verranno inseriti i messaggi e `qmanagername` è il gestore code proprietario di `myqueue`.

Esecuzione dell'esempio amq0put

La versione COBOL non ha alcun parametro. Si connette al gestore code predefinito e quando viene eseguito viene richiesto:

```
Please enter the name of the target queue
```

Prende l'input da StdIn e aggiunge ogni riga di input alla coda di destinazione. Una riga vuota indica che non ci sono più dati.

Programmi di esempio del messaggio di riferimento

Gli esempi di messaggi di riferimento consentono il trasferimento di un oggetto di grandi dimensioni da un nodo ad un altro (di solito su sistemi diversi) senza che sia necessario memorizzare l'oggetto nelle code WebSphere MQ sui nodi di origine o di destinazione.

Viene fornita una serie di programmi di esempio per dimostrare come i messaggi di riferimento possono essere inseriti in una coda, ricevuti dalle uscite dei messaggi e presi da una coda. I programmi di esempio utilizzano i messaggi di riferimento per spostare i file. Se si desidera spostare altri oggetti come i database o se si desidera eseguire controlli di sicurezza, definire la propria uscita, in base al nostro esempio, amqsxrm. Le seguenti sezioni descrivono i programmi di esempio del messaggio di riferimento.

La versione del programma di esempio di uscita del messaggio di riferimento da utilizzare dipende dalla piattaforma su cui è in esecuzione il canale. Su tutte le piattaforme, utilizzare amqsxrma all'estremità di invio. Utilizzare amqsxrma all'estremità ricevente se il destinatario è in esecuzione in qualsiasi prodotto WebSphere MQ tranne WebSphere MQ per IBM i.

Esecuzione degli esempi del messaggio di riferimento

Utilizzare queste informazioni per informazioni su come eseguire i programmi di esempio Messaggio di riferimento.

Gli esempi del messaggio di riferimento vengono eseguiti come segue:

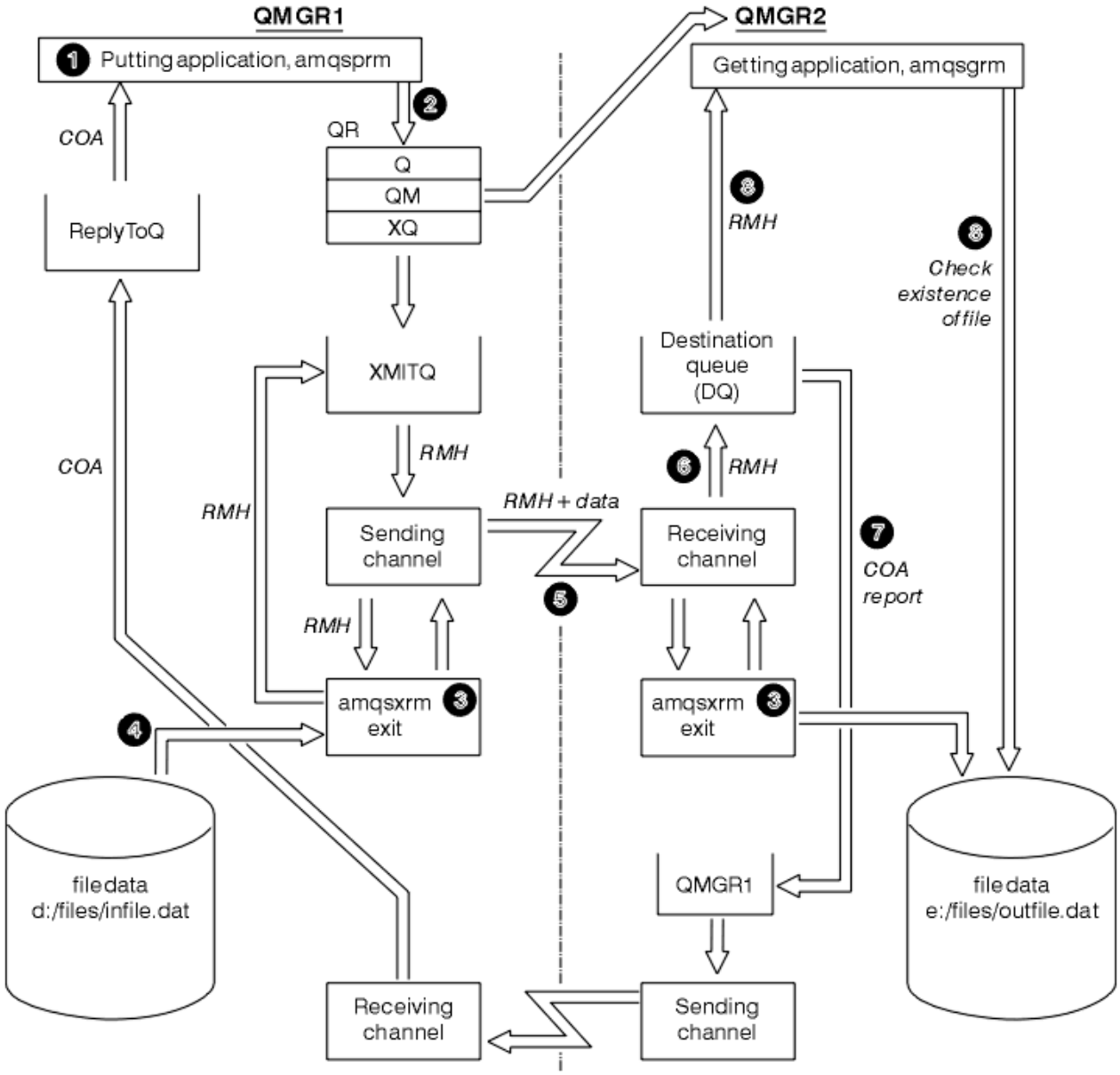


Figura 21. Esecuzione degli esempi del messaggio di riferimento

1. Impostare l'ambiente per avviare i listener, i canali e i controlli dei trigger e definire i canali e le code.

   Per descrivere come impostare l'esempio di messaggio di riferimento, si fa riferimento alla macchina di invio come MACHINE1 con un gestore code denominato QMGR1 e la macchina di ricezione come MACHINE2 con un gestore code denominato QMGR2.

   Nota: Le seguenti definizioni consentono di creare un messaggio di riferimento per inviare un file con un tipo di oggetto FLATFILE dal gestore code QMGR1 a QMGR2 e per creare nuovamente il file come definito nella chiamata a AMQSPRM (o AMQSPRMA su IBM i). Il messaggio di riferimento (inclusi i dati del file) viene inviato utilizzando il canale CHL1 e la coda di trasmissione XMITQ e inserito nella coda DQ. I report di eccezione e COA vengono inviati nuovamente a QMGR1 utilizzando il canale REPORT e la coda di trasmissione QMGR1.

   L'applicazione che riceve il messaggio di riferimento (AMQSGRM) viene attivata utilizzando la coda di avvio INITQ e elaborando PROC. Verificare che i campi CONNAME siano impostati correttamente e che il campo MSGEXIT rifletta la struttura di directory, in base al tipo di macchina e alla posizione in cui è installato il prodotto WebSphere MQ .

Le definizioni MQSC hanno utilizzato uno stile AIX per definire le uscite. È importante notare che i dati del messaggio FLATFILE sono sensibili al maiuscolo / minuscolo e l'esempio non funzionerà a meno che non siano in maiuscolo.

Sulla macchina MACHINE1, gestore code QMGR1

Sintassi 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
```

Nota: Se non si specifica un nome di gestore code, il sistema utilizza il gestore code predefinito.

```
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)
```

Sulla macchina MACHINE2, gestore code QMGR2

Sintassi 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. Una volta creati gli oggetti WebSphere MQ :
   a. Dove applicabile alla piattaforma, avviare il listener per i gestori code di invio e ricezione
   b. Avviare i canali CHL1 e REPORT
   c. Sul gestore code di ricezione avviare il controllo trigger per la coda di iniziazione INITQ
3. Richiamare il programma di esempio AMQSPRM dalla riga comandi utilizzando i seguenti parametri:

   -m Nome del gestore code locale; il valore predefinito è il gestore code predefinito

   -i Nome e ubicazione del file di origine

   -o Nome e ubicazione del file di destinazione

   -q Nome della coda

-g Nome del gestore code in cui si trova la coda, definita nel parametro -q. Il valore predefinito è il gestore code specificato nel parametro -m

-t Tipo oggetto

-w Intervallo di attesa, ossia il tempo di attesa per i report di eccezione e COA dal gestore code di ricezione

Ad esempio, per utilizzare l'esempio con gli oggetti definiti precedentemente, utilizzare i seguenti parametri:

```
-mQMGR1 -iInput File -oOutput File -qQR -tFLATFILE -w120
```

L'aumento del tempo di attesa consente l'invio di un file di grandi dimensioni attraverso una rete prima che il programma metta in timeout i messaggi.

```
amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
```

Nota: Per le piattaforme UNIX and Linux , è necessario utilizzare due barre retroverse (\\) invece di una per indicare la directory del file di destinazione. Pertanto, il comando **amqsprm** è simile al seguente:

```
amqsprm -i /files/infile.dat -o e:\\files\\outfile.dat -q QR
-m QMGR1 -w 30 -t FLATFILE
```

L'esecuzione del programma put Reference Message effettua le seguenti operazioni:

- Il messaggio di riferimento viene inserito nella coda QR sul gestore code QMGR1.
- Il file di origine e il percorso sono `d:\files\infile.dat` ed esistono sul sistema in cui viene immesso il comando di esempio.
- Se la coda QR è una coda remota, il messaggio di riferimento viene inviato a un altro gestore code, su un sistema differente, dove viene creato un file con il nome e il percorso `e:\files\outfile.dat`. Il contenuto di questo file è lo stesso del file di origine.
- amqsprm attende per 30 secondi un report COA dal gestore code di destinazione.
- Il tipo di oggetto è `flatfile`, quindi il canale utilizzato per spostare i messaggi dalla coda QR deve specificarlo nel campo *MsgData* .

4. Quando si definiscono i canali, selezionare l'uscita del messaggio sia all'estremità di invio che a quella di ricezione per essere amqsxrm. Ciò è definito in WebSphere MQ per Windows come segue:

```
msgexit('pathname\amqsxrm.dll(MsgExit)')
```

Ciò è definito in WebSphere MQ per AIX, WebSphere MQ per HP-UXe WebSphere MQ per Solaris come segue:

```
msgexit('pathname/amqsxrm(MsgExit)')
```

Se si specifica un nome percorso, specificare il nome completo. Se si omette il nome percorso, si presuppone che il programma si trovi nel percorso specificato nel file `qm.ini` (o, in WebSphere MQ per Windows, il percorso specificato nel registro).

5. L'uscita del canale legge l'intestazione del messaggio di riferimento e trova il file a cui fa riferimento.
6. L'uscita del canale può quindi segmentare il file prima di inviarlo lungo il canale insieme all'intestazione. In WebSphere MQ per AIX, WebSphere MQ per HP-UXe WebSphere MQ per Solaris, modificare il proprietario del gruppo della directory di destinazione in 'mqm' in modo che l'uscita del messaggio di esempio possa creare il file in quella directory. Inoltre, modificare le autorizzazioni della directory di destinazione per consentire ai membri del gruppo mqm di scrivervi. I dati del file non vengono memorizzati sulle code WebSphere MQ .

7. Quando l'ultimo segmento del file viene elaborato dall'uscita del messaggio ricevente, il messaggio di riferimento viene inserito nella coda di destinazione specificata da amqsprm. Se questa coda viene attivata (ovvero, la definizione specifica gli attributi della coda *Trigger*, *InitQ*e *Process*), il programma specificato dal parametro PROC della coda di destinazione viene attivato. Il programma da attivare deve essere definito nel campo *ApplId* dell'attributo *Process* .
8. Quando il messaggio di riferimento raggiunge la coda di destinazione (DQ), viene inviato un report COA all'applicazione di inserimento (amqsprm).
9. L'esempio Richiama messaggio di riferimento, amqsgrm, richiama i messaggi dalla coda specificata nel messaggio del trigger di input e verifica l'esistenza del file.

Progettazione dell'esempio Inserisci messaggio di riferimento (amqsprma.c, AMQSPRM4)

Questo argomento fornisce una descrizione dettagliata di un esempio di messaggio Put Reference.

Questo esempio crea un messaggio di riferimento che fa riferimento a un file e lo inserisce in una coda specificata:

1. L'esempio si connette a un gestore code locale utilizzando MQCONN.
2. Apre quindi (MQOPEN) una coda modello utilizzata per ricevere i messaggi di report.
3. L'esempio crea un messaggio di riferimento contenente i valori richiesti per spostare il file, ad esempio, i nomi dei file di origine e di destinazione e il tipo di oggetto. Come esempio, l'esempio fornito con WebSphere MQ crea un messaggio di riferimento per inviare il file `d:\x\file.in` da `QMGR1` a `QMGR2` e ricreare il file come `d:\y\file.out` utilizzando i seguenti parametri:

   ```
   amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
   ```

   Dove `QR` è una definizione di coda remota che fa riferimento a una coda di destinazione su `QMGR2`.

   Nota: Per le piattaforme UNIX and Linux , utilizzare due barre rovesciate (\\) invece di una per indicare la directory del file di destinazione. Pertanto, il comando **amqsprm** è simile al seguente:

   ```
   amqsprm -q QR -m QMGR1 -i /x/file.in -o d:\\y\\file.out -t FLATFILE
   ```
4. Il messaggio di riferimento viene inserito (senza dati di file) nella coda specificata dal parametro /q. Se si tratta di una coda remota, il messaggio viene inserito nella coda di trasmissione corrispondente.
5. L'esempio attende, per il periodo di tempo specificato nel parametro /w (che per impostazione predefinita è di 15 secondi), i report COA, che, insieme ai report di eccezioni, vengono inviati alla coda dinamica creata sul gestore code locale (QMGR1).

Progettazione dell'esempio Uscita messaggio di riferimento (amqsxrma.c, AMQSXRM4)

Questo esempio riconosce i messaggi di riferimento con un tipo di oggetto che corrisponde al tipo di oggetto nel campo dati utente dell'uscita messaggi della definizione del canale.

Per questi messaggi, si verifica quanto segue:

- Sul canale mittente o server, la lunghezza specificata dei dati viene copiata dall'offset specificato del file specificato nello spazio rimanente nel buffer dell'agente dopo il messaggio di riferimento. Se la fine del file non viene raggiunta, il messaggio di riferimento viene reinserito nella coda di trasmissione dopo l'aggiornamento del campo *DataLogicalOffset* .
- Sul canale richiedente o destinatario, se il campo *DataLogicalOffset* è zero e il file specificato non esiste, viene creato. I dati che seguono il messaggio di riferimento vengono aggiunti alla fine del file specificato. Se il messaggio di riferimento non è l'ultimo per il file specificato, viene eliminato. Altrimenti, viene restituito all'uscita del canale, senza i dati accodati, da inserire nella coda di destinazione.

Per i canali mittente e server, se il campo *DataLogicalLength* nel messaggio di riferimento di input è zero, la parte rimanente del file, da *DataLogicalOffset* alla fine del file, deve essere inviata lungo il canale. Se non è zero, viene inviata solo la lunghezza specificata.

Se si verifica un errore (ad esempio, se l'esempio non può aprire un file), MQCXP.*ExitResponse* è impostata su MQXCC_SUPPRESS_FUNCTION in modo che il messaggio in fase di elaborazione venga inserito nella coda di messaggi non recapitabili invece di continuare con la coda di destinazione. Un codice di feedback viene restituito in MQCXP.*Feedback* e restituito all'applicazione che inserisce il messaggio nel campo *Feedback* del descrittore del messaggio di un messaggio di report. Ciò è dovuto al fatto che l'eccezione richiesta dall'applicazione di inserimento viene segnalata impostando MQRO_EXCEPTION nel campo *Report* di MQMD.

Se la codifica o il CCSID (*CodedCharacterSetId*) del messaggio di riferimento è diverso da quello del gestore code, il messaggio di riferimento viene convertito nella codifica locale e nel CCSID. Nel nostro esempio, amqsprm, il formato dell'oggetto è MQFMT_STRING, per cui amqsxrm converte i dati dell'oggetto nel CCSID locale all'estremità di ricezione prima che i dati vengano scritti nel file.

Non specificare il formato del file trasferito come MQFMT_STRING se il file contiene caratteri multibyte (ad esempio, DBCS o Unicode). Ciò è dovuto al fatto che un carattere multibyte potrebbe essere suddiviso quando il file è segmentato all'estremità di invio. Per trasferire e convertire tale file, specificare il formato diverso da MQFMT_STRING in modo che l'uscita del messaggio di riferimento non lo converta e converte il file all'estremità di ricezione quando il trasferimento è completo.

Compilazione dell'esempio di uscita del messaggio di riferimento

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Per compilare amqsxrma, utilizzare i comandi riportati di seguito:

Su 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
```

Su 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
```

SULinux

```
$ 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
```

Su 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
```

Su Windows

WebSphere MQ ora fornisce la libreria mqm con i pacchetti client e i pacchetti server, quindi il seguente esempio utilizza `mqm.lib` invece di `mqmvx.lib`:

```
cl amqsqrma.c /link /out:amqsxrm.dll /dll mqm.lib mqm.lib /def:amqsxrm.def
```

Per informazioni generali sulla scrittura e la compilazione delle uscite del canale, consultare “Scrittura di programmi di uscita canale” a pagina 401

Progettazione dell'esempio Richiama messaggio di riferimento (amqsgrma.c, AMQSGRM4)

Questo argomento illustra la progettazione dell'esempio Richiama messaggio di riferimento.

La logica del programma è la seguente:

1. L'esempio viene attivato ed estrae il nome della coda e del gestore code dal messaggio del trigger di input.
2. Si connette quindi al gestore code specificato utilizzando MQCONN e apre la coda specificata utilizzando MQOPEN.
3. L'esempio emette MQGET con un intervallo di attesa di 15 secondi in un loop per richiamare i messaggi dalla coda.
4. Se un messaggio è un messaggio di riferimento, l'esempio verifica l'esistenza del file che è stato trasferito.
5. Chiude la coda e si disconnette dal gestore code.

I programmi di esempio Richiesta

I programmi di esempio Richiesta dimostrano l'elaborazione client/server. Gli esempi sono i client che inseriscono i messaggi di richiesta in una coda del server di destinazione elaborata da un programma server. Essi attendono che il programma server inserisci un messaggio di risposta in una coda di risposta.

Gli esempi di richiesta inserano una serie di messaggi di richiesta sulla coda server di destinazione utilizzando la chiamata MQPUT. Questi messaggi specificano la coda locale, SYSTEM.SAMPLE.REPLY come coda di risposta, che può essere una coda locale o remota. I programmi attendono i messaggi di risposta, quindi li visualizzano. Le risposte vengono inviate solo se la coda del server di destinazione viene elaborata da un'applicazione server o se un'applicazione viene attivata a tale scopo (i programmi di esempio Inquire, Set ed Echo sono progettati per essere attivati). L'esempio C attende 1 minuto (l'esempio COBOL attende 5 minuti), per l'arrivo della prima risposta (per consentire l'attivazione di un'applicazione server) e 15 secondi per le risposte successive, ma entrambi gli esempi possono terminare senza ottenere alcuna risposta. Consultare “Funzioni dimostrate nei programmi di esempio” a pagina 98 per i nomi dei programmi di esempio Richiesta.

Esecuzione dei programmi di esempio Richiesta

Esecuzione degli esempi amqsreq0.c, amqsreq e amqsreqc

La versione C del programma prende tre parametri:

1. Il nome della coda del server di destinazione (necessario)
2. Il nome del gestore code (facoltativo)
3. La coda di risposta (facoltativo)

Ad esempio, immettere uno dei seguenti:

- `amqsreq myqueue qmanagername replyqueue`
- `amqsreqc myqueue qmanagername`
- `amq0req0 myqueue`

dove `myqueue` è il nome della coda del server di destinazione, `qmanagername` è il nome del gestore code proprietario di `myqueue`e `replyqueue` è il nome della coda di risposta.

Se si omette il nome del gestore code, si presume che il gestore code predefinito sia proprietario della coda. Se si omette il nome della coda di risposta, viene fornita la coda di risposta predefinita.

Esecuzione dell'esempio amq0req0.cbl

La versione COBOL non ha alcun parametro. Si connette al gestore code predefinito e quando viene eseguito viene richiesto:

```
Please enter the name of the target server queue
```

Il programma prende il suo input da StdIn e aggiunge ogni linea alla coda del server di destinazione, prendendo ogni riga di testo come contenuto di un messaggio di richiesta. Il programma termina quando viene letta una riga nulla.

Esecuzione dell'esempio AMQSREQ4

Il programma C crea messaggi prendendo i dati da stdin (la tastiera) con un tempo vuoto che termina l'immissione. Il programma utilizza un massimo di tre parametri: il nome della coda di destinazione (obbligatorio), il nome del gestore code (facoltativo) e il nome della coda di risposta (facoltativo). Se non viene specificato alcun nome di gestore code, viene utilizzato il gestore code predefinito. Se non viene specificata alcuna coda di risposta, SYSTEM.SAMPLE.REPLY REPLY.

Di seguito viene riportato un esempio di come richiamare il programma di esempio C, specificando la coda di risposta, ma lasciando al gestore code il valore predefinito:

```
CALL PGM(QMQM/AMQSREQ4) PARM('SYSTEM.SAMPLE.LOCAL' '' 'SYSTEM.SAMPLE.REPLY')
```

Nota: Tenere presente che i nomi delle code sono sensibili al maiuscolo / minuscolo. Tutte le code create dal programma di creazione del file di esempio AMQSAMP4 hanno nomi creati in caratteri maiuscoli.

Esecuzione dell'esempio AMQ0REQ4

Il programma COBOL crea messaggi accettando i dati dalla tastiera. Per avviare il programma, richiamare il programma e specificare il nome della coda di destinazione come parametro. Il programma accetta l'immissione dalla tastiera in un buffer e crea un messaggio di richiesta per ogni riga di testo. Il programma si arresta quando si immette una riga vuota sulla tastiera.

Esecuzione dell'esempio Richiesta utilizzando il trigger

Se l'esempio viene utilizzato con il trigger e uno dei programmi di esempio Inquire, Set o Echo, la riga di input deve essere il nome della coda a cui si desidera che il programma attivato acceda.

Sistemi UNIX, Linux e Windows

Per eseguire gli esempi utilizzando il trigger:

1. Avviare il programma di controllo trigger RUNMQTRM in una sessione (la coda di iniziazione SYSTEM.SAMPLE.TRIGGER è disponibile per l'uso).
2. Avviare il programma amqsreq in un'altra sessione.
3. Assicurarsi di aver definito una coda del server di destinazione.

   Le code di esempio disponibili da utilizzare come coda del server di destinazione per l'esempio di richiesta per inserire i messaggi sono:

   - SYSTEM.SAMPLE.INQ - per il programma di esempio Inquire
   - SYSTEM.SAMPLE.SET - per il programma di esempio Set
   - SYSTEM.SAMPLE.ECHO - per il programma di esempio Echo

   Queste code hanno un tipo di trigger FIRST, quindi se ci sono già messaggi nelle code prima di eseguire l'esempio Richiesta, le applicazioni server non vengono attivate dai messaggi inviati.
4. Assicurarsi di aver definito una coda per il programma di esempio Inquire, Set o Echo da utilizzare.

Ciò significa che il controllo trigger è pronto quando l'esempio di richiesta invia un messaggio.

Nota: Le definizioni del processo di esempio create utilizzando RUNMQSC e il file amqscos0.tst attivano gli esempi C. Modificare le definizioni del processo in amqscos0.tst e utilizzare RUNMQSC con questo file aggiornato per utilizzare versioni COBOL.

Figura 22 a pagina 150 dimostra come utilizzare insieme gli esempi Richiesta e Interroga.

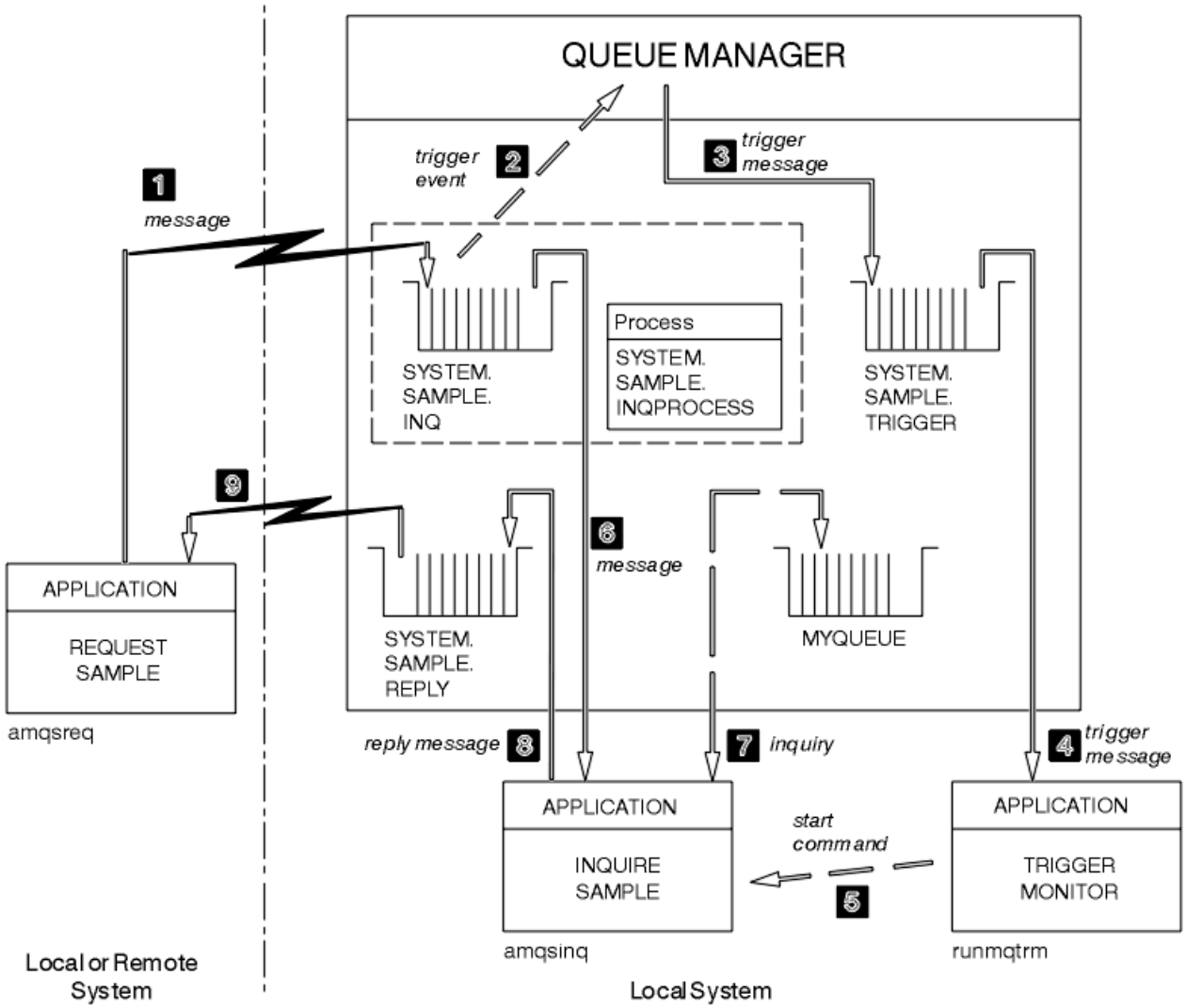


Figura 22. Richiedi e richiedi esempi utilizzando il trigger

In Figura 22 a pagina 150 l'esempio Richiesta inserisce i messaggi nella coda del server di destinazione, SYSTEM.SAMPLE.INQe l'esempio Inquire interroga la coda MYQUEUE. In alternativa, è possibile utilizzare una delle code di esempio definite quando è stato eseguito amqscos0.tsto qualsiasi altra coda definita, per l'esempio Inquire.

Nota: I numeri in Figura 22 a pagina 150 mostrano la sequenza di eventi.

Per eseguire gli esempi Richiesta e Interroga, utilizzando il trigger:

1. Verificare che le code che si desidera utilizzare siano definite. Eseguire amqscos0.tstper definire le code di esempio e definire una coda MYQUEUE.
2. Eseguire il comando di controllo trigger RUNMQTRM:

```
RUNMQTRM -m qmanagername -q SYSTEM.SAMPLE.TRIGGER
```

3. Esegui l'esempio di richiesta

```
amqsreq SYSTEM.SAMPLE.INQ
```

Nota: L'oggetto processo definisce cosa deve essere attivato. Se il client e il server non sono in esecuzione sulla stessa piattaforma, qualsiasi processo avviato dal controllo trigger deve definire *ApplType*, altrimenti il server utilizza le definizioni predefinite (ovvero, il tipo di applicazione normalmente associato alla macchina server) e causa un errore.

Per un elenco di tipi di applicazione, vedere ApplType.

4. Immettere il nome della coda che si desidera venga utilizzata dall'esempio di interrogazione:

```
MYQUEUE
```

5. Immettere una riga vuota (per terminare il programma Richiesta).
6. L'esempio di richiesta visualizzerà quindi un messaggio, contenente i dati del programma Inquire ottenuti da MYQUEUE.

È possibile utilizzare più di una coda; in questo caso, immettere i nomi delle altre code al passo “4” a pagina 151.

Per ulteriori informazioni sull'attivazione, consultare “Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger” a pagina 331.

Progettazione del programma di esempio Richiesta

Il programma apre la coda del server di destinazione in modo che possa inserire i messaggi. Utilizza la chiamata MQOPEN con l'opzione MQOO_OUTPUT. Se non è in grado di aprire la coda, il programma visualizza un messaggio di errore contenente il codice motivo restituito dalla chiamata MQOPEN.

Il programma apre quindi la coda di risposta denominata SYSTEM.SAMPLE.REPLY in modo che possa ricevere i messaggi di risposta. Per questo, il programma usa la chiamata MQOPEN con l'opzione MQOO_INPUT_EXCLUSIVE. Se non è in grado di aprire la coda, il programma visualizza un messaggio di errore contenente il codice motivo restituito dalla chiamata MQOPEN.

Per ogni riga di input, il programma legge il testo in un buffer e utilizza la chiamata MQPUT per creare un messaggio di richiesta contenente il testo di quella riga. In questa chiamata il programma utilizza l'opzione di report MQRO_EXCEPTION_WITH_DATA per richiedere che qualsiasi messaggio di report inviato sul messaggio di richiesta includa i primi 100 byte dei dati del messaggio. Il programma continua fino a quando non raggiunge la fine dell'input o la chiamata MQPUT ha esito negativo.

Il programma utilizza quindi la chiamata MQGET per rimuovere i messaggi di risposta dalla coda e visualizza i dati contenuti nelle risposte. La chiamata MQGET utilizza le opzioni MQGMO_WAIT, MQGMO_CONVERT e MQGMO_ACCEPT_TRUNCATED. Il *WaitInterval* è 5 minuti nella versione COBOL e 1 minuto nella versione C, per la prima risposta (per consentire l'attivazione di un'applicazione server) e 15 secondi per le risposte successive. Il programma attende questi periodi se non è presente alcun messaggio sulla coda. Se non arriva alcun messaggio prima della scadenza di questo intervallo, la chiamata ha esito negativo e restituisce il codice di errore MQRC_NO_MSG_AVAILABLE. La chiamata utilizza anche l'opzione MQGMO_ACCEPT_TRUNCATED_MSG, in modo che i messaggi più lunghi della dimensione del buffer dichiarata vengano troncati.

Il programma dimostra come cancellare i dati dei campi *MsgId* e *CorrelId* della struttura MQMD dopo ogni chiamata MQGET perché la chiamata imposta questi campi sui valori contenuti nel messaggio richiamato. La cancellazione di questi campi significa che le chiamate MQGET successive richiamano i messaggi nell'ordine in cui sono conservati nella coda.

Il programma continua fino a quando la chiamata MQGET non restituisce il codice motivo MQRC_NO_MSG_AVAILABLE o la chiamata MQGET non riesce. Se la chiamata ha esito negativo, il programma visualizza un messaggio di errore che contiene il codice di errore.

Il programma chiude quindi la coda del server di destinazione e la coda di risposta utilizzando la chiamata MQCLOSE.

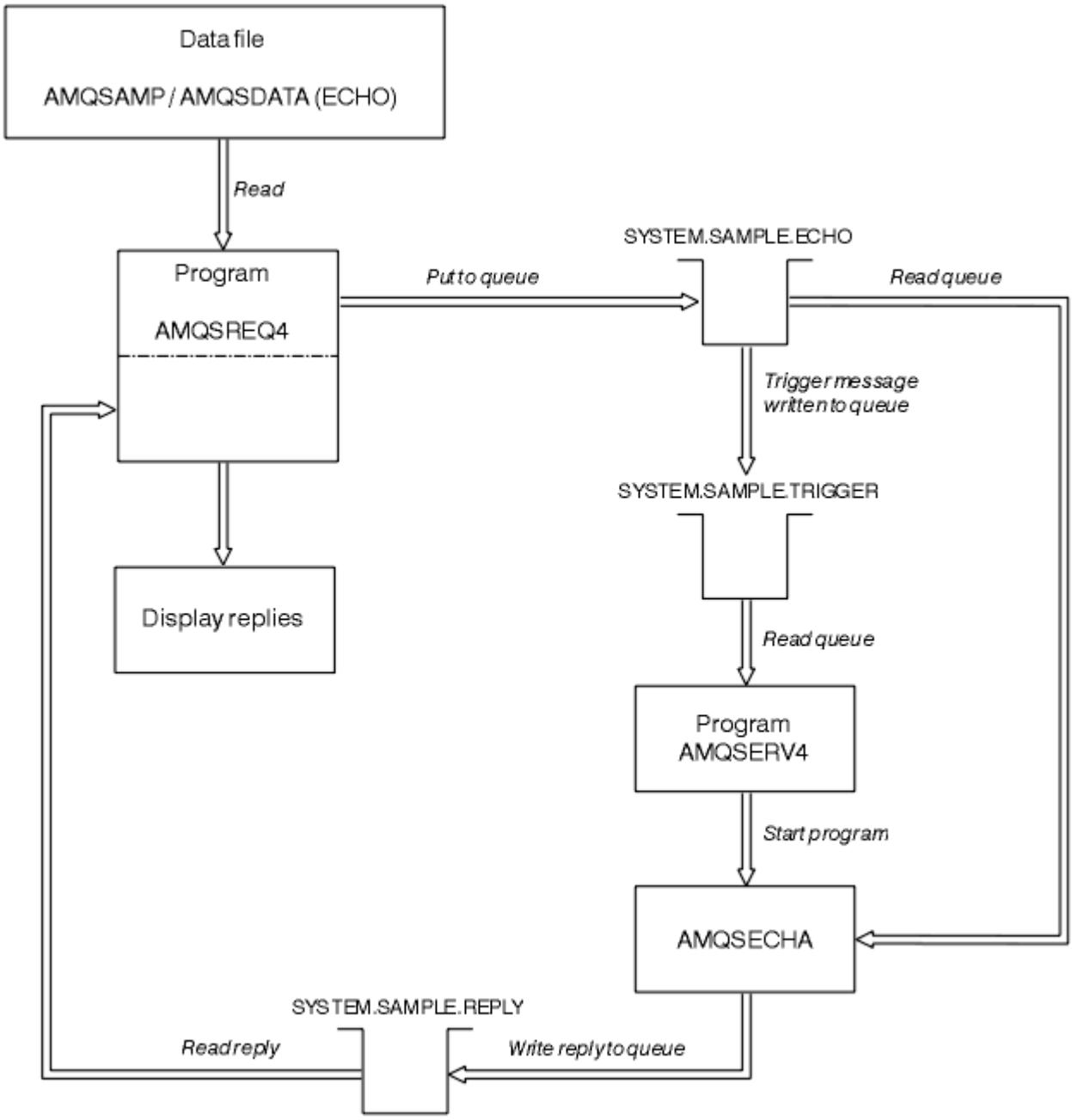


Figura 23. Diagramma di flusso del programma IBM i Client / Server (Echo) di esempio

I programmi di esempio Set

I programmi di esempio Set inibiscono le operazioni di inserimento su una coda utilizzando la chiamata MQSET per modificare l'attributo *InhibitPut* della coda. Inoltre, vengono fornite informazioni sulla progettazione di programmi di esempio Set.

Consultare "Funzioni dimostrate nei programmi di esempio" a pagina 98 per i nomi di questi programmi.

I programmi sono concepiti per essere eseguiti come programmi attivati, quindi il loro unico input è una struttura MQTMC2 (messaggio trigger) che contiene il nome di una coda di destinazione con gli attributi che devono essere interrogati. La versione C utilizza anche il nome del gestore code. La versione COBOL utilizza il gestore code predefinito.

Affinché il processo di attivazione funzioni, assicurarsi che il programma di esempio Set che si desidera utilizzare sia attivato dai messaggi in arrivo sulla coda SYSTEM.SAMPLE.SET. A tale scopo, specificare il nome del programma di esempio Set che si desidera utilizzare nel campo *ApplicId* della definizione del

processo SYSTEM.SAMPLE.SETPROCESS. La coda di esempio ha un tipo di trigger FIRST; se ci sono già dei messaggi sulla coda prima di eseguire l'esempio Richiesta, l'esempio Imposta non viene attivato dai messaggi inviati.

Una volta impostata correttamente la definizione:

- Per i sistemi UNIX, Linux e Windows , avviare il programma **runmqtrm** in una sessione, quindi avviare il programma amq in un altro.
- Per IBM i, avviare il programma AMQSERV4 in una sessione, quindi avviare il programma AMQSREQ4 in un altro. È possibile utilizzare AMQSTRG4 invece di AMQSERV4, ma i potenziali ritardi di inoltro dei lavori potrebbero rendere meno semplice seguire ciò che sta accadendo.

Utilizzare i programmi di esempio Richiesta per inviare messaggi di richiesta, ciascuno contenente solo un nome coda, alla coda SYSTEM.SAMPLE.SET. Per ogni messaggio di richiesta, i programmi di esempio Set inviano un messaggio di risposta contenente una conferma che le operazioni di inserimento sono state inibite sulla coda specificata. Le risposte vengono inviate alla coda di risposta specificata nel messaggio di richiesta.

Progettazione del programma di esempio Set

Il programma apre la coda denominata nella struttura del messaggio trigger che è stata passata quando è stata avviata. (Per chiarezza, chiameremo questa *coda di richiesta*.) Il programma utilizza la chiamata MQOPEN per aprire questa coda per l'input condiviso.

Il programma utilizza la chiamata MQGET per rimuovere i messaggi da questa coda. Questa chiamata utilizza le opzioni MQGMO_ACCEPT_TRUNCATED_MSG e MQGMO_WAIT, con un intervallo di attesa di 5 secondi. Il programma verifica il descrittore di ogni messaggio per verificare se si tratta di un messaggio di richiesta; in caso contrario, il programma elimina il messaggio e visualizza un messaggio di avvertenza.

Per ogni messaggio di richiesta rimosso dalla coda di richieste, il programma legge il nome della coda (che chiameremo *coda di destinazione*) contenuta nei dati e apre tale coda utilizzando la chiamata MQOPEN con l'opzione MQOO_SET. Quindi, il programma utilizza la chiamata MQSET per impostare il valore dell'attributo *InhibitPut* della coda di destinazione su MQQA_PUT_INIITED.

Se la chiamata MQSET ha esito positivo, il programma utilizza la chiamata MQPUT1 per inserire un messaggio di risposta nella coda di risposta. Questo messaggio contiene la stringa `PUT inhibited`.

Se la chiamata MQOPEN o MQSET ha esito negativo, il programma utilizza la chiamata MQPUT1 per inserire un messaggio `report` nella coda di risposta. Nel campo *Feedback* del descrittore del messaggio di questo messaggio di report è presente il codice motivo restituito dalla chiamata MQOPEN o MQSET, a seconda di quale ha avuto esito negativo.

Dopo la chiamata MQSET, il programma chiude la coda di destinazione utilizzando la chiamata MQCLOSE.

Quando non ci sono messaggi rimanenti sulla coda di richiesta, il programma chiude tale coda e si disconnette dal gestore code.

Il programma di esempio SSL/TLS

AMQSSLC è un programma C di esempio che illustra come utilizzare le strutture MQCNO e MQSCO per fornire informazioni di connessione client SSL/TLS sulla chiamata MQCONNX. Ciò consente a una applicazione MQI client di fornire la definizione del suo canale di connessione client e le impostazioni SSL/TLS al runtime senza una CCDT (client channel definition table).

Se viene fornito un nome connessione, il programma crea una definizione di canale di connessione client in una struttura MQCD.

Se viene fornito il nome d'origine del file del repository delle chiavi, il programma costruisce una struttura MQSCO; se viene fornito anche un URL del responder OCSP, il programma costruisce una struttura MQAIR del record delle informazioni di autenticazione.

Il programma si connette al gestore code utilizzando MQCONNX. Esso interroga e stampa il nome del gestore code a cui è connesso.

Questo programma deve essere collegato come applicazione client MQI. Tuttavia, può essere collegato come applicazione MQI regolare. Quindi, si collega semplicemente a un gestore code locale e ignora le informazioni di connessione client

AMQSSLC accetta i seguenti parametri, tutti facoltativi:

-m QmgrName
Nome del gestore code a cui connettersi

-c ChannelName
Nome del canale da utilizzare

-x ConnName
Nome connessione server

Parametri SSL/TLS:

Stem -k KeyRepos
Il nome della radice del file di repository chiavi. Questo è il percorso completo del file senza il suffisso .kdb. Ad esempio:

```
/home/user/client
C:\User\client
```

-s CipherSpec
La stringa CipherSpec del canale SSL/TLS corrispondente a SSLCIPH nella definizione del canale SVRCONN sul gestore code.

-f
Specifica che devono essere utilizzati solo algoritmi certificati FIPS 140-2.

-b VALUE1[,VALUE2...]
Specifica che devono essere utilizzati solo algoritmi compatibili con Suite B. Questo parametro è un elenco separato da virgole di uno o più dei seguenti valori: NONE,128_BIT,192_BIT. Questi valori hanno lo stesso significato di quelli per la variabile di ambiente MQSUITEB e l'equivalente impostazione EncryptionPolicySuiteB nella stanza SSL del file di configurazione client.

-p Politica
Specifica la politica di convalida del certificato da utilizzare. Può essere uno dei seguenti valori:

ANY
Applicare ciascuna delle politiche di convalida del certificato supportate dalla libreria dei socket protetti e accettare la catena di certificati se una delle politiche considera valida la catena di certificati. Questa impostazione può essere utilizzata per la massima retrocompatibilità con i vecchi certificati digitali che non sono conformi ai moderni standard di certificazione.

RFC5280
Applicare solo la politica di convalida del certificato conforme RFC 5280. Questa impostazione fornisce una convalida più rigorosa rispetto all'impostazione ANY, ma rifiuta alcuni certificati digitali meno recenti.

Il valore predefinito è qualsiasi.

Parametro di revoca certificato OCSP:

-o URL
L'URL del responder OCSP

Esecuzione del programma di esempio SSL/TLS

Per eseguire il programma di esempio SSL/TLS, è necessario prima configurare l'ambiente SSL o TLS. Si esegue quindi l'esempio dalla riga comandi, fornendo un numero di parametri.

Informazioni su questa attività

Le seguenti istruzioni eseguono il programma di esempio utilizzando i certificati personali. Modificando il comando è possibile, ad esempio, utilizzare i certificati CA e verificarne lo stato utilizzando un responder OCSP. Consultare le istruzioni contenute nell'esempio.

Procedura

1. Creare un gestore code denominato QM1. Per ulteriori informazioni, consultare crtmqm.
2. Creare un repository delle chiavi per il gestore code. Per ulteriori informazioni, consultare Impostazione di un repository delle chiavi su sistemi UNIX, Linux, and Windows.
3. Creare un archivio chiavi per il client. Chiamalo *clientkey.kdb*.
4. Creare un certificato personale per il gestore code. Per ulteriori informazioni, consultare Creazione di un certificato personale autofirmato su sistemi UNIX, Linux, and Windows.
5. Creare un certificato personale per il client.
6. Estrarre il certificato personale dal repository delle chiavi del server e aggiungerlo al repository del client. Per ulteriori informazioni, consultare Estrazione della parte pubblica di un certificato autofirmato da un repository delle chiavi su sistemi UNIX, Linux e Windows , e Aggiunta di un certificato CA (o della parte pubblica di un certificato autofirmato) in un repository delle chiavi, su sistemi UNIX, Linux o Windows.
7. Estrarre il certificato personale dal repository delle chiavi del client e aggiungerlo al repository delle chiavi del server.
8. Creare un canale di connessione server utilizzando il comando MQSC:

   ```
   DEFINE CHANNEL(QM1SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCIPH(NULL_SHA)
   ```

   Per ulteriori informazioni, consultare Canale di connessione server
9. Definire e avviare un listener del canale sul gestore code. Per ulteriori informazioni, vedere DEFINE LISTENER e START LISTENER.
10. Eseguire il programma di esempio utilizzando il comando seguente:

    ```
    AMQSSSLC -m QM1 -c QM1SVRCONN -x localhost
    -k "c:\Program Files\IBM\WebSphere MQ\clientkey" -s NULL_SHA
    -o http://dummy.OCSP.responder
    ```

Risultati

Il programma di esempio esegue le seguenti operazioni:

1. Si connette a qualsiasi gestore code specificato o al gestore code predefinito, utilizzando le opzioni specificate.
2. Apre il gestore code e ne interroga il nome.
3. Chiude il gestore code.
4. Si disconnette dal gestore code.

Se il programma di esempio viene eseguito correttamente, viene visualizzato un output simile al seguente esempio:

```
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
```

Se il programma di esempio rileva un problema, visualizza un messaggio di errore appropriato, ad esempio se si specifica un URL del risponditore OCSP non valido, si riceve il seguente messaggio:

```
MQCONNX ended with reason code 2553
```

Per un elenco di codici di errore, vedere Codici di errore API.

I programmi di esempio Trigger

La funzione fornita nell'esempio di trigger è una serie secondaria di quella fornita nel controllo trigger nel programma **runmqtrm** .

Consultare “Funzioni dimostrate nei programmi di esempio” a pagina 98 per i nomi di questi programmi.

Progettazione del campione di attivazione

Il programma di esempio di trigger apre la coda di avvio utilizzando la chiamata MQOPEN con l'opzione MQOO_INPUT_AS_Q_DEF. Richiama i messaggi dalla coda di iniziazione utilizzando la chiamata MQGET con le opzioni MQGMO_ACCEPT_TRUNCATED_MSG e MQGMO_WAIT, specificando un intervallo di attesa illimitato. Il programma elimina i campi *MsgId* e *CorrelId* prima di ogni chiamata MQGET per ottenere i messaggi in sequenza.

Una volta richiamato un messaggio dalla coda di iniziazione, il programma verifica il messaggio controllando la dimensione del messaggio per assicurarsi che sia della stessa dimensione di una struttura MQTM. Se questo test ha esito negativo, il programma visualizza un'avvertenza.

Per messaggi trigger validi, l'esempio di trigger copia i dati da questi campi: *ApplicId*, *EnvrData*, *Versione* *ApplType*. Gli ultimi due di questi campi sono numerici, pertanto il programma crea sostituzioni di caratteri da utilizzare in una struttura MQTMC2 per sistemi UNIX, Linuxe Windows .

L'esempio di trigger emette un comando di avvio per l'applicazione specificata nel campo *ApplicId* del messaggio di trigger e passa una struttura MQTMC2 o MQTMC (una versione di caratteri del messaggio di trigger). Nei sistemi UNIX, Linux e Windows , il campo *EnvData* viene utilizzato come estensione della stringa di comandi di richiamo.

Infine, il programma chiude la coda di iniziazione.

Esecuzione dei programmi di esempio Trigger

Questo argomento contiene informazioni sull'esecuzione dei programmi di esempio Triggering.

Esecuzione degli esempi amqstrg0.c, amqstrg e amqstrgc

Il programma accetta 2 parametri:

1. Il nome della coda di iniziazione (necessario)
2. Il nome del gestore code (facoltativo)

Se un gestore code non è specificato, si connette a quello predefinito. Una coda di iniziazione di esempio sarà stata definita quando è stato eseguito amqscos0.tst; il nome di tale coda è SYSTEM.SAMPLE.TRIGGERed è possibile utilizzarlo quando si esegue questo programma.

Nota: La funzione in questo esempio è un sottoinsieme della funzione di attivazione completa fornita nel programma **runmqtrm** .

Progettazione del server trigger

La progettazione del server trigger è simile a quella del controllo trigger, tranne che per il fatto che il server trigger:

- Consente MQAT_CICS e applicazioni MQAT_OS400
- Per applicazioni CICS , sostituisce *EnvData*, ad esempio, per specificare la regione CICS , dal messaggio trigger nel comando STRCICSUSR
- Apre la coda di iniziazione per l'input condiviso, in modo che molti server trigger possano essere eseguiti contemporaneamente

Nota: I programmi avviati da AMQSERV4 non devono utilizzare la chiamata MQDISC perché questo arresta il server trigger. Se i programmi avviati da AMQSERV4 utilizzano la chiamata MQCONN, ricevono il codice di errore MQRC_ALREADY_CONNECTED.

Esempi TUXEDO

Informazioni sui programmi di esempio Put e Get per TUXEDO e sulla creazione dell'ambiente server in TUXEDO.

Prima di eseguire questi esempi, è necessario creare l'ambiente server.

Nota: In questo argomento, il carattere barra retroversa (\) viene utilizzato per suddividere i comandi lunghi su più di una riga. Non immettere questo carattere. Immettere ciascun comando come riga singola.

Creazione di un ambiente server

Informazioni sulla creazione dell'ambiente server per WebSphere MQ per diverse piattaforme.

Si presume che l'utente disponga di un ambiente TUXEDO funzionante.

Creazione dell'ambiente server per WebSphere MQ per AIX (a 32 bit)

1. Creare una directory (ad esempio, < APPDIR>) in cui l'ambiente del server viene creato ed eseguire tutti i comandi in questa directory.
2. Esportare le seguenti variabili di ambiente, dove TUXDIR è la directory root per TUXEDO e *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato 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. Aggiungere quanto segue al file TUXEDO udataobj/RM

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa -lmqm
```

4. Eseguire i comandi:

```
$ 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. Modificare ubbstxcx.cfg e aggiungere i dettagli del nome della macchina, delle directory di lavoro e del gestore code come necessario:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Creare il TLOGDEVICE:

```
$tmadmin -c
```

Viene visualizzata una richiesta. A questo prompt, immettere:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Avviare il gestore code:

```
$ strmqm
```

8. Avviare Tuxedo:

```
$ tmboot -y
```

È ora possibile utilizzare i programmi doputs e dogets per inserire i messaggi in una coda e richiamarli da una coda.

Creazione di un ambiente server per WebSphere MQ per AIX (64 bit)

1. Creare una directory (ad esempio, < APPDIR>) in cui l'ambiente del server viene creato ed eseguire tutti i comandi in questa directory.
2. Esportare le seguenti variabili di ambiente, dove TUXDIR rappresenta la directory root per TUXEDO, e *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ 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. Aggiungere quanto segue al file TUXEDO udataobj/RM

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa64 -lmqm
```

4. Eseguire i comandi:

```
$ 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. Modificare ubbstxcx.cfg e aggiungere i dettagli del nome della macchina, delle directory di lavoro e del gestore code come necessario:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Creare il TLOGDEVICE:

```
$tmadmin -c
```

Viene visualizzata una richiesta. A questo prompt, immettere:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Avviare il gestore code:

```
$ strmqm
```

8. Avviare Tuxedo:

```
$ tmboot -y
```

È ora possibile utilizzare i programmi doputs e dogets per inserire i messaggi in una coda e richiamarli da una coda.

Creazione di un ambiente server per WebSphere MQ per Solaris (a 32 bit)

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

1. Creare una directory (ad esempio, *APPDIR*) in cui l'ambiente del server viene creato ed eseguire tutti i comandi in questa directory.
2. Esportare le seguenti variabili di ambiente, dove TUXDIR è la directory root per 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. Aggiungere quanto segue al file TUXEDO udataobj/RM.

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib/libmqmxa.a MQ_INSTALLATION_PATH/lib/libmqm.so \
/opt/tuxedo/lib/libtux.a
```

4. Eseguire i comandi:

```
$ 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. Modificare ubbstxcx.cfg e aggiungere i dettagli del nome della macchina, delle directory di lavoro e del gestore code come necessario:

```
$ tmloadcf    -y ubbstxcx.cfg
```

6. Creare il TLOGDEVICE:

```
$tmadmin -c
```

Viene visualizzata una richiesta. A questo prompt, immettere:

```
> crdl -z /APPDIR/TLOG1
```

7. Avviare il gestore code:

```
 $ strmqm
```

8. Avviare Tuxedo:

```
 $ tmboot -y
```

È ora possibile utilizzare i programmi doputs e dogets per inserire i messaggi in una coda e richiamarli da una coda.

Creazione dell'ambiente server per WebSphere MQ per Solaris (64 bit)

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

1. Creare una directory (ad esempio, < APPDIR>) in cui l'ambiente del server viene creato ed eseguire tutti i comandi in questa directory.
2. Esportare le seguenti variabili di ambiente, dove TUXDIR è la directory root per 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. Aggiungere quanto segue al file TUXEDO udataobj/RM.

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib64/libmqmxa64.a MQ_INSTALLATION_PATH/lib64/libmqm.so \
/opt/tuxedo/lib64/libtux.a
```

4. Eseguire i comandi:

```
$ 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. Modificare ubbstxcx.cfg e aggiungere i dettagli del nome della macchina, delle directory di lavoro e del gestore code come necessario:

```
$ tmloadcf    -y ubbstxcx.cfg
```

6. Creare il TLOGDEVICE:

```
$tmadmin -c
```

   Viene visualizzata una richiesta. A questo prompt, immettere:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Avviare il gestore code:

```
$ strmqm
```

8. Avviare Tuxedo:

```
$ tmboot -y
```

È ora possibile utilizzare i programmi doputs e dogets per inserire i messaggi in una coda e richiamarli da una coda.

Creazione dell'ambiente server per WebSphere MQ per HP-UX (32-bit)

Nota: L'ambiente del server TUXEDO a 32 bit può essere creato solo sulla piattaforma Itanium.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

1. Creare una directory (ad esempio, < APPDIR>) in cui l'ambiente del server viene creato ed eseguire tutti i comandi in questa directory.
2. Esportare le seguenti variabili di ambiente, dove TUXDIR è la directory root per 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. Aggiungere quanto segue al file TUXEDO udataobj/RM

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib/libmqmxa.so MQ_INSTALLATION_PATH/lib/libmqm.so \
/opt/tuxedo/lib/libtux.sl
```

4. Eseguire i comandi:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
```

   Dopo aver eseguito i comandi mkfldhdr e viewc, il file di intestazione amqstxvx.h viene creato nella directory dell'applicazione TUXEDO. Copiare questo file dalla directory dell'applicazione TUXEDO nella directory di inclusione TUXEDO ed eseguire i seguenti comandi.

```
$ 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. Modificare ubbstxcx.cfg e aggiungere i dettagli del nome della macchina, delle directory di lavoro e del gestore code come necessario:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Creare il TLOGDEVICE:

```
$tmadmin -c
```

   Viene visualizzata una richiesta. A questo prompt, immettere:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Avviare il gestore code:

```
$ strmqm
```

8. Avviare TUXEDO:

```
$ tmboot -y
```

È ora possibile utilizzare i programmi doputs e dogets per inserire i messaggi in una coda e richiamarli da una coda.

Creazione dell'ambiente server per WebSphere MQ per HP-UX (64 bit)

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

1. Creare una directory (ad esempio, < APPDIR>) in cui l'ambiente del server viene creato ed eseguire tutti i comandi in questa directory.
2. Esportare le seguenti variabili di ambiente, dove TUXDIR è la directory root per 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. Aggiungere quanto segue al file TUXEDO udataobj/RM

   Sulla piattaforma 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
```

   Nota: Le librerie WebSphere MQ fornite sulla piattaforma HP-UX IA64 (IPF) hanno un'estensione del nome file .so.

4. Eseguire i comandi:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
```

Dopo aver eseguito i comandi mkfldhdr e viewc, il file di intestazione amqstxvx.h viene creato nella directory dell'applicazione TUXEDO. Copiare questo file dalla directory dell'applicazione TUXEDO nella directory di inclusione TUXEDO ed eseguire i seguenti comandi.

```
$ buildtms    -o MQXA -r MQSeries_XA_RMI
```

Sulla piattaforma 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. Modificare ubbstxcx.cfg e aggiungere i dettagli del nome della macchina, delle directory di lavoro e del gestore code come necessario:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Creare il TLOGDEVICE:

```
$tmadmin -c
```

Viene visualizzata una richiesta. A questo prompt, immettere:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Avviare il gestore code:

```
$ strmqm
```

8. Avviare TUXEDO:

```
$ tmboot -y
```

È ora possibile utilizzare i programmi doputs e dogets per inserire i messaggi in una coda e richiamarli da una coda.

Creazione dell'ambiente server per WebSphere MQ per Windows (32 bit)

Nota: Modificare i campi identificati da <> nei seguenti percorsi di directory:

< MQMDIR>	il percorso di directory specificato quando è installato WebSphere MQ , ad esempio `g:\Program Files\IBM\WebSphere MQ`
< DIRTUX>	il percorso di directory specificato quando è stato installato TUXEDO, ad esempio `f:\tuxedo`

< DIRAPP>	il percorso della directory da utilizzare per l'applicazione di esempio, ad esempio `f:\tuxedo\apps\mqapp`

Per creare l'ambiente del server e gli esempi:

1. Creare una directory dell'applicazione in cui creare l'applicazione di esempio, ad esempio:

```
f:\tuxedo\apps\mqapp
```

2. Copiare i seguenti file di esempio dalla directory di esempio WebSphere MQ alla directory dell'applicazione:

```
amqstxmn.mak
amqstxen.env
ubbstxcn.cfg
```

3. Modificare ciascuno di tali file per impostare i nomi di directory e i percorsi di directory utilizzati sull'installazione.
4. Modificare ubbstxcn.cfg (consultare Figura 24 a pagina 165) per aggiungere i dettagli del nome macchina e del gestore code a cui si desidera connettersi.
5. Aggiungere la riga seguente al file TUXEDO < TUXDIR> udataobj\rm

```
MQSeries_XA_RMI;MQRMIXASwitchDynamic;
   <MQMDIR>\tools\lib\mqmxa.lib <MQMDIR>\tools\lib\mqm.lib
```

   dove < MQMDIR> viene sostituito come mostrato nell'esempio precedente. Anche se viene visualizzata qui come due righe, la nuova voce deve essere una riga nel file.
6. Impostare le seguenti variabili di ambiente:

```
TUXDIR=<TUXDIR>
TUXCONFIG=<APPDIR>\tuxconfig
FIELDTBLS=<MQMDIR>\tools\c\samples\amqstxvx.fld
LANG=C
```

7. Creare una periferica TLOG per TUXEDO. A tale scopo, richiamare `tmadmin -c` e immettere il comando:

```
crdl -z <APPDIR>\TLOG
```

   dove <*APPDIR*> viene sostituito.
8. Impostare la directory corrente su < APPDIR> e richiamare il makefile di esempio (amqstxmn.mak) come makefile del progetto esterno. Ad esempio, con Microsoft Visual C++, immettere il comando:

```
msvc amqstxmn.mak
```

   Selezionare **build** per creare tutti i programmi di esempio.

```
*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
```

Figura 24. Esempio di file ubbstxcn.cfg per WebSphere MQ per Windows

Nota: Modificare i nomi e i percorsi delle directory in modo che corrispondano all'installazione. Modificare anche il nome del gestore code MYQUEUEMANAGER nel nome del gestore code a cui si desidera connettersi. Le altre informazioni che è necessario aggiungere sono identificate dai caratteri <> .

Il file ubbconfig di esempio per WebSphere MQ per Finestre è elencato in Figura 24 a pagina 165. Viene fornito come ubbstxcn.cfg nella directory degli esempi di WebSphere MQ .

Il makefile di esempio (consultare Figura 25 a pagina 166) fornito per WebSphere MQ per Windows è denominato ubbstxmn.make si trova nella directory degli esempi di 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
```

Figura 25. Makefile TUXEDO di esempio per WebSphere MQ per Windows

Creazione dell'ambiente server per WebSphere MQ per Windows (64 bit)

Nota: Modificare i campi identificati da <> nei seguenti percorsi di directory:

< MQMDIR>	il percorso di directory specificato quando è installato WebSphere MQ , ad esempio `g:\Program Files\IBM\WebSphere MQ`
< DIRTUX>	il percorso di directory specificato quando è stato installato TUXEDO, ad esempio `f:\tuxedo`
< DIRAPP>	il percorso della directory da utilizzare per l'applicazione di esempio, ad esempio `f:\tuxedo\apps\mqapp`

Per creare l'ambiente del server e gli esempi:

1. Creare una directory dell'applicazione in cui creare l'applicazione di esempio, ad esempio:

   ```
   f:\tuxedo\apps\mqapp
   ```

2. Copiare i seguenti file di esempio dalla directory di esempio WebSphere MQ alla directory dell'applicazione:

   ```
   amqstxmn.mak
   amqstxen.env
   ubbstxcn.cfg
   ```

3. Modificare ciascuno di tali file per impostare i nomi di directory e i percorsi di directory utilizzati sull'installazione.
4. Modificare `ubbstxcn.cfg` (consultare Figura 26 a pagina 167) per aggiungere i dettagli del nome della macchina e del gestore code a cui si desidera connettersi.
5. Aggiungere la seguente riga nel file TUXEDO <TUXDIR>`udataobj\rm`

   ```
   MQSeries_XA_RMI;MQRMIXASwitchDynamic;
      <MQMDIR>\tools\lib64\mqmxa64.lib <MQMDIR>\tools\lib64\mqm.lib
   ```

dove < MQMDIR> viene sostituito. Anche se viene visualizzata qui come due righe, la nuova voce deve essere una riga nel file.

6. Impostare le seguenti variabili di ambiente:

```
TUXDIR=<TUXDIR>
TUXCONFIG=<APPDIR>\tuxconfig
FIELDTBLS=<MQMDIR>\tools\c\samples\amqstxvx.fld
LANG=C
```

7. Creare una periferica TLOG per TUXEDO. A tale scopo, richiamare `tmadmin -c` e immettere il comando:

```
crdl -z <APPDIR>\TLOG
```

dove <*APPDIR*> viene sostituito come mostrato nell'esempio precedente.

8. Impostare la directory corrente su < APPDIR> e richiamare il makefile di esempio (amqstxmn.mak) come makefile del progetto esterno. Ad esempio, con Microsoft Visual C++, immettere il comando:

```
msvc amqstxmn.mak
```

Selezionare **build** per creare tutti i programmi di esempio.

```
*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
```

Figura 26. Esempio di file ubbstxcn.cfg per WebSphere MQ per Windows

Nota: Modificare i nomi e i percorsi delle directory in modo che corrispondano all'installazione. Modificare anche il nome del gestore code MYQUEUEMANAGER nel nome del gestore code a cui si desidera connettersi. Le altre informazioni che è necessario aggiungere sono identificate dai caratteri <> .

Il file ubbconfig di esempio per WebSphere MQ per Finestre è elencato in Figura 26 a pagina 167. Viene fornito come ubbstxcn.cfg nella directory degli esempi di WebSphere MQ .

Il makefile di esempio (consultare Figura 27 a pagina 168) fornito per WebSphere MQ per Windows è denominato ubbstxmn.make si trova nella directory degli esempi di 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
```

Figura 27. Makefile TUXEDO di esempio per WebSphere MQ per Windows

Programma server di esempio per TUXEDO

Il programma del server di esempio (amqstxsx) è progettato per essere eseguito con i programmi di esempio Put (amqstxpx.c) e Get (amqstxgx.c). Il programma server di esempio viene eseguito automaticamente quando TUXEDO viene avviato.

Nota: È necessario avviare il gestore code ***prima*** di avviare TUXEDO.

Il server di esempio fornisce due servizi TUXEDO, MPUT1 e MGET1:

- Il servizio MPUT1 è guidato dall'esempio PUT e utilizza MQPUT1 nel punto di sincronizzazione per inserire un messaggio in un'unità di lavoro controllata da TUXEDO. Prende i parametri QName e Message Text, che sono forniti dall'esempio PUT.
- Il servizio MGET1 si apre e chiude la coda ogni volta che riceve un messaggio. Acquisisce i parametri QName e Message Text, forniti dall'esempio GET.

Tutti i messaggi di errore, i codici di errore e i messaggi di stato vengono scritti nel file di log TUXEDO.

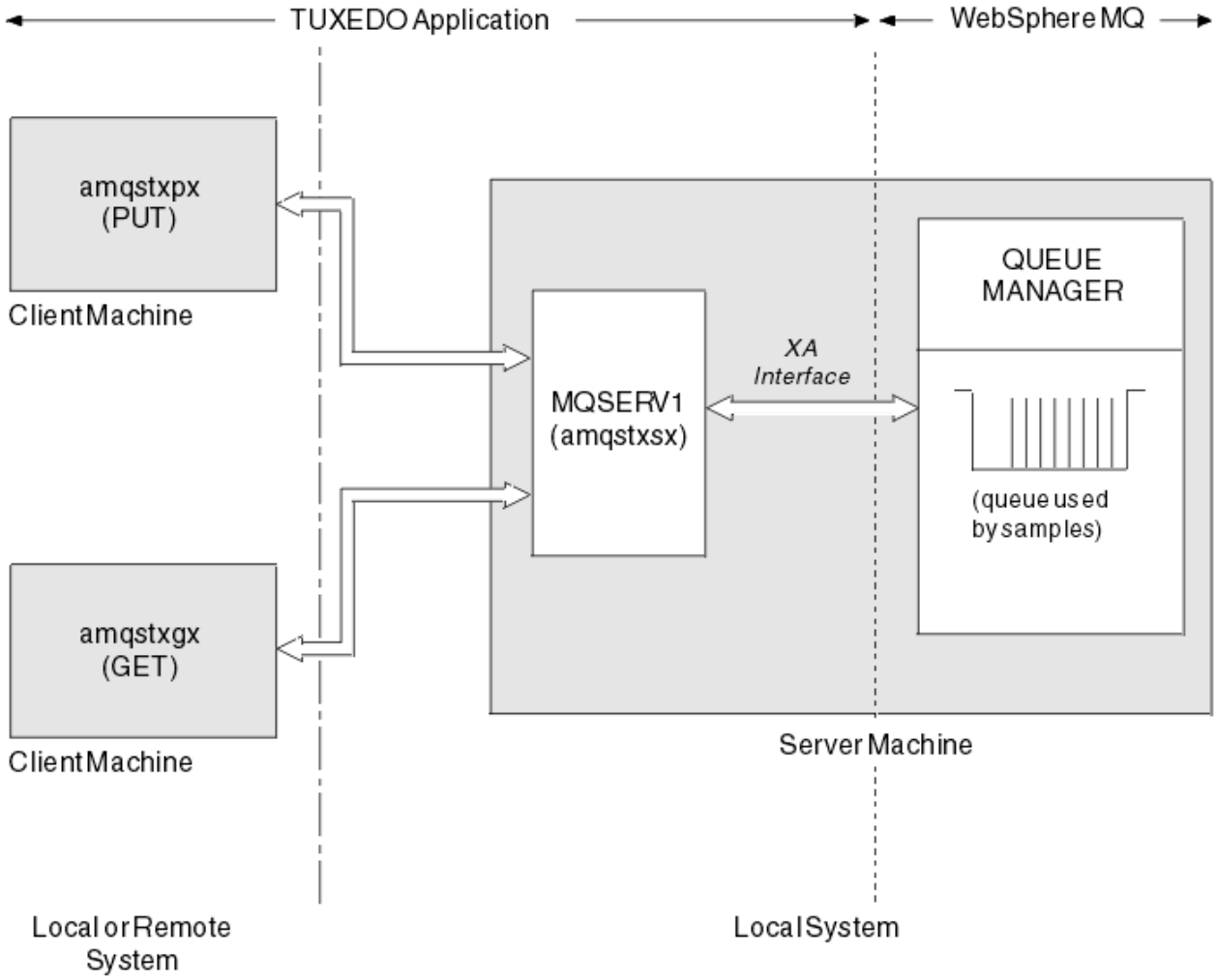


Figura 28. Funzionamento degli esempi TUXEDO

Programma di esempio di inserimento per TUXEDO

Questo esempio consente di inserire un messaggio in una coda più volte, in batch, dimostrando il punto di sincronizzazione utilizzando TUXEDO come gestore risorse.

Il programma del server di esempio `amqstxsx` deve essere in esecuzione affinché l'esempio di inserimento abbia successo; il programma di esempio del server si connette al gestore code e utilizza l'interfaccia XA. Per eseguire l'esempio, immettere:

- `doputs -n queuename -b batchsize -c trancount -t message`

Ad esempio:

- `doputs -n myqueue -b 5 -c 6 -t "Hello World"`

Ciò inserisce 30 messaggi nella coda denominata `myqueue`, in sei batch, ciascuno con cinque messaggi. In caso di problemi, esegue il backup di un batch di messaggi, altrimenti ne esegue il commit.

Tutti i messaggi di errore vengono scritti nel file di log TUXEDO e in stderr. Tutti i codici di errore vengono scritti in stderr.

Ottieni esempio per TUXEDO

Questo esempio consente di richiamare i messaggi da una coda in batch.

Il programma del server di esempio `amqstxsx` deve essere in esecuzione affinché l'esempio di inserimento abbia successo; il programma di esempio del server si connette al gestore code e utilizza l'interfaccia XA. Per eseguire l'esempio, immettere:

• `dogets -n queuename -b batchsize -c trancount`

Ad esempio:

• `dogets -n myqueue -b 6 -c 4`

Ciò toglie 24 messaggi dalla coda denominata `myqueue`, in sei batch, ognuno con quattro messaggi. Se si esegue questa operazione dopo l'esempio di inserimento, che inserisce 30 messaggi su `myqueue`, si hanno solo sei messaggi su `myqueue`. Il numero di batch e la relativa dimensione possono variare tra l'inserimento e il richiamo dei messaggi.

Tutti i messaggi di errore vengono scritti nel file di log TUXEDO e in stderr. Tutti i codici di errore vengono scritti in stderr.

Utilizzo dell'uscita di sicurezza SSPI su sistemi Windows

Questo argomento descrive come utilizzare i programmi di uscita canale SSPI sui sistemi Windows . Il codice di uscita fornito è in due formati: oggetto e origine.

Codice oggetto

Il file di codice oggetto è denominato amqrspin.dll. Per client e server, viene installato come parte standard di WebSphere MQ per Finestre nella cartella *MQ_INSTALLATION_PATH*`/exits/`*INSTALLATION_NAME* . Ad esempio, `C:\Program Files\IBM\WebSphere MQ\exits\installation2`. Viene caricato come uscita utente standard. È possibile eseguire l'uscita del canale di sicurezza fornita e utilizzare i servizi di autenticazione nella definizione di canale.

A tale scopo, specificare una delle seguenti opzioni:

```
SCYEXIT('amqrspin(SCY_KERBEROS)')

SCYEXIT('amqrspin(SCY_NTLM)')
```

Per fornire il supporto per un canale limitato, specificare quanto segue sul canale SVRCONN:

```
SCYDATA('remote_principal_name')
```

dove *nome_principale_remoto* è nel formato DOMINIO\utente. Il canale sicuro viene stabilito solo se il nome del principal remoto corrisponde a *nome_principale_remoto*.

Per utilizzare i programmi di uscita canale forniti tra i sistemi che operano all'interno di un dominio di sicurezza Kerberos , creare un nome servicePrincipalper il gestore code.

Codice sorgente

Il file del codice sorgente di uscita è denominato amqsspin.c. Si trova in `C:\Program Files\IBM\WebSphere MQ\Tools\c\Samples`.

Se si modifica il codice sorgente, è necessario ricompilare l'origine modificata.

È possibile compilarlo e collegarlo allo stesso modo di qualsiasi altra uscita del canale per la piattaforma pertinente, ad eccezione del fatto che è necessario accedere alle intestazioni SSPI al momento della compilazione e che le librerie di sicurezza SSPI, insieme alle librerie associate consigliate, devono essere accessibili al momento del collegamento.

Prima di eseguire questo comando, assicurarsi che cl.exe, la libreria Visual C++ e la cartella `include` siano disponibili nel percorso. Ad esempio:

```
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
```

Nota: Il codice di origine non include alcuna disposizione per la traccia o la gestione degli errori. Se si modifica e si utilizza il codice sorgente, aggiungere le proprie routine di traccia e di gestione degli errori.

Esecuzione degli esempi utilizzando le code remote

È possibile dimostrare l'accodamento remoto eseguendo gli esempi sui gestori code connessi.

Il programma amqscos0.tst fornisce una definizione locale di una coda remota (SYSTEM.SAMPLE.REMOTE) che utilizza un gestore code remoto denominato OTHER. Per utilizzare questa definizione di esempio, modificare OTHER con il nome del secondo gestore code che si desidera utilizzare. Inoltre, è necessario impostare un canale di messaggi tra i due gestori code; per informazioni su come eseguire questa operazione, vedere Definizione dei canali.

I programmi di esempio di richiesta inseriscono il loro nome gestore code locale nel campo *ReplyToQMgr* dei messaggi che inviano. Gli esempi Inquire e Set inviano i messaggi di risposta alla coda e al gestore code dei messaggi denominati nei campi *ReplyToQ* e *ReplyToQMgr* dei messaggi di richiesta elaborati.

Programma di esempio Monitoraggio coda cluster (AMQSCLM)

Questo esempio utilizza le funzioni di bilanciamento del carico di lavoro del cluster IBM WebSphere MQ integrate per indirizzare i messaggi alle istanze delle code a cui sono collegate le applicazioni che utilizzano. Questa direzione automatica impedisce la creazione di messaggi su un'istanza di una coda cluster a cui non è collegata alcuna applicazione di consumo.

Panoramica

È possibile impostare un cluster che abbia più di una definizione per la stessa coda su gestori code differenti. Questa configurazione fornisce il vantaggio di una maggiore disponibilità e bilanciamento del carico di lavoro. Tuttavia, non vi è alcuna funzionalità integrata in IBM WebSphere MQ per modificare in modo dinamico la distribuzione dei messaggi in un cluster in base allo stato delle applicazioni collegate. Per questo motivo, un'applicazione di consumo deve essere sempre collegata a ogni istanza di una coda per garantire che i messaggi vengano elaborati.

Il programma di esempio di monitoraggio della coda cluster monitora lo stato delle applicazioni collegate. Il programma regola in modo dinamico la configurazione di bilanciamento del carico di lavoro integrato per indirizzare i messaggi alle istanze di una coda cluster con applicazioni di consumo collegate. In alcune situazioni questo programma può essere utilizzato per allentare la necessità che un'applicazione di consumo sia sempre connessa a ogni istanza di una coda. Reinvia anche i messaggi che vengono accodati su un'istanza di una coda senza applicazioni collegate. Il nuovo invio dei messaggi consente di instradare i messaggi verso un'applicazione che utilizza e che viene temporaneamente chiusa.

Il programma è progettato per essere utilizzato dove le applicazioni di consumo sono applicazioni di lunga durata, piuttosto che collegare e scollegare frequentemente le applicazioni.

Il programma di esempio di controllo della coda cluster è il programma eseguibile compilato del file di esempio C `amqsclma.c`.

Ulteriori informazioni sui cluster e sul carico di lavoro sono disponibili in Utilizzo dei cluster per la gestione del carico di lavoro

AMQSCLM: Progettazione e pianificazione dell'utilizzo dell'esempio

Le informazioni relative al funzionamento del programma di esempio di monitoraggio della coda cluster, puntano a considerare quando si imposta un sistema su cui eseguire il programma di esempio e le modifiche che possono essere apportate al codice sorgente di esempio.

Progetta

Il programma di esempio di monitoraggio della coda del cluster monitora le code cluster locali che dispongono di applicazioni collegate. Il programma controlla le code specificate dall'utente. Il nome della coda potrebbe essere specifico, ad esempio `APP.TEST01`o generico. I nomi generici devono avere un

formato conforme a PCF (Programmable Command Format). Esempi di nomi generici sono APP.TEST*o APP*.

Ogni gestore code in un cluster che possiede un'istanza di una coda locale da monitorare, richiede la connessione di un'istanza del programma di esempio di monitoraggio della coda del cluster.

Instradamento messaggio dinamico

Il programma di esempio di monitoraggio della coda cluster utilizza il valore **IPPROCS** (aperto per il conteggio del processo di input) di una coda per determinare se tale coda ha dei consumer. Un valore maggiore di 0 indica che la coda ha almeno un'applicazione di consumo collegata. Tali code sono attive. Il valore 0 indica che la coda non ha programmi di consumo collegati. Tali code sono inattive.

Per una coda cluster con più istanze in un cluster, WebSphere MQ utilizza la proprietà di priorità del carico di lavoro del cluster **CLWLPRTY** di ogni istanza della coda per stabilire a quali istanze inviare messaggi. WebSphere MQ invia messaggi alle istanze disponibili di una coda con il più alto valore **CLWLPRTY** .

Il programma di esempio di controllo della coda cluster attiva una coda cluster impostando il valore **CLWLPRTY** locale su 1. Il programma disattiva una coda cluster impostando il valore **CLWLPRTY** su 0.

La tecnologia di cluster WebSphere MQ propaga la proprietà **CLWLPRTY** aggiornata di una coda cluster a tutti i gestori code pertinenti nel cluster. Ad esempio,

- Un gestore code con un'applicazione collegata che inserisce messaggi nella coda.
- Un gestore code che possiede una coda locale con lo stesso nome nello stesso cluster.

La propagazione viene eseguita utilizzando i gestori code del repository completo del cluster. I nuovi messaggi per la coda del cluster vengono indirizzati alle istanze con il valore **CLWLPRTY** più alto all'interno del cluster.

Trasferimento messaggio accodato

La modifica dinamica del valore di **CLWLPRTY** influenza l'instradamento dei nuovi messaggi. Questa modifica dinamica non influisce sui messaggi già accodati su un'istanza della coda senza utenti collegati o sui messaggi che hanno attraversato il meccanismo di bilanciamento del carico di lavoro prima che un valore **CLWLPRTY** modificato venisse propagato nel cluster. Di conseguenza, i messaggi rimangono su qualsiasi coda inattiva e non vengono elaborati da un'applicazione che li utilizza. Per risolvere questo problema, il programma di esempio di controllo della coda del cluster è in grado di richiamare i messaggi da una coda locale senza consumer e di inviare tali messaggi alle istanze remote della stessa coda in cui sono collegati i consumer.

Il programma di esempio di controllo della coda cluster trasferisce i messaggi da una coda locale inattiva a una o più code remote attive richiamando i messaggi (utilizzando **MQGET**) e inserendo i messaggi (utilizzando **MQPUT**) nella stessa coda cluster. Questo trasferimento fa in modo che la gestione del carico di lavoro del cluster WebSphere MQ selezioni un'istanza di destinazione differente, in base a un valore CLWLPRTY superiore rispetto a quello dell'istanza della coda locale. La persistenza e il contesto del messaggio vengono conservati durante il trasferimento del messaggio. L'ordine dei messaggi e le opzioni di bind non vengono conservate.

Pianificazione

Il programma di esempio di monitoraggio della coda del cluster modifica la configurazione del cluster quando si verifica una modifica nella connettività delle applicazioni utilizzate. Le modifiche vengono trasmesse dai gestori code in cui il programma di esempio di monitoraggio della coda del cluster sta monitorando le code, ai gestori code del repository completo nel cluster. I gestori code del repository completo elaborano gli aggiornamenti della configurazione e li inviano nuovamente a tutti i gestori code pertinenti nel cluster. I gestori code rilevanti includono i gestori code che possiedono code cluster con lo stesso nome (dove è in esecuzione un'istanza del programma di esempio di monitoraggio della coda del cluster) e qualsiasi gestore code in cui un'applicazione ha aperto la coda del cluster per inserire i messaggi negli ultimi 30 giorni.

Le modifiche vengono elaborate asincrono nel cluster. Pertanto, dopo ogni modifica, diversi gestori code nel cluster potrebbero avere viste diverse della configurazione per un periodo di tempo.

Il programma di esempio di monitoraggio della coda cluster è adatto solo per i sistemi in cui le applicazioni che utilizzano raramente si collegano o si scollegano; ad esempio, le applicazioni che utilizzano da lungo tempo. Quando viene utilizzato per monitorare i sistemi in cui le applicazioni che utilizzano sono collegate solo per brevi periodi, la latenza che si verifica durante la distribuzione degli aggiornamenti della configurazione potrebbe far sì che i gestori code nel cluster abbiano una vista non corretta delle code in cui sono collegati i consumer. Questa latenza potrebbe causare messaggi instradati in modo non corretto.

Quando si esegue il monitoraggio di molte code, una frequenza relativamente bassa di modifica nei consumer collegati in tutte le code potrebbe aumentare il traffico di configurazione del cluster nel cluster. L'aumento del traffico di configurazione cluster può causare un carico eccessivo su uno o più dei seguenti gestori code.

- I gestori code in cui è in esecuzione il programma di esempio di monitoraggio della coda cluster
- I gestori code del repository completo
- Un gestore code con un'applicazione connessa che inserisce i messaggi nella coda
- Un gestore code che possiede una coda locale con lo stesso nome nello stesso cluster

È necessario valutare l'utilizzo del processore sui gestori code del repository completo. Un ulteriore utilizzo del processore è visibile come traffico di messaggi sulla coda del repository completo SYSTEM.CLUSTER.COMMAND.QUEUE. Se i messaggi si accumulano su tale coda, ciò indica che i gestori code del repository completo non sono in grado di tenere il passo con la frequenza di modifica della configurazione del cluster nel sistema.

Quando molte code vengono monitorate dal programma di esempio di monitoraggio della coda cluster, il programma di esempio e il gestore code eseguono una quantità di lavoro. Questo lavoro viene eseguito, anche quando non vi sono modifiche ai consumatori collegati. L'argomento **-i** può essere modificato per diminuire l'utilizzo del processore del programma di esempio sul sistema locale, diminuendo la frequenza del ciclo di monitoraggio.

Per facilitare il rilevamento di un'attività eccessiva, il programma di esempio di monitoraggio della coda del cluster riporta il tempo di elaborazione medio per l'intervallo di polling, il tempo di elaborazione trascorso e il numero di modifiche alla configurazione. I report vengono consegnati in un messaggio informativo, **CLM0045I**, ogni 30 minuti o ogni 600 intervalli di polling, a seconda di quale dei due si verifica prima.

Requisiti di utilizzo del monitoraggio della coda cluster

Il programma di esempio di monitoraggio della coda del cluster ha requisiti e limitazioni. È possibile modificare il codice sorgente di esempio fornito per modificare alcune di tali limitazioni in modo che possa essere utilizzato. Gli esempi elencati in questa sezione descrivono le modifiche che è possibile apportare.

- Il programma di esempio di monitoraggio della coda cluster è progettato per essere utilizzato per monitorare le code in cui le applicazioni che utilizzano sono collegate o non collegate. Se il sistema utilizza applicazioni che si collegano e scollegano frequentemente, il programma di esempio potrebbe generare un'eccessiva attività di configurazione del cluster nell'intero cluster. Ciò potrebbe avere un impatto sulle prestazioni dei gestori code nel cluster.
- Il programma di esempio di monitoraggio della coda del cluster dipende dalla tecnologia cluster e dal sistema WebSphere MQ sottostante. Il numero di code monitorate, la frequenza di monitoraggio e la frequenza di modifica dello stato di ciascuna coda influiscono sul carico sul sistema globale. Questi fattori devono essere considerati quando si selezionano le code da monitorare e l'intervallo di polling del monitoraggio.
- Un'istanza del programma di esempio di monitoraggio della coda del cluster deve essere connessa a ogni gestore code nel cluster che possiede un'istanza di coda da monitorare. Non è necessario connettere il programma di esempio ai gestori code nel cluster che non possiedono le code.

- Il programma di esempio di monitoraggio della coda cluster deve essere eseguito con l'autorizzazione appropriata per accedere a tutte le risorse WebSphere MQ richieste. Ad esempio,
  - Il gestore code a cui connettersi
  - Il SISTEMA SYSTEM.ADMIN.COMMAND.QUEUE
  - Tutte le code da monitorare quando viene effettuato il trasferimento del messaggio
- Il server dei comandi deve essere in esecuzione per ogni gestore code con il programma di esempio di monitoraggio della coda cluster connesso.
- Ogni istanza del programma di esempio di monitoraggio della coda cluster richiede l'uso esclusivo di una coda locale (non cluster) sul gestore code a cui è connessa. Questa coda locale viene utilizzata per controllare il programma di esempio e ricevere messaggi di risposta da richieste effettuate al server dei comandi del gestore code.
- Tutte le code che devono essere monitorate da una singola istanza del programma di esempio di monitoraggio della coda cluster devono trovarsi nello stesso cluster. Se un gestore code ha code in più cluster che richiedono il controllo, sono richieste più istanze del programma di esempio. Per ogni istanza è necessaria una coda locale per i messaggi di controllo e di risposta.
- Tutte le code da monitorare devono trovarsi in un singolo cluster. Le code configurate per utilizzare un elenco nomi cluster non vengono monitorate.
- L'abilitazione del trasferimento dei messaggi dalle code inattive è facoltativa. Si applica a tutte le code controllate dall'istanza del programma di esempio di monitoraggio della coda cluster. Se solo un sottoinsieme delle code monitorate richiede il trasferimento del messaggio abilitato, sono necessarie due istanze del programma di esempio di monitoraggio della coda cluster. Un programma di esempio ha il trasferimento messaggi abilitato e l'altro ha il trasferimento messaggi disabilitato. Ciascuna istanza del programma di esempio necessita di una coda locale per i messaggi di controllo e di risposta.
- WebSphere MQ il bilanciamento del carico di lavoro del cluster invierà, per impostazione predefinita, i messaggi alle istanze delle code con cluster che si trovano sullo stesso gestore code a cui è connessa un'applicazione di inserimento. Questo deve essere disabilitato mentre la coda locale è inattiva nelle seguenti circostanze:
  - Le applicazioni di inserimento si connettono ai gestori code che possiedono le istanze di una coda inattiva monitorata
  - I messaggi in coda vengono trasferiti dalle code inattive alle code attive.

  La preferenza di bilanciamento del carico di lavoro locale sulla coda può essere disabilitata staticamente, tramite l'impostazione del valore CLWLUSEQ su ANY. In questa configurazione i messaggi inseriti nelle code locali vengono distribuiti alle istanze della coda locale e remota per bilanciare il carico di lavoro, anche quando sono presenti applicazioni che utilizzano il sistema locale. In alternativa, è possibile configurare il programma di esempio di monitoraggio della coda del cluster per impostare temporaneamente il valore **CLWLUSEQ** su ANY mentre la coda non ha consumer collegati, il che fa sì che solo i messaggi locali vengano inviati alle istanze locali di una coda mentre tale coda è attiva.
- Il sistema WebSphere MQ e le applicazioni non devono utilizzare **CLWLPRTY** per le code da monitorare o per i canali utilizzati. Altrimenti, le azioni del programma di esempio di monitoraggio della coda cluster sugli attributi della coda **CLWLPRTY** potrebbero avere effetti indesiderati.
- Il programma di esempio di monitoraggio della coda del cluster registra le informazioni di runtime in una serie di file di report. È necessaria una directory per memorizzare questi report e il programma di esempio di monitoraggio della coda cluster deve disporre dell'autorizzazione per scrivervi.

AMQSCLM: Preparazione ed esecuzione dell'esempio

Per poter eseguire l'esempio di monitoraggio della coda cluster, è necessario configurare il gestore code per accettare in modo sicuro le richieste di connessione in entrata dalle applicazioni in esecuzione in modalità client.

Prima di iniziare

I seguenti passi devono essere completati prima di eseguire l'esempio di monitoraggio della coda cluster.

1. Creare una coda di lavoro su ciascun gestore code per l'utilizzo interno dell'esempio.

   Ogni istanza dell'esempio richiede una coda non cluster locale per uso interno esclusivo. È possibile scegliere il nome della coda. L'esempio utilizza il nome AMQSCLM.CONTROL.QUEUE. Ad esempio, in Windows, è possibile creare questa coda utilizzando il comando **MQSC**

   ```
   DEFINE QLOCAL(AMQSCLM.CONTROL.QUEUE)
   ```

   È possibile lasciare i valori di **MAXDEPTH** e **MAXMSGL** come predefiniti.

2. Creare una directory per i log dei messaggi di errore e informativi.

   L'esempio scrive i messaggi diagnostici nei file di report. È necessario scegliere una directory in cui memorizzare i file. Ad esempio, in Windows, è possibile creare una directory utilizzando il seguente comando:

   ```
   mkdir C:\AMQSCLM\rpts
   ```

   I file di report creati dall'esempio hanno la seguente convenzione di denominazione:

   ```
   QmgrName.ClusterName.RPT0n.LOG
   ```

3. (Facoltativo) Definire l'esempio di monitoraggio della coda cluster come un servizio IBM WebSphere MQ .

   Per monitorare le code, l'esempio deve essere sempre in esecuzione. Per accertarsi che l'esempio di controllo della coda del cluster sia sempre in esecuzione, è possibile definire l'esempio come un servizio gestore code. La definizione dell'esempio come servizio significa che AMQSCLM viene avviato all'avvio del gestore code. Puoi utilizzare il seguente esempio **RUNMQSC** per definire l'esempio di monitoraggio della coda del cluster come servizio IBM WebSphere MQ .

   ```
   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')
   ```

   dove < Root di installazione> è l'ubicazione dell'installazione.

Definizione	Descrizione
service	Specifica il nome del servizio. È possibile scegliere il nome servizio.
descr	Specifica una descrizione testuale del servizio.
control	Indica che il servizio viene avviato e arrestato contemporaneamente al gestore code.
servtype	Indica un oggetto servizio del server, che indica che è possibile eseguire una sola istanza alla volta per questo gestore code.
startcmd	Specifica l'ubicazione e nome del programma.
startarg	Specifica gli argomenti dell'esempio. Notare l'utilizzo di *+ QMNAME +*. Il nome del gestore code viene sostituito automaticamente.
stdout	Il nome file completo a cui viene reindirizzato l'output standard. L'esempio scrive in questo file solo i messaggi che confermano che l'esempio è terminato. L'esempio fa ciò perché il file di errore standard è già stato chiuso in una fase precedente del processo di terminazione dell'esempio.
stderr	Il nome file completo a cui viene reindirizzato l'output di errore standard. L'esempio scrive nel file di errore standard tutti i messaggi di errore prima della chiusura dell'esempio.

Informazioni su questa attività

Questa attività consente di avviare e arrestare l'esempio di monitoraggio della coda cluster in modi diversi. Inoltre, consente di eseguire l'esempio in una modalità che genera file di report contenenti informazioni statistiche sulle code monitorate.

Il programma di esempio può essere eseguito utilizzando il seguente comando.

```
AMQSCLM -m QMgrName -c ClusterName (-q QNameMask | -f QListFile) -r MonitorQName
[-l ReportDir] [-t] [-u ActiveVal] [-i Interval] [-d] [-s] [-v]
```

La tabella elenca gli argomenti che è possibile utilizzare con l'esempio di controllo della coda cluster, insieme ad ulteriori informazioni su ciascuno di essi.

Argomento	Variabile	Ulteriori informazioni
-m	**QMgrName**	Il gestore code da controllare.
-c	**ClusterName**	Il cluster contenente le code da monitorare.
-q	**QNameMask**	La coda o le code da monitorare. Un * finale monitora tutte le code con nomi che corrispondono a zero o più caratteri finali.
-f	**QListFile**	Il percorso completo e il nome file di un file che contiene un elenco di nomi coda di maschere nome coda da monitorare. Il file deve contenere un nome coda / maschera per riga. È possibile specificare **-q** o **-f**, ma non entrambi.
-r	**MonitorQName**	La coda locale utilizzata esclusivamente dall'esempio.
-l	**ReportDir**	Il percorso di directory in cui memorizzare i messaggi di informazioni registrati in una serie di wrapping < fn> Per ogni gestore code e combinazione di code viene generato un file di report limitato a una determinata dimensione. Il programma di registrazione scrive sempre nello stesso file, ma conserva anche le versioni precedenti del file. < /fn> file di report.
-t		(Facoltativo) Abilita il trasferimento dei messaggi in coda dalle code locali inattive alle code attive. Se non abilitato, solo i nuovi messaggi che entrano nel cluster vengono instradati dinamicamente alle istanze attive di una coda.
-u	**ActiveVal**	(Facoltativo) Modifica automaticamente la proprietà **CLWLUSEQ** di una istanza della coda monitorata in ANY quando è inattivo e nella proprietà **ActiveVal** quando è attivo. **ActiveVal** può essere LOCAL o QMGR. Se questo argomento non è impostato in un sistema in cui le applicazioni di inserimento si connettono allo stesso gestore code o in cui è abilitato il trasferimento dei messaggi, le code monitorate devono avere un **CLWLUSEQ** valore ANYo QMGR con il gestore code con un valore ANY.
-i	**Interval**	(Facoltativo) L'intervallo di tempo, in secondi, con cui il monitoraggio controlla le code. Il valore predefinito è 300 secondi (5 minuti).
-d		(Facoltativo) Abilita ulteriori output di diagnostica. L'output di debug potrebbe essere utile quando si configura inizialmente il sistema o quando si utilizza il codice di esempio.
-s		(Facoltativo) Abilita l'output statistico minimo per intervallo.
-v		(Facoltativo) Registrare le informazioni del report in `standard out`, oltre ai file di report.

Esempi di elenco argomenti:

```
-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
```

File di elenco code di esempio:

```
Q1
QUEUE.*
ABC
ABD
```

Procedura

1. Avviare l'esempio di monitoraggio coda cluster. È possibile avviare l'esempio in uno dei seguenti modi:
   - Utilizzare un prompt dei comandi con le autorizzazioni utente appropriate.
   - Utilizzare il comando MQSC **START SERVICE** , se l'esempio è configurato come un servizio IBM WebSphere MQ .

   L'elenco degli argomenti è lo stesso in entrambi i casi.

   L'esempio non avvia il controllo delle code per 10 secondi dopo l'inizializzazione del programma. Questo ritardo consente alle applicazioni di connettersi prima alle code monitorate, evitando modifiche non necessarie allo stato attivo della coda.
2. Arrestare l'esempio di monitoraggio della coda cluster. L'esempio si arresta automaticamente quando il gestore code viene arrestato, arrestato, sospeso o se la connessione al gestore code è interrotta. Esistono diversi modi per arrestare l'esempio senza arrestare il gestore code:
   - Configurare la coda locale utilizzata esclusivamente dall'esempio per disabilitare la funzione Get.
   - Inviare un messaggio con un **CorrelId** di `"STOP CLUSTER MONITOR\0\0\0\0"`, alla coda locale utilizzata esclusivamente dall'esempio.
   - Terminare il processo di esempio. Ciò potrebbe causare la perdita di messaggi non persistenti trasferiti alle code attive. Potrebbe anche risultare nella coda locale utilizzata dall'esempio tenuta aperta per un numero di secondi dopo la terminazione. Questa situazione impedisce l'avvio immediato di una nuova istanza dell'esempio di monitoraggio della coda cluster.

   Se l'esempio è stato avviato come un servizio IBM WebSphere MQ , **STOP SERVICE** non ha alcun effetto. È possibile utilizzare uno dei metodi di terminazione descritti come meccanismo **STOP SERVICE** configurato nel gestore code.

Operazioni successive

Controllare lo stato dell'esempio.

Se la creazione report è abilitata, è possibile esaminare i file di report per lo stato. Utilizzare il seguente comando per esaminare il file di report più recente.

```
QMgrName.ClusterName.RPT01.LOG
```

Per esaminare i file di report più vecchi, utilizzare i comandi riportati di seguito.

```
QMgrName.ClusterName.RPT02.LOG
QMgrName.ClusterName.RPT03.LOG
```

I file di report raggiungono una dimensione massima di circa 1 MB. Quando il file RPT01 si riempie, viene creato un nuovo file RPT01 . Il vecchio file RPT01 viene ridenominato in RPT02. RPT02 viene ridenominato in RPT03. Il vecchio RPT03 viene eliminato.

L'esempio crea messaggi informativi nelle situazioni seguenti:

- All'avvio
- Al termine

- Quando contrassegna una coda **ACTIVE** o **INACTIVE**
- quando riaccoda i messaggi da una coda inattiva a un'istanza o istanze attive

L'esempio crea un messaggio di errore *CLMnnnnE* per segnalare un problema che richiede attenzione.

Ogni 30 minuti, il campione riporta il tempo di elaborazione medio per intervallo di polling e il tempo di elaborazione trascorso. Queste informazioni sono contenute nel messaggio CLM0045I.

Quando i messaggi statistici sono abilitati **-s**, l'esempio riporta le seguenti informazioni statistiche su ciascun controllo coda:

- Tempo impiegato per elaborare le code (in millisecondi)
- Numero di code controllate
- Numero di modifiche attive / inattive effettuate
- Numero di messaggi trasferiti

Queste informazioni sono riportate nel messaggio CLM0048I.

I file di report potrebbero crescere rapidamente in modalità di debug e impacchettarsi rapidamente. In questa situazione, il limite di dimensione di 1 MB per i singoli file potrebbe essere superato.

AMQSCLM: Risoluzione dei problemi

Le seguenti sezioni contengono informazioni sugli scenari che potrebbero essere rilevati durante l'utilizzo dell'esempio. Vengono fornite informazioni sulle potenziali spiegazioni per uno scenario e le opzioni su come risolverlo.

Scenario: AMQSCLM non è in fase di avvio

Spiegazione potenziale: sintassi non corretta.

Azione: controllare l'output dell'errore standard per la sintassi corretta

Spiegazione potenziale: il gestore code non è disponibile.

Azione: controllare il file di report per l'ID messaggio CLM0010E.

Spiegazione potenziale: impossibile aprire o creare file o file di report.

Azione: controllare l'output di errore standard per i messaggi di errore durante l'inizializzazione.

Scenario: AMQSCLM non sta modificando una coda in ATTIVO o INATTIVO

Spiegazione potenziale: la coda non si trova nell'elenco di code da monitorare

Azione: controllare i parametri **-q** e **-f** .

Spiegazione potenziale: la coda non è una coda locale nel cluster corretto.

Azione: verificare che la coda sia locale e nel cluster corretto.

Spiegazione potenziale: AMQSCLM non è in esecuzione per questo gestore code e cluster.

Azione : avviare AMQSCLM per il gestore code e il cluster pertinenti.

Spiegazione potenziale: la coda viene lasciata INATTIVO, **CLWLPRTY**= 0, perché non ha consumer. In alternativa, viene lasciato ATTIVO **CLWLPRTY**> =1, perché ha almeno 1 consumer.

Azione: verificare se le applicazioni che utilizzano sono collegate alla coda.

Spiegazione potenziale: il server dei comandi del gestore code non è in esecuzione.

Azione: verificare la presenza di errori nei file di report.

Scenario: i messaggi non vengono instradati intorno alle code INATTIVO

Spiegazione potenziale: i messaggi vengono inseriti direttamente nel gestore code che possiede la coda inattiva e il valore **CLWLUSEQ** della coda non è ANYe l'argomento **-u** non viene utilizzato per AMQSCLM.

Azione: controllare il valore **CLWLUSEQ** del gestore code pertinente oppure verificare che l'argomento **-u** sia utilizzato per AMQSCLM.

Spiegazione potenziale: non sono presenti code attive su alcun gestore code. I messaggi vengono bilanciati in modo uniforme tra tutte le code inattive fino a quando una coda diventa attiva.

Azione: controllare lo stato delle code su tutti i gestori code.

Spiegazione potenziale: i messaggi vengono inseriti in un gestore code differente nel cluster rispetto a quello proprietario della coda inattiva e il valore **CLWLPRTY** aggiornato 0 non viene propagato al gestore code dell'applicazione di inserimento.

Azione: verificare che i canali cluster tra il gestore code monitorato e il gestore code del repository completo siano in esecuzione. Verificare che i canali tra il gestore code di inserimento e il gestore code del repository completo siano in esecuzione. Controllare i log degli errori dei gestori code del repository completo, di inserimento e monitorati.

Spiegazione potenziale: le istanze della coda remota sono attive (CLWLPRTY=1), ma i messaggi non possono essere instradati a queste istanze della coda perché il canale mittente del cluster dal gestore code locale non è in esecuzione.

Azione: controllare lo stato dei canali mittenti del cluster dal gestore code locale al gestore code remoto, o ai gestori, con un'istanza attiva della coda.

Scenario: AMQSCLM non trasferisce messaggi da una coda inattiva

Spiegazione potenziale: il trasferimento del messaggio non è abilitato (**-t**).

Azione: accertarsi che il trasferimento del messaggio sia abilitato (**-t**).

Spiegazione potenziale: la coda non si trova nell'elenco di code da monitorare.

Azione: controllare i parametri **-q** e **-f** .

Spiegazione potenziale: AMQSCLM non è in esecuzione per questo o per altri gestori code nel cluster che possiedono istanze della stessa coda.

Azione: avviare AMQSCLM.

Spiegazione potenziale: la coda ha **CLWLUSEQ**=LOCAL o **CLWLUSEQ**=QMGRe l'argomento **-u** non è impostato.

Azione: impostare il parametro **-u** o modificare la coda o la configurazione del gestore code su ANY.

Spiegazione potenziale: non ci sono istanze attive della coda nel cluster.

Azione: verificare la presenza di istanze della coda con un valore **CLWLPRTY**pari o superiore a 1.

Spiegazione potenziale:le istanze della coda remota hanno consumer (**IPPROCS**> = 1) ma sono inattive su tali gestori code (**CLWLPRTY**= 0) perché AMQSCLM non sta monitorando tali istanze remote.

Azione: verificare che AMQSCLM sia in esecuzione su tali gestori code e / o che la coda si trovi nell'elenco di code da monitorare controllando i valori dei parametri **-q** e **-f** .

Spiegazione potenziale: le istanze della coda remota sono attive (**CLWLPRTY**= 1), ma vengono visualizzate come inattive sul gestore code locale (**CLWLPRTY**= 0). Questa situazione è dovuta al fatto che il valore **CLWLPRTY** aggiornato non viene propagato a questo gestore code.

Azione: verificare che i gestori code remoti siano connessi ad almeno uno dei gestori code del repository completo nel cluster. Assicurarsi che i gestori code del repository completo funzionino correttamente. Verificare che i canali tra i gestori code del repository completo e i gestori code monitorati siano in esecuzione.

Spiegazione potenziale: i messaggi non sono sottoposti a commit, quindi non sono richiamabili.

Azione: verificare il corretto funzionamento dell'applicazione mittente.

Spiegazione potenziale: AMQSCLM non ha accesso alla coda locale in cui sono accodati i messaggi.

Azione : questo scenario potrebbe essere dovuto al fatto che AMQSCLM non è in esecuzione come utente con autorizzazioni sufficienti per accedere alla coda.

Spiegazione potenziale: il server dei comandi del gestore code non è in esecuzione.

Azione: avviare il server dei comandi del gestore code.

Spiegazione potenziale: AMQSCLM ha rilevato un errore.

Azione: verificare la presenza di errori nei file di report.

Spiegazione potenziale: le istanze della coda remota sono attive (CLWLPRTY=1), ma i messaggi non possono essere trasferiti a tali istanze della coda perché il canale mittente del cluster dal gestore code locale non è in esecuzione. Questo è spesso accompagnato da un'avvertenza CLM0030W nel log del report amqsclm.

Azione: controllare lo stato dei canali mittenti del cluster dal gestore code locale al gestore code remoto, o ai gestori, con un'istanza attiva della coda.

Programma di esempio per Connection Endpoint Lookup (CEPL)

IBM WebSphere MQ L'esempio Ricerca endpoint di connessione fornisce un modulo di uscita semplice ma potente che offre agli utenti WebSphere MQ un metodo per richiamare le definizioni di connessione da un repository LDAP, ad esempio Tivoli Directory Server.

Tivoli Directory Server v6.3 Client deve essere installato per poter utilizzare CEPL.

Per utilizzare questo esempio è necessaria una conoscenza pratica della gestione di WebSphere MQ sulle piattaforme supportate.

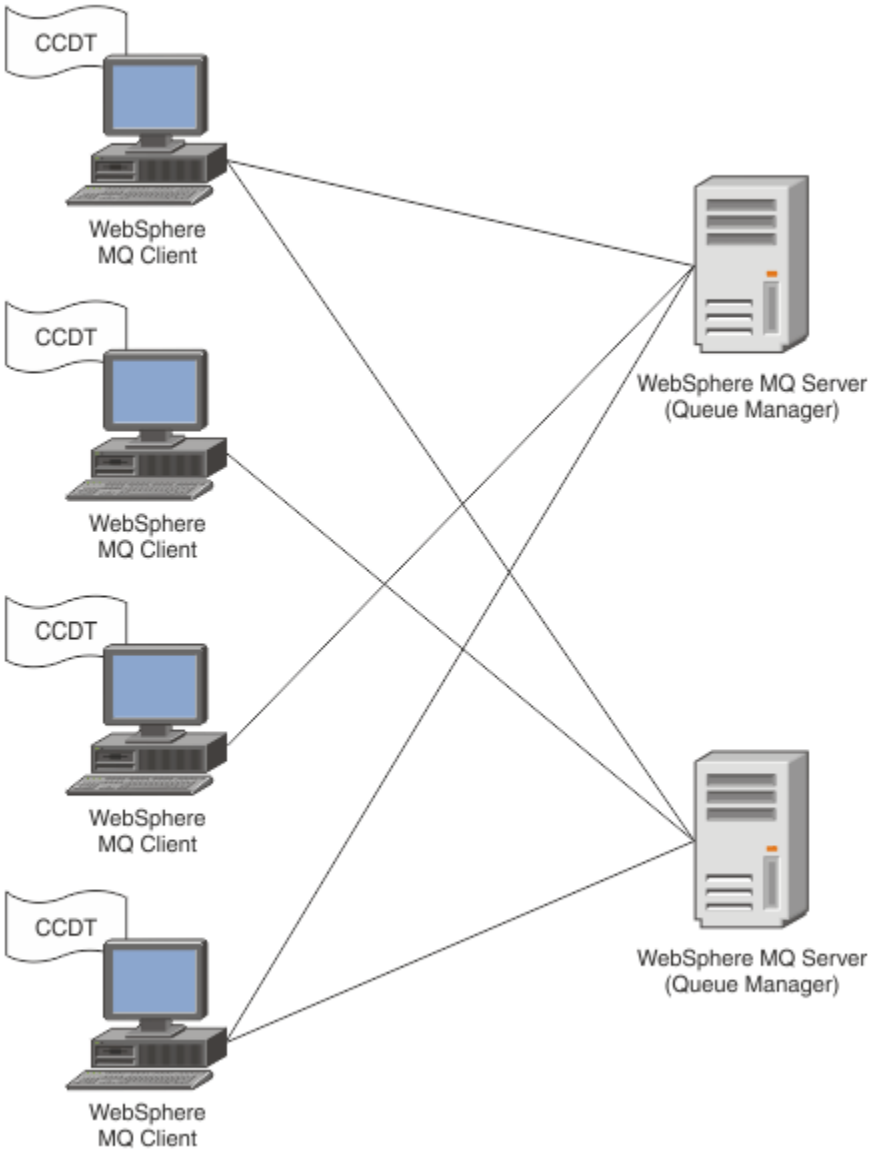
Introduzione

Configurare un repository globale, ad esempio una directory LDAP (Lightweight Directory Access Protocol), per memorizzare le definizioni di connessione client per facilitare la gestione e la manutenzione.

Utilizzo di un'applicazione client IBM WebSphere MQ per stabilire una connessione a un gestore code mediante CCDT (Client Connection Definition Table).

La CCDT viene creata tramite l'interfaccia di gestione MQSC standard WebSphere MQ .
L'utente deve essere connesso a un gestore code per creare definizioni di connessione client, anche se i dati contenuti nella definizione non sono limitati al gestore code. Il file

CCDT generato deve essere distribuito manualmente tra le applicazioni e le macchine client.

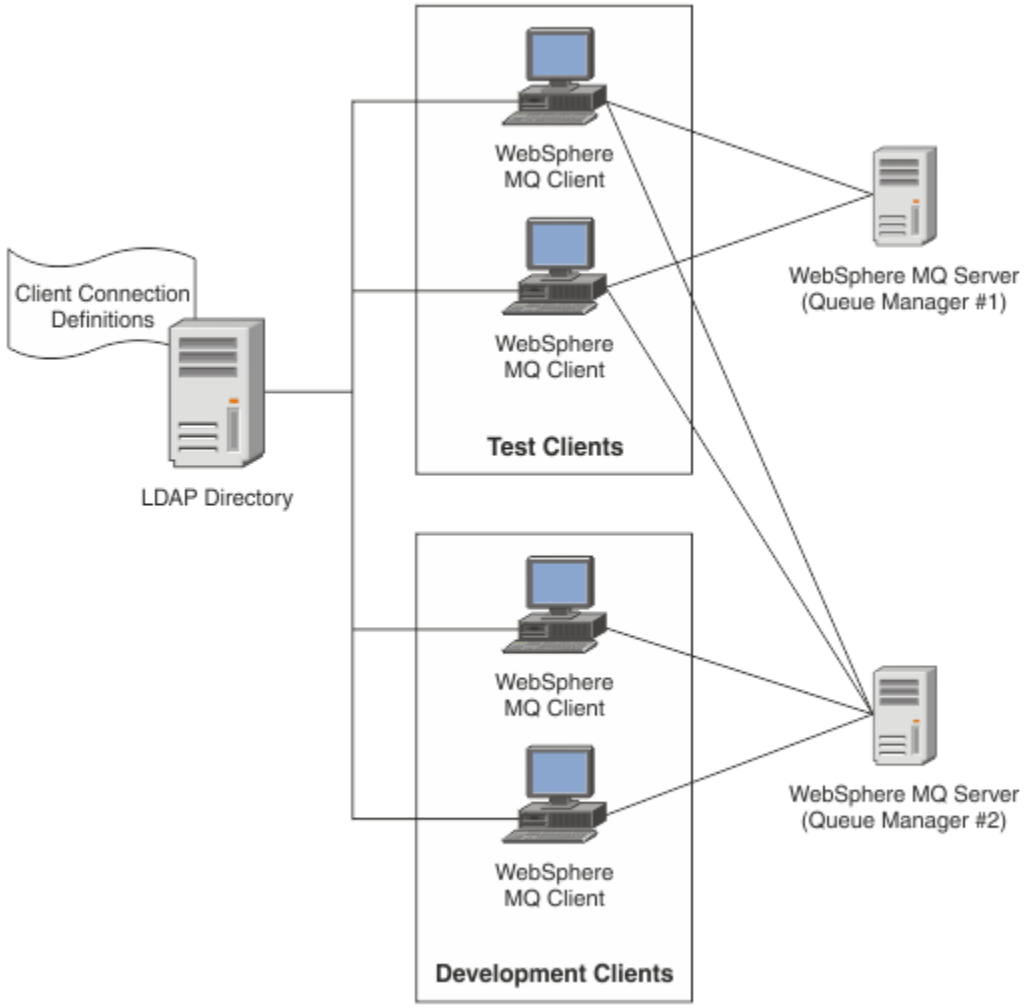


Il file CCDT deve essere distribuito a ciascun client WebSphere MQ . Dove migliaia di clienti possono esistere sia localmente che globalmente, diventerebbe presto difficile da mantenere e amministrare. Un approccio più flessibile è necessario per garantire che ciascun client disponga delle definizioni client corrette.

Un approccio di questo tipo è quello di memorizzare le definizioni di connessione client in un repository globale come una directory LDAP (Lightweight Directory Access Protocol). Una directory LDAP può anche fornire ulteriori funzionalità di sicurezza, indicizzazione e ricerca, consentendo a ogni client di accedere solo a quelle definizioni di connessione che li riguardano.

La directory LDAP può essere configurata in modo che solo definizioni specifiche siano disponibili per determinati gruppi di utenti. Ad esempio, i client di verifica possono accedere al gestore

code #1 e #2, mentre i client di sviluppo possono accedere solo al gestore code #2 .



Il modulo di uscita può ricercare un contenitore LDAP, ad esempio IBM Tivoli Directory Server, per richiamare le definizioni di canale. Utilizzando tali definizioni di connessione, un'applicazione client WebSphere MQ può stabilire una connessione a un gestore code.

Il modulo di uscita è un modulo di uscita di pre - connessione che consente di ottenere la definizione del canale durante la chiamata MQCONN/MQCONNX da un repository LDAP.

Il modulo di uscita e lo schema potrebbero essere implementati da:

- I clienti che hanno già creato una base di competenze utilizzando la tecnologia basata su file CCDT esistente e che desiderano semplificare i costi di gestione e distribuzione.
- Clienti esistenti che utilizzano già la propria tecnologia di proprietà per la distribuzione delle definizioni di connessione client.
- Clienti nuovi o esistenti che attualmente non utilizzano alcun tipo di soluzione di connessione client e che desiderano utilizzare le funzioni offerte da IBM WebSphere MQ.
- Clienti nuovi o esistenti che desiderano utilizzare direttamente o ottimizzare il proprio modello di messaggistica in linea con qualsiasi architettura di business LDAP corrente.

Ambienti supportati

Verificare di disporre di un sistema operativo supportato e del software pertinente prima di eseguire l'esempio Ricerca endpoint di connessione.

Il programma di esempio per IBM WebSphere MQ Connection Endpoint Lookup richiede il seguente software:

- IBM WebSphere MQ V7.0o successive
- Tivoli Directory Server V6.3 Client o versioni successive

Sistemi operativi supportati:

1. Windows (XP/2003/2008)
2. Solaris (SPARC e x86-64)
3. AIX
4. Linux
   - RHEL v4 e v5 su System p
   - SUSE v9 e v10 su System p
   - RHEL v4 e v5 System x32 bit e x64 bit
   - SUSE v9 e v10 System x32 bit e x64 bit
5. HP IA64.

Nota: Il programma di esempio non è disponibile per le piattaforme z/OS, i/5e HP PARISC.

Installazione e configurazione

Installazione e configurazione del modulo di uscita e schema dell'endpoint di connessione.

Installazione del modulo di uscita

Durante l'installazione di WebSphere MQ, il modulo di uscita viene installato in `tools/samples/c/preconnexit/bin`. Per le piattaforme a 32 bit, il modulo di uscita deve essere copiare in `exit/<install name>/` prima di poter essere utilizzato. Per piattaforme a 64 bit, il modulo di uscita deve essere copiato in exit64/<installation name> / prima che possa essere utilizzato.

Installazione dello schema dell'endpoint di connessione

L'uscita utilizza lo schema dell'endpoint di connessione, *ibm-amq.schema* . Il file di schema deve essere importato in qualsiasi server LDAP prima di poter utilizzare l'uscita. Dopo aver importato lo schema, è necessario aggiungere i valori per gli attributi.

Di seguito è riportato un esempio per l'importazione dello schema dell'endpoint di connessione. L'esempio presuppone che venga utilizzato IBM Tivoli Directory Server (ITDS).

- Assicurarsi che IBM Tivoli Directory Server sia in esecuzione, quindi copiare o inviare via FTP il file *ibm-amq.schema* al server ITDS.
- Sul server ITDS, immettere il seguente comando per installare lo schema nell'archivio ITDS, dove ID LDAP e password LDAP sono il DN root e la password per il server LDAP:

  ldapadd -D "ID LDAP" -w "password LDAP" -f ibm-amq.schema
- In una finestra di comandi, immettere il seguente comando o utilizzare uno strumento di terze parti per sfogliare lo schema per la verifica:

  ldapsearch objectclass=ibm -amqClientConnection

Fare riferimento alla documentazione del server LDAP per ulteriori dettagli sull'importazione del file di schema.

Configurazione

Una nuova sezione chiamata **PreConnect** deve essere aggiunta al file di configurazione client, ad esempio il file *mqclient.ini* . La sezione PreConnect contiene le seguenti parole chiave:

Modulo : il nome del modulo contenente il codice di uscita API. Se questo campo contiene il percorso completo del modulo, viene utilizzato come se fosse la cartella *exit* o *exit64* nell'installazione di WebSphere MQ .

Funzione : nome del punto di ingresso funzionale nella libreria che contiene il codice di uscita PreConnect . La definizione della funzione aderisce al prototipo MQ_PRECONNECT_EXIT.

Dati : URI del repository LDAP contenente le definizioni di canale.

Il seguente frammento è un esempio delle modifiche richieste per il file *mqclient.ini* .

```
PreConnect:
  Module=amqlcelp
  Function=PreConnectExit
  Data=ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
  Sequence=1
```

Panoramica di exit e schema

Sintassi e parametri utilizzati per stabilire una connessione a un gestore code.

WebSphere MQ v7.5 definisce la sintassi seguente per un punto di entrata in un modulo di uscita.

```
void MQENTRY MQ_PRECONNECT_EXIT ( PMQNXP  pExitParms
                                , PMQCHAR pQMgrName
                                , PPMQCNO ppConnectOpts
                                , PMQLONG pCompCode
                                , PMQLONG pReason)
```

Durante l'esecuzione della chiamata MQCONN/X, WebSphere MQ C Client carica il modulo di uscita contenente un'implementazione della sintassi della funzione. Richiama quindi una funzione di uscita per richiamare definizioni di canale. Le definizioni di canale richiamati vengono quindi utilizzate per stabilire la connessione a un gestore code.

Parametri

pExitParms

Tipo: input/output PMQNXP

La struttura del parametro di uscita PreConnection . La struttura è assegnata e gestita dal chiamante dell'uscita.

```
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 */
};
```

pQMgrNome

Tipo: input/output PMQCHAR

Il nome del gestore code. In fase di input, questo parametro è la stringa del filtro fornita alla chiamata API MQCONN tramite il parametro **QMgrName** . Questo campo potrebbe essere vuoto, esplicito o contenere determinati caratteri jolly. Il campo viene modificato dall'uscita. Il parametro è NULL quando l'uscita viene chiamata con MQXR_TERM.

ppConnectOpzioni

Tipo: ppConnectOpts input/output

Opzioni che controllano l'azione di MQCONNX. Questo è un puntatore a una struttura di opzioni di connessione MQCNO che controlla l'azione della chiamata API MQCONN. Il parametro è NULL quando l'uscita viene chiamata con MQXR_TERM. Il client MQI fornisce sempre una struttura MQCNO all'uscita, anche se non è stata originariamente fornita dall'applicazione. Se un'applicazione fornisce

una struttura MQCNO, il client effettua un duplicato per inoltrarlo all'uscita in cui viene modificato. Il client conserva la proprietà di MQCNO. Un MQCD a cui si fa riferimento tramite MQCNO ha la precedenza su qualsiasi definizione di connessione fornita tramite l'array. Il client utilizza la struttura MQCNO per connettersi al gestore code e gli altri vengono ignorati.

Codice pComp

Tipo: input / output PMQLONG

Codice di completamento. Puntatore a un MQLONG che riceve il codice di completamento delle uscite. Deve essere uno dei seguenti valori:

MQCC_OK - Completamento riuscito

MQCC_WARNING - Avvertenza (completamento parziale)

MQCC_FAILED - Chiamata non riuscita

pReason

Tipo: input / output PMQLONG

Codice di qualificazione motivo pComp. Puntatore ad un MQLONG che riceve il codice motivo di uscita. Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE - (0, x '000') Nessun motivo per la notifica.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Informazioni sul contesto LDAP MQ

L'uscita utilizza la seguente struttura di dati per le informazioni di contesto.

MQNLDAPCTX

La struttura MQNLDAPCTX ha il seguente prototipo 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 */
};
```

Codice di esempio per la creazione dell'uscita di ricerca endpoint di connessione

Frammenti di codice per la compilazione dell'origine su Windows e piattaforme distribuite.

Compilazione dell'origine

È possibile compilare l'origine con qualsiasi libreria client LDAP, ad esempio, IBM Tivoli Directory Server 6.3 Librerie client. Questa documentazione presume che si stiano utilizzando le librerie client di Tivoli Directory Server 6.3 .

Nota: La libreria di uscita di pre - connessione è stata verificata con i seguenti server LDAP:

- IBM Tivoli Directory Server V6.3
- Novell eDirectory V8.2

I seguenti frammenti di codice descrivono come compilare le uscite su Windowse su altre piattaforme distribuite:

Compilazione dell'uscita sulla piattaforma Windows

È possibile utilizzare il seguente frammento per compilare l'origine di uscita su Windows:

```
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
```

Nota: È possibile che vengano visualizzate delle avvertenze durante la compilazione delle librerie client IBM Tivoli Directory Server 6.3 con il compilatore Microsoft Visual Studio 2005 o superiore, se si utilizzano le librerie client IBM Tivoli Directory Server 6.3 compilate con il compilatore Microsoft Visual Studio 2003.

Compilazione dell'uscita su altre piattaforme distribuite

È possibile utilizzare il seguente frammento per compilare l'origine di uscita su altre piattaforme distribuite, ad esempio, Linux. Alcune opzioni del compilatore potrebbero differire su altre piattaforme distribuite.

```
#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
```

IBM Tivoli Directory Server fornisce librerie di collegamento statiche e dinamiche, ma è possibile utilizzare solo una forma di librerie. Questo script presuppone che si stiano utilizzando le librerie statiche.

```
#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)
```

Richiamo modulo di uscita

il modulo di uscita PreConnect può essere richiamato con tre diversi codici di errore. Questa sezione descrive ogni motivo di uscita in modo più approfondito.

INIT_MQXR

L'uscita viene richiamata con il codice motivo MQXR_INIT per inizializzare e stabilire la connessione a un server LDAP.

Prima della chiamata *MQXR_INIT* , il campo *pExitDataPtr* della struttura MQNXP sarebbe stato popolato con l'attributo Data dalla stanza PreConnect all'interno del file *mqclient.ini* (ad esempio, LDAP).

Un URL LDAP è costituito almeno dal protocollo, nome host, numero di porta e DN di base per la ricerca. L'uscita analizza l'URL LDAP contenuto nel campo *pExitDataPtr* , assegna una struttura del contesto di ricerca LDAP MQNLDAPCTX e la popola di conseguenza. L'indirizzo di tale struttura viene memorizzato nel campo *pExitUserAreaPtr* . Se non si analizza correttamente l'URL LDAP, si verifica l'errore MQCC_FAILED.

A questo punto, l'uscita si connette e si collega al server LDAP utilizzando i parametri MQNLDAPCTX. Anche gli handle API LDAP risultanti vengono memorizzati all'interno di questa struttura.

MQXR_PRECONNECT
Il modulo di exit viene richiamato con il codice di errore MQXR_PRECONNECT per richiamare le definizioni di canale da un server LDAP.

L'uscita ricerca nel server LDAP le definizioni di canale corrispondenti al filtro fornito. Se il parametro *QMgrName* contiene un nome gestore code specifico, la ricerca restituisce tutte le definizioni di canale il cui valore dell'attributo LDAP *ibm -amqQueueManagerName* corrisponde al nome gestore code fornito.

Se il parametro *QMgrName* è '*' o ' ' (vuoto), la ricerca restituisce tutte le definizioni di canale il cui attributo endpoint *ibm -amqIsClientDefault* Connection è impostato su true.

Dopo una ricerca riuscita, l'exit prepara una o più definizioni MQCD e ritorna al chiamante.

MQXR_TERM
L'uscita viene chiamata con questo codice di errore quando l'uscita deve essere ripulita. Durante questa operazione, l'uscita si disconnette dal server LDAP, rilascia tutta la memoria assegnata e gestita dall'uscita. Ciò includerà la struttura *MQNLDAPCTX* , l'array del puntatore e ogni MQCD a cui fa riferimento. Tutti gli altri campi sono impostati sui valori predefiniti. I parametri di uscita *pQMgrName* e *ppConnectOpts* non sono utilizzati durante *MQXR_TERM* e possono essere NULL.

Schemi LDAP
I dati di connessione client vengono memorizzati in un repository globale denominato LDAP (Lightweight Directory Access Protocol). Un client WebSphere MQ utilizza una directory LDAP per ottenere le definizioni di connessione. La struttura delle definizioni di connessione client WebSphere MQ all'interno della directory LDAP è nota come schema LDAP. Uno schema LDAP è la raccolta di definizioni del tipo di attributo, definizioni della classe di oggetti e altre informazioni che un server utilizza per determinare se un filtro o un'asserzione del valore di attributo corrisponde agli attributi di una voce e se consentire, aggiungere e modificare le operazioni.

Memorizzazione dei dati nella directory LDAP

Le definizioni di connessione client si trovano in un ramo specifico all'interno della struttura di directory nota come punto di connessione. Come tutti gli altri nodi all'interno di una directory LDAP, al punto di connessione è associato un DN (Distinguished Name). È possibile utilizzare questo nodo come punto di partenza per qualsiasi query eseguita sulla directory. Utilizzare il filtro quando si interroga la directory LDAP per restituire un sottoinsieme di definizioni di connessione client. È possibile limitare l'accesso alle strutture ad albero secondarie in base alle autorizzazioni concesse in altre parti della struttura ad albero di directory, ad esempio, a utenti, dipartimenti o gruppi.

Definizione di attributi e classi personali
Memorizzare la definizione di canale client modificando lo schema LDAP. Tutte le definizioni di dati LDAP richiedono oggetti e attributi. Gli oggetti e gli attributi sono identificati da un numero OID (object identifier) che identifica in modo univoco l'oggetto o l'attributo. Tutte le classi all'interno di uno schema LDAP ereditano direttamente o indirettamente dall'oggetto superiore. L'oggetto di definizione del canale client contiene gli attributi dell'oggetto superiore. Tutte le definizioni di dati LDAP richiedono oggetti e attributi:

- Le definizioni oggetto sono raccolte di attributi LDAP.
- Gli attributi sono tipi di dati LDAP.

La descrizione di ciascun attributo e il modo in cui vengono associati alle normali proprietà WebSphere MQ sono descritti in Attributi LDAP .

Attributi LDAP

Gli attributi LDAP definiti sono specifici per WebSphere MQ e vengono associati direttamente alle proprietà di collegamento client.

WebSphere MQ Attributi stringa directory canale client

Gli attributi della stringa di caratteri con la relativa associazione alle proprietà WebSphere MQ sono riportati nella seguente tabella. Gli attributi possono contenere i valori della sintassi directoryString (Unicode con codificaUTF-8 , ossia un sistema di codifica a byte variabile che include IA5/ASCII come sottoinsieme). La sintassi è specificata dal relativo OID (object identification number).

Tabella 21. WebSphere MQ Attributi stringa della directory del canale client

Attributo LDAP	Descrizione	Proprietà WebSphere MQ
CN	Il nome comune composto dal nome del canale e dal nome del gestore code di definizione.	
ibm -amqChannelamqChannel	Il nome della definizione del canale.	CHANNEL
ibm -amqConnectionamqConnection	L'identificativo della connessione di comunicazione.	CONNAME
ibm -amqDescription	La descrizione del canale.	DESCR
ibm -amqLocalIndirizzo	L'indirizzo di comunicazione locale del canale.	LOCLADDR
ibm -amqModeNome	a bThe LU 6.2 .	MODENAME
ibm -amqPassword	La password che può essere utilizzata.	Password
ibm -amqQueueManagerName	Il nome del gestore code o del gruppo di gestori code a cui un'applicazione client WebSphere MQ può richiedere la connessione.	QMNAME
ibm -amqSecurityExitUserDati	I dati utente passati all'uscita di sicurezza.	SCYDATA
ibm -amqSecurityExitName	Il nome del programma di uscita che deve essere eseguito dall'uscita di sicurezza del canale.	SCYEXIT
ibm -amqSslCipherSpec	Un singolo CipherSpec per una connessione SSL.	SSLCIPH
ibm -amqSslPeerName	Verifica il DN (Distinguished Name) del certificato dal gestore code peer o dal client all'altra estremità di un canale WebSphere MQ .	SSLPEER
ibm -amqTransactionProgramName	Il nome del programma di transazione.	TPNAME
ibm - IDamqUser	L'ID utente che l'MCA deve utilizzare quando tenta di avviare una sessione SNA sicura con un MCA remoto.	USERID

WebSphere MQ

Gli attributi con valori predefiniti (ad esempio, un tipo enumerato) vengono memorizzati come numeri interi standard. Questi valori vengono memorizzati nella directory LDAP come valori interi e non utilizzando il nome costante associato.

Tabella 22. WebSphere MQ Attributi interi directory canale client		
Attributo LDAP	**Descrizione**	**Proprietà WebSphere MQ**
ibm -amqConnectionAffinità	Determina se le applicazioni client, che si connettono più volte tramite lo stesso nome gestore code, utilizzano lo stesso canale client.	AFFINITÀ
ibm -amqClientChannelWeight	Una ponderazione per influenzare quale definizione di canale di connessione client viene utilizzata.	CLNTWGHT
ibm -amqHeartBeatInterval	Il tempo approssimativo tra i flussi di heartbeat che devono essere inviati da un MCA di invio quando non sono presenti messaggi nella coda di trasmissione.	HBINT
ibm -amqKeepAliveInterval	Un valore di timeout per un canale.	KAINT
ibm -amqMaximumMessageLength	La lunghezza massima di un messaggio che è possibile trasmettere sul canale.	MAXMSGL
ibm -amqSharingConversazioni	Il numero massimo di conversazioni che condividono ciascuna istanza del canale TCP/IP.	SHARECNV
ibm -amqTransportamqTransport	Il tipo di trasporto da utilizzare.	TRPTYPE

WebSphere MQ attributo booleano canale client

Questo attributo booleano non è associato ad alcuna proprietà WebSphere MQ . La sintassi di questo attributo indica un valore booleano.

Tabella 23. WebSphere MQ attributo booleano canale client	
Attributo LDAP	**Descrizione**
ibm -amqIsClientDefault	Questo attributo booleano viene definito per risolvere il problema di ricerca delle voci il cui attributo ibm -amqQueueManagerName non è stato definito.

WebSphere MQ

Le proprietà WebSphere MQ vengono memorizzate come attributo di elenco separato da virgole a valore singolo all'interno della directory LDAP. Gli attributi vengono definiti allo stesso modo degli altri attributi della stringa della directory. Gli attributi dell'elenco e la relativa associazione alle proprietà WebSphere MQ sono descritti nella seguente tabella.

Tabella 24. WebSphere MQ		
Attributo LDAP	**Descrizione**	**Proprietà WebSphere MQ**
ibm -amqHeaderCompressione	Un elenco delle tecniche di compressione dei dati di intestazione supportate dal canale.	COMPHDR
ibm -amqMessageCompressione	Un elenco di tecniche di compressione dei dati dei messaggi supportate dal canale.	COMPMSG
ibm -amqSenddatiExitUser	I dati utente passati all'uscita di invio.	SENDDATA
ibm -amqSendNomeExitUser	Il nome del programma di uscita che deve essere eseguito dall'uscita di invio del canale.	SENDEXIT
ibm -amqReceivedatiExitUser	I dati utente passati all'uscita di ricezione.	RCVDATA

Tabella 24. WebSphere MQ (Continua)		
Attributo LDAP	**Descrizione**	**Proprietà WebSphere MQ**
ibm -amqReceiveExitName	Il nome del programma di uscita utente che deve essere eseguito dall'uscita utente di ricezione del canale.	RCVEXIT

Nome comune (Common Name)
Il CN (common name) è costituito dal nome del canale e dal nome del gestore code di definizione.

È un attributo preesistente.

Il formato del CN è:

```
CN=CHANNEL_NAME(DEFINING_Q_MGR_NAME)
```

Ad esempio:

```
CN=TC1(QM_T1)
```

È possibile specificare solo un valore per questo attributo.

Questo attributo è un attributo stringa e i valori non sono sensibili al maiuscolo / minuscolo. La corrispondenza della sottostringa viene ignorata. La corrispondenza della sottostringa è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca, utilizzando una sottostringa (ad esempio, CN=jim * dove CN è un attributo) e contiene uno o più caratteri jolly.

ibm - NomeamqChannel
Questo attributo specifica il nome della definizione di canale.

Questo attributo ha un singolo valore di stringa con un massimo di 20 caratteri non sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza sottostringa è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca, utilizzando una sottostringa e contiene uno o più caratteri jolly.

ibm -amqDescription
Questo attributo LDAP fornisce la descrizione del canale.

Questo attributo ha un singolo valore stringa con un massimo di 64 byte, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

ibm -amqConnectionNome
Questo attributo LDAP è l'identificativo della connessione di comunicazione. Specifica i particolari collegamenti di comunicazione che devono essere utilizzati da questo canale.

Questo attributo ha un singolo valore stringa con un massimo di 264 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

ibm - IndirizzoamqLocal
Questo attributo specifica l'indirizzo di comunicazione locale per il canale.

Questo attributo ha un singolo valore stringa con un massimo di 48 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

ibm -amqModeNome
Questo attributo è per l'utilizzo con connessioni LU 6.2. Dà una definizione extra per le caratteristiche di sessione della connessione quando viene eseguita un'assegnazione di sessione di comunicazione.

Questo attributo ha un singolo valore di stringa di esattamente 8 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

ibm -amqPassword
Questo attributo LDAP specifica una password che può essere utilizzata da MCA quando si tenta di avviare una sessione LU sicura 6.2 con un MCA remoto.

Questo attributo ha un singolo valore intero con un massimo di 12 cifre. Non è un attributo preesistente.

ibm -amqQueueManagerName
Questo attributo specifica il nome del gestore code o del gruppo di gestore code a cui un'applicazione client WebSphere MQ può richiedere la connessione.

Questo attributo ha un singolo valore stringa con un massimo di 48 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

Dati ibm -amqSecurityExitUser
Questo attributo LDAP specifica i dati utente passati all'uscita di sicurezza.

Questo attributo ha un singolo valore stringa con un massimo di 999 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

ibm -amqSecurityExitName
Questo attributo LDAP specifica il nome del programma di uscita che deve essere eseguito dall'uscita di sicurezza del canale.

Lasciare vuoto se non è attiva alcuna uscita di sicurezza del canale.

Questo attributo ha un singolo valore stringa con un massimo di 999 caratteri, che non sono sensibili al maiuscolo / minuscolo. Questo attributo non è pre - esistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

ibm -amqSslCipherSpec
Questo attributo LDAP specifica una singola CipherSpec per una connessione SSL.

Questo attributo ha un singolo valore stringa con un massimo di 32 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

ibm -amqSslPeerName
Questo attributo LDAP viene utilizzato per controllare il DN (Distinguished Name) del certificato dal gestore code peer o dal client all'altra estremità di un canale WebSphere MQ .

Questo attributo LDAP ha un singolo valore stringa con un massimo di 1024 byte, che non sono sensibili al maiuscolo / minuscolo. Non è una preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

ibm -amqTransactionProgramName
Questo attributo LDAP specifica il nome del programma di transazione. Viene utilizzato con le connessioni LU 6.2 .

Questo attributo ha un singolo valore stringa con un massimo di 64 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è una preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

ibm -amqUserID
Questo attributo LDAP specifica l'ID utente che deve essere utilizzato da MCA quando si tenta di avviare una sessione SNA sicura con un MCA remoto.

Questo attributo ha un singolo valore di stringa di esattamente 12 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

Affinità ibm -amqConnection
Questo attributo LDAP specifica se le applicazioni client, che si collegano più volte utilizzando lo stesso nome gestore code, utilizzano lo stesso canale client.

Questo attributo ha un singolo valore intero. Non è un attributo preesistente.

ibm -amqClientChannelWeight
Questo attributo LDAP specifica una ponderazione che influenza quale definizione di canale di connessione client viene utilizzata.

L'attributo di ponderazione del canale client viene utilizzato per modificare la selezione delle definizioni del canale client quando è disponibile più di una definizione adatta.

Questo attributo ha un singolo valore intero. Non è un attributo preesistente.

ibm -amqHeartBeatInterval
Questo attributo LDAP specifica il tempo approssimativo tra i flussi heartbeat che devono essere passati da un MCA di invio quando non ci sono messaggi nella coda di trasmissione.

Questo attributo ha un singolo valore intero. Non è un attributo preesistente. Il valore predefinito è 1. Il valore predefinito è impostato nell'operazione della variabile di ambiente MQSERVER corrente.

ibm -amqKeepAliveInterval
Questo attributo LDAP viene utilizzato per specificare un valore di timeout per un canale.

Il valore di questo attributo viene passato allo stack di comunicazioni specificando il tempo keepalive per il canale. È possibile utilizzarlo per specificare un valore keepalive diverso per ogni canale.

Questo attributo ha un singolo valore intero. Non è un attributo preesistente.

ibm -amqMaximumMessageLength
Questo attributo LDAP specifica la lunghezza massima di un messaggio che può essere trasmesso sul canale.

Il valore predefinito di questo attributo è 104857600 come da operazione della variabile di ambiente MQSERVER corrente. Questo attributo ha un singolo valore intero e non è un attributo preesistente.

ibm - ConversazioniamqSharing
Questo attributo LDAP specifica il numero massimo di conversazioni che condividono ciascuna istanza del canale TCP/IP.

Questo attributo ha un singolo valore intero. Questo attributo non è un attributo preesistente.

ibm - TipoamqTransport
Questo attributo LDAP specifica il tipo di trasporto da utilizzare.

Questo attributo ha un singolo valore intero. Non è un attributo preesistente.

ibm -amqIsClientDefault
Questo attributo booleano risolve il problema di ricerca delle voci in cui l'attributo ibm -amqQueueManagerName non è stato definito.

I moduli di uscita di preconnessione generalmente ricercano i server LDAP con il valore dell'attributo ibm -amqQueueManagerName come criterio di ricerca. Tale query restituisce tutte le voci in cui il valore dell'attributo ibm -amqQueueManagerName corrisponde al nome del gestore code specificato nella chiamata MQCONN/X. Tuttavia, quando si utilizzano le tabelle di definizione del canale client (CCDT), è possibile impostare il nome del gestore code su una chiamata MQCONN/X come vuoto oppure anteporre al nome un asterisco (*). Se il nome del gestore code è vuoto, il client si connette al gestore code predefinito. Se il nome è preceduto da un asterisco (*) per il gestore code, il client connette qualsiasi gestore code.

Allo stesso modo, l'attributo ibm -amqQueueManagerName in una voce può essere lasciato indefinito. In questo caso, si prevede che il client che utilizza queste informazioni sull'endpoint possa connettersi a qualsiasi gestore code. Ad esempio, una voce contiene le seguenti righe:

```
ibm-amqChannelName = "CHANNEL1"
ibm-amqConnectionName = myhost(1414)
```

In questo esempio, il client tenta di connettersi al gestore code specificato in esecuzione su `myhost`.

Tuttavia, nei server LDAP, non viene eseguita una ricerca su un valore di attributo che non è stato definito. Ad esempio, se una voce contiene le informazioni di connessione tranne ibm -amqQueueManagerName, i risultati della ricerca non includeranno questa voce. Per risolvere questo problema, puoi impostare ibm -amqIsClientDefault. Si tratta di un attributo booleano e si presume che abbia un valore FALSE se non definito.

Per le voci in cui ibm -amqQueueManagerName non è stato definito e si prevede che facciano parte della ricerca, impostare ibm -amqIsClientDefault su TRUE. Quando viene specificato uno spazio vuoto o un asterisco (*) come nome del gestore code in una chiamata a MQCONN/X, l'uscita di preconnessione ricerca nel server LDAP tutte le voci in cui il valore di attributo ibm -amqIsClientDefault è impostato su TRUE.

Nota: Non impostare o definire l'attributo ibm -amqQueueManagerName se ibm -amqIsClientDefault è impostato su TRUE.

Compressione ibm -amqHeader
Questo attributo LDAP è un elenco di tecniche di compressione dei dati di intestazione supportate dal canale.

La dimensione massima di questo attributo è 48 caratteri. Non è un attributo preesistente.

È possibile specificare solo un valore per questo attributo.

Questo attributo elenco viene specificato come stringhe di directory utilizzando un formato separato da virgole. Ad esempio, i valori specificati per **ibm-amqHeaderCompression** sono `0` che è associato a `NONE`. Tutti i valori che superano il limite massimo consentito verranno ignorati dal client. Ad esempio, ibm -amqHeaderCompression contiene un massimo di 2 numeri interi nell'elenco.

ibm - CompressioneamqMessage
Questo attributo LDAP è un elenco di tecniche di compressione dei dati dei messaggi supportate dal canale.

La dimensione massima di questo attributo è 48 caratteri. Non è un attributo preesistente.

Questo attributo non supporta più valori.

Questo attributo elenco viene specificato come stringhe di directory utilizzando un formato separato da virgole. Ad esempio, il valore specificato per questo attributo è 1,2,4, che si associa alla sequenza di compressione sottostante RLE, ZLIBFAST e ZLIBHIGH.

Tutti i valori che superano il limite massimo consentito vengono ignorati dal client. Ad esempio, la compressione ibm -amqMessagecontiene un massimo di 16 numeri interi nell'elenco.

ibm - DatiExitUseramqSend
Questo attributo LDAP specifica i dati utente passati all'uscita di invio.

Questo attributo LDAP ha un singolo valore di stringa con un massimo di 999 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

Nota: ibm-amqSendExitName e **ibm-amqSendExitUserData** devono essere sincronizzati in coppie. I dati utente devono essere sincronizzati con il nome di uscita. Quindi, se uno è specificato, anche l'altro deve essere specificato simmetricamente, anche se non contiene dati.

ibm -amqSendExitName
Questo attributo LDAP specifica il nome del programma exit che deve essere eseguito dall'exit di invio del canale.

Questo attributo ha un singolo valore stringa con un massimo di 999 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

Nota: ibm-amqSendExitName e **ibm-amqSendExitUserData** devono essere sincronizzati in coppie. I dati utente devono essere sincronizzati con il nome uscita. Quindi, se uno è specificato, anche l'altro deve essere specificato simmetricamente anche se non contiene dati.

ibm - DatiExitUseramqReceive
Questo attributo LDAP specifica i dati utente passati all'uscita di ricezione.

È possibile eseguire una sequenza di uscite di ricezione. La stringa di dati utente per una serie di uscite è separata da una virgola, spazi o entrambi.

Questo attributo ha un singolo valore stringa con un massimo di 999 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

Nota: ibm-amqReceiveExitName e **ibm-amqReceiveExitUserData** devono essere sincronizzati in coppie. I dati utente devono essere sincronizzati con il nome uscita. Quindi, se uno è specificato, anche l'altro deve essere specificato simmetricamente anche se non contiene dati.

ibm -amqReceiveExitName
Questo attributo LDAP specifica il nome del programma di uscita utente che deve essere eseguito dall'uscita utente di ricezione del canale.

Questo attributo è un elenco di nomi di programmi da eseguire in successione. Lasciare vuoto, se non è attiva alcuna uscita utente di ricezione del canale.

Questo attributo ha un singolo valore stringa con un massimo di 999 caratteri, che non sono sensibili al maiuscolo / minuscolo. Non è un attributo preesistente.

La corrispondenza della sottostringa viene ignorata. La corrispondenza della stringa secondaria è una regola di corrispondenza utilizzata nello schema secondario che specifica il comportamento dell'attributo in un filtro di ricerca.

Nota: **ibm-amqReceiveExitName** e **ibm-amqReceiveExitUserData** devono essere sincronizzati in coppie. I dati utente devono essere sincronizzati con il nome uscita. Quindi, se uno è specificato, anche l'altro deve essere specificato simmetricamente, anche se non contiene dati.

Scrittura di un'applicazione di accodamento

Utilizzare queste informazioni per informazioni sulla scrittura delle applicazioni di accodamento, la connessione e la disconnessione da un gestore code, la pubblicazione / sottoscrizione e l'apertura e la chiusura di oggetti.

Utilizzare i seguenti collegamenti per ulteriori informazioni sulla scrittura di applicazioni:

Concetti correlati

"Concetti dello sviluppo di applicazioni" a pagina 8
È possibile utilizzare una scelta di linguaggi procedurali o orientati agli oggetti per scrivere applicazioni IBM WebSphere MQ . Utilizzare i link in questo argomento per informazioni sui concetti di IBM WebSphere MQ che sono utili per sviluppatori di applicazioni.

"Scelta del linguaggio di programmazione da utilizzare" a pagina 79
Utilizza queste informazioni per scoprire i linguaggi di programmazione e i framework supportati da IBM WebSphere MQe alcune considerazioni per utilizzarli.

"Progettazione di applicazioni IBM WebSphere MQ" a pagina 90
Una volta stabilito il modo in cui le applicazioni possono sfruttare le piattaforme e gli ambienti disponibili, è necessario decidere come utilizzare le funzioni offerte da WebSphere MQ.

"Programmi di WebSphere MQ di esempio" a pagina 97
Utilizzare questa raccolta di argomenti per informazioni sui programmi WebSphere MQ di esempio su piattaforme differenti.

"Scrittura delle applicazioni client" a pagina 354
Informazioni necessarie per scrivere le applicazioni client su WebSphere MQ.

"Utilizzo dei servizi Web in WebSphere MQ" a pagina 948
È possibile sviluppare applicazioni IBM WebSphere MQ per servizi Web utilizzando il trasporto IBM WebSphere MQ per SOAP o il bridge IBM WebSphere MQ per HTTP.

"Creazione di un'applicazione IBM WebSphere MQ" a pagina 431
Utilizzare queste informazioni per informazioni sulla generazione di un'applicazione IBM WebSphere MQ su piattaforme differenti.

"Gestione degli errori del programma" a pagina 550
Queste informazioni spiegano gli errori associati alle chiamate MQI delle applicazioni quando effettua una chiamata o quando il suo messaggio viene consegnato alla destinazione finale.

Panoramica su Message Queue Interface

Informazioni sui componenti MQI (Message Queue Interface).

L'interfaccia della coda messaggi è composta da:

- *Chiamate* tramite cui i programmi possono accedere al gestore code e alle relative funzioni

- *Strutture* che i programmi utilizzano per trasmettere e richiamare i dati dal gestore code
- *Tipi di dati elementari* per la trasmissione e l'acquisizione di dati dal gestore code

WebSphere MQ per Windows e WebSphere MQ su sistemi UNIX and Linux fornisce anche:

- Chiamate tramite le quali WebSphere MQ per Windows e WebSphere MQ su programmi di sistemi UNIX and Linux può eseguire il commit e il backout delle modifiche.
- *Includi file* che definiscono i valori delle costanti fornite su queste piattaforme.
- *File di libreria* per collegare le applicazioni.
- Una suite di programmi di esempio che dimostrano come utilizzare MQI su queste piattaforme. Per ulteriori informazioni su questi esempi, consultare “Programmi di esempio per piattaforme distribuite” a pagina 97.
- Codice di origine ed eseguibile di esempio per i bind ai gestori transazioni esterni.

Utilizzare i seguenti link per ulteriori informazioni su MQI:

- “Chiamate MQI” a pagina 196
- “Chiamate punto di sincronizzazione” a pagina 197
- “Conversione dati, tipi di dati, definizioni di dati e strutture” a pagina 198
- “File di libreria e programmi stub IBM WebSphere MQ” a pagina 198
- “Parametri comuni a tutte le chiamate” a pagina 203
- “Specifica dei buffer” a pagina 203
- “Gestione del segnale UNIX and Linux” a pagina 204

Concetti correlati

“Connessione e disconnessione da un gestore code” a pagina 207
Per utilizzare i servizi di programmazione WebSphere MQ , un programma deve disporre di una connessione a un gestore code. Utilizzare queste informazioni per informazioni su come connettersi e disconnettersi da un gestore code.

“Apertura e chiusura di oggetti” a pagina 215
Queste informazioni forniscono informazioni sull'apertura e la chiusura degli oggetti WebSphere MQ .

“Inserimento di messaggi in una coda” a pagina 225
Utilizzare queste informazioni per informazioni su come inserire i messaggi su una coda.

“Richiamo dei messaggi da una coda” a pagina 241
Utilizzare queste informazioni per ottenere messaggi da una coda.

“Richiesta di informazioni e impostazione degli attributi dell'oggetto” a pagina 322
Gli attributi sono proprietà che definiscono le caratteristiche di un oggetto WebSphere MQ .

“Commit e backout delle unità di lavoro” a pagina 325
Queste informazioni descrivono come eseguire il commit e il backout di tutte le operazioni get e put recuperabili che si sono verificate in un'unità di lavoro.

“Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger” a pagina 331
Informazioni sui trigger e su come avviare le applicazioni IBM WebSphere MQ utilizzando i trigger.

“Utilizzo di MQI e cluster” a pagina 349
Esistono opzioni speciali sulle chiamate e sui codici di ritorno relativi al cluster.

Chiamate MQI

Utilizzare queste informazioni per informazioni sulle chiamate in MQI.

Le chiamate in MQI possono essere raggruppate come segue:

MQCONN, MQCONNX e MQDISC
Utilizzare queste chiamate per connettere un programma a (con o senza opzioni) e disconnettere un programma da un gestore code. Se si scrivono i programmi CICS per z/OS, non è necessario utilizzare

queste chiamate. Tuttavia, si consiglia di utilizzarli se si desidera trasferire l'applicazione su altre piattaforme.

MQOPEN e MQCLOSE
Utilizzare queste chiamate per aprire e chiudere un oggetto, ad esempio una coda.

MQPUT e MQPUT1
Utilizzare queste chiamate per inserire un messaggio in una coda.

MQGET
Utilizzare questa chiamata per esaminare i messaggi su una coda o per rimuovere i messaggi da una coda.

MQSUB, MQSUBRQ
Utilizzare queste chiamate per registrare una sottoscrizione ad un argomento e per richiedere le pubblicazioni corrispondenti alla sottoscrizione.

MQINQ
Utilizzare questa chiamata per esaminare gli attributi di un oggetto.

MQSET
Utilizzare questa chiamata per impostare alcuni attributi di una coda. Non è possibile impostare gli attributi di altri tipi di oggetto.

MQBEGIN, MQCMIT e MQBACK
Utilizzare queste chiamate quando il coordinatore di un'unità di lavoro è WebSphere MQ . MQBEGIN avvia l'unità di lavoro. MQCMIT e MQBACK terminano l'unità di lavoro, eseguendo il commit o il rollback degli aggiornamenti effettuati durante l'unità di lavoro. Vengono utilizzati i comandi nativi di avvio del controllo di commit, commit e rollback.

MQCRTMH, MQBUFMH, MQMHBUF, MQDLTMH
Utilizzare queste chiamate per creare un handle del messaggio, per convertire un handle del messaggio in un buffer o un buffer in un handle del messaggio e per eliminare un handle del messaggio.

MQSETMP, MQINQMP, MQDLTMP
Utilizzare queste chiamate per impostare una proprietà del messaggio su un handle del messaggio, analizzare una proprietà del messaggio ed eliminare una proprietà da un handle del messaggio.

MQCB, MQCB_FUNCTION, MQCTL
Utilizzare queste chiamate per registrare e controllare una funzione di callback.

MQSTAT
Utilizzare questa chiamata per richiamare le informazioni sullo stato delle precedenti operazioni di inserimento asincrone.

Consultare Descrizioni chiamata per una descrizione delle chiamate MQI.

Chiamate punto di sincronizzazione

Utilizzare queste informazioni per informazioni sulle chiamate sync point su diverse piattaforme.

Le chiamate del punto di sincronizzazione sono disponibili come segue:

Chiamate IBM WebSphere MQ su piattaforme Windows, UNIX e Linux

I seguenti prodotti forniscono le chiamate MQCMIT e MQBACK:

- IBM WebSphere MQ per Windows
- IBM WebSphere MQ su sistemi UNIX and Linux

Utilizzare le chiamate al punto di sincronizzazione nei programmi per indicare al gestore code che tutte le operazioni MQGET e MQPUT dall'ultimo punto di sincronizzazione devono essere rese permanenti (sottoposte a commit) o devono essere sottoposte a backout. Per eseguire il commit e il backout delle

modifiche nell'ambiente CICS , utilizzare comandi quali EXEC CICS SYNCPOINT e EXEC CICS SYNCPOINT ROLLBACK.

Conversione dati, tipi di dati, definizioni di dati e strutture

Utilizzare queste informazioni per informazioni sulle conversioni di dati, sui tipi di dati elementari, sulle definizioni di dati WebSphere MQ e sulle strutture quando si utilizza l'interfaccia coda messaggi.

Conversione dati

La chiamata MQXCNVC (convert characters) converte i dati del carattere del messaggio da una serie di caratteri a un altra. Ad eccezione di WebSphere MQ per z/OS, questa chiamata viene utilizzata solo da un'uscita di conversione dati.

Consultare MQXCNVC - Converti caratteri per la sintassi utilizzata con la chiamata MQXCNVC e “Scrittura delle uscite di conversione dati” a pagina 417 per istruzioni sulla scrittura e il richiamo delle uscite di conversione dati.

Tipi di dati elementari

Per i linguaggi di programmazione supportati, MQI fornisce tipi di dati elementari o campi non strutturati.

Questi tipi di dati sono descritti completamente in Tipi di dati elementari.

Definizioni di dati WebSphere MQ

I file di definizione dati forniti con WebSphere MQ contengono:

- Definizioni di tutte le costanti WebSphere MQ e codici di ritorno
- Definizioni delle strutture WebSphere MQ e tipi di dati
- Definizioni costanti per l'inizializzazione delle strutture
- Prototipi di funzione per ogni chiamata (solo per PL/I e il linguaggio C)

Per una descrizione completa dei file di definizione dei dati di WebSphere MQ , consultare “File di definizione dati IBM WebSphere MQ” a pagina 81.

Strutture

Le strutture, utilizzate con le chiamate MQI elencate in “Chiamate MQI” a pagina 196, vengono fornite nei file di definizione dati per ciascuno dei linguaggi di programmazione supportati.

Consultare Riepilogo tipi di dati della struttura per un riepilogo delle strutture.

File di libreria e programmi stub IBM WebSphere MQ

I programmi stub e i file di libreria forniti sono elencati qui, per ogni piattaforma.

Per ulteriori informazioni su come utilizzare i programmi stub e i file di libreria quando si crea un'applicazione eseguibile, consultare “Creazione di un'applicazione IBM WebSphere MQ” a pagina 431. Per informazioni sul collegamento ai file della libreria C++, consultare Utilizzo di C++ *WebSphere MQ Utilizzo di C+ +*.

IBM WebSphere MQ per Windows

Su IBM WebSphere MQ per Windows, è necessario collegare il programma ai file della libreria MQI forniti per l'ambiente in cui si sta eseguendo l'applicazione, oltre a quelli forniti dal sistema operativo:

Tabella 25. File di libreria per applicazioni Windows

File di libreria	**Ambiente**
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	Server per C (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	Client per C (32 bit)

Tabella 25. File di libreria per applicazioni Windows (Continua)	
File di libreria	**Ambiente**
MQ_INSTALLATION_PATH\Tools\Lib\mqmxa.lib	Interfaccia XA server per C (32-bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqcxa.lib	Interfaccia client XA per C (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	Client MTS per C (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcics4.lib	Supporto server TXSeries CICS per C (32-bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqccics4.lib	Supporto client TXSeries CICS per C (32-bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqmzf.lib	Uscite di servizi installabili per C (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb.lib	Server per IBM COBOL (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb.lib	Server per Micro Focus COBOL (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb.lib	Client per IBM COBOL (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb.lib	Client per Micro Focus COBOL (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\imqs23vn.lib	Server per C++ (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\imqc23vn.lib	Client per C+ + (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\imqb23vn.lib	Base per C++ (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\imqx23vn.lib	Client MTS per C+ + (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	Server per C (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	Client per C (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmxa.lib	Interfaccia XA server per C (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqcxa.lib	Interfaccia XA client per C (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	Client MTS per C (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb.lib	Server per IBM COBOL (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb.lib	Server per Micro Focus COBOL (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb.lib	Client per IBM COBOL (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb.lib	Client per Micro Focus COBOL (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\imqs23vn.lib	Server per C++ (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\imqc23vn.lib	Client per C++ (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\imqb23vn.lib	Base per C++ (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\imqx23vn.lib	Client MTS per C++ (64 bit)

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Utilizzare `amqmdnet.dll` per compilare programmi .NET. Per ulteriori informazioni, consultare “Compilazione dei programmi WebSphere MQ .NET” a pagina 596 nella sezione “Usando .Net” a pagina 562 .

Questi file vengono forniti per la compatibilit ... con le release precedenti:

mqic32.lib
mqic32xa.lib

IBM WebSphere MQ per AIX

In IBM WebSphere MQ per AIX, è necessario collegare il programma ai file della libreria MQI forniti per l'ambiente in cui si sta eseguendo l'applicazione, oltre a quelli forniti dal sistema operativo.

In un'applicazione non con thread:

Tabella 26. File di libreria per applicazioni AIX senza thread

File di libreria	**Ambiente**
libmqm.a	Server per C
libmqic.a & **libmqm.a**	Client per C
libmqmzf.a	Uscite di servizio installabili per C
libmqmxa.a	Interfaccia XA server
libmqmxa64.a	Interfaccia XA alternativa server
libmqcxa.a	Interfaccia XA client
libmqcxa64.a	Interfaccia XA alternativa del client
libmqmcbrt.o	Libreria di runtime WebSphere MQ per il supporto Micro Focus COBOL
libmqmcb.a	Server per COBOL
libmqicb.a	Client per COBOL
libimqc23ia.a	Client per C++
libimqs23ia.a	Server per C++

In un'applicazione con thread:

Tabella 27. File di libreria per applicazioni AIX con thread

File di libreria	**Ambiente**
libmqm_r.a	Server per C
libmqic_r.a & **libmqm_r.a**	Client per C
libmqmzf_r.a	Uscite di servizio installabili per C
libmqmxa_r.a	Interfaccia XA server
libmqmxa64_r.a	Interfaccia XA alternativa server
libmqcxa_r.a	Interfaccia XA client
libmqcxa64_r.a	Interfaccia XA alternativa del client
libimqc23ia_r.a	Client per C++
libimqs23ia_r.a	Server per C++

IBM WebSphere MQ per HP-UX

Su IBM WebSphere MQ per HP-UX, è necessario collegare il programma ai file della libreria MQI forniti per l'ambiente in cui si sta eseguendo l'applicazione, oltre a quelli forniti dal sistema operativo.

Piattaforma IA64 (IPF)

In un'applicazione non con thread:

Tabella 28. File di libreria per applicazioni HP-UX senza thread	
File di libreria	**Ambiente**
libmqm.so	Server per C
libmqic.so & **libmqm.so**	Client per C
libmqmzf.so	Uscite di servizio installabili per C
libmqmxa.so	Interfaccia XA server
libmqmxa64.so	Interfaccia XA alternativa server
libmqcxa.so	Interfaccia XA client
libmqcxa64.so	Interfaccia XA alternativa del client
libimqi23ah.so	C++
libmqmcbrt.o	Libreria di runtime WebSphere MQ per il supporto Micro Focus COBOL
libmqmcb.so	Server per COBOL
libmqicb.so	Client per COBOL

In un'applicazione con thread:

Tabella 29. File di libreria per applicazioni con thread HP-UX	
File di libreria	**Ambiente**
libmqm_r.so	Server per C
libmqmzf_r.so & **libmqm_r.so**	Uscite di servizio installabili per C
libmqmxa_r.so	Interfaccia XA server
libmqmxa64_r.so	Interfaccia XA alternativa server
libmqcxa_r.so	Interfaccia XA client
libmqcxa64_r.so	Interfaccia XA alternativa del client
libimqi23ah_r.so	C++

IBM WebSphere MQ per Linux

Su IBM WebSphere MQ per Linux, è necessario collegare il programma ai file della libreria MQI forniti per l'ambiente in cui si sta eseguendo l'applicazione, oltre a quelli forniti dal sistema operativo.

In un'applicazione non con thread:

Tabella 30. File di libreria per applicazioni Linux senza thread	
File di libreria	**Ambiente**
libmqm.so	Server per C
libmqic.so & **libmqm.so**	Client per C
libmqmzf.so	Uscite di servizio installabili per C
libmqmxa.so	Interfaccia XA server

Tabella 30. File di libreria per applicazioni Linux senza thread (Continua)	
File di libreria	**Ambiente**
libmqmxa64.so	Interfaccia XA alternativa server
libmqcxa.so	Interfaccia XA client
libmqcxa64.so	Interfaccia XA alternativa del client
libimqc23gl.so	Client per C++
libimqs23gl.so	Server per C++

In un'applicazione con thread:

Tabella 31. File di libreria per applicazioni Linux con thread	
File di libreria	**Ambiente**
libmqm_r.so	Server per C
libmqic_r.so & **libmqm_r.so**	Client per C
libmqmzf_r.so	Uscite di servizio installabili per C
libmqmxa_r.so	Interfaccia XA server
libmqmxa64_r.so	Interfaccia XA alternativa server
libmqcxa_r.so	Interfaccia XA client
libmqcxa64_r.so	Interfaccia XA alternativa del client
libimqc23gl_r.so	Client per C++
libimqs23gl_r.so	Server per C++

IBM WebSphere MQ per Solaris

Su IBM WebSphere MQ per Solaris, è necessario collegare il proprio programma ai file della libreria MQI forniti per l'ambiente in cui si sta eseguendo l'applicazione, oltre a quelli forniti dal sistema operativo.

Tabella 32. File di libreria per applicazioni Solaris	
File di libreria	**Ambiente**
libmqm.so	Server e client per C
libmqmzse.so	Per C
libmqic.so	Client per C
libmqmcs.so	Servizi comuni per C
libmqmzf.so	Uscite di servizio installabili per C
libmqmxa.so	Interfaccia XA server
libmqmxa64.so	Interfaccia XA alternativa server
libmqcxa.so	Interfaccia XA client
libmqcxa64.so	Interfaccia XA alternativa del client
libimqc23as.a	Client per C++
libimqs23as.a	Server per C++

Parametri comuni a tutte le chiamate

Esistono due tipi di parametri comuni a tutte le chiamate: handle e codici di ritorno.

Utilizzo delle maniglie

Tutte le chiamate MQI utilizzano uno o più *handle*. Questi identificano il gestore code, la coda o un altro oggetto, messaggio o sottoscrizione, come appropriato per la chiamata.

Perché un programma comunichi con un gestore code, il programma deve avere un identificativo univoco con cui riconosce tale gestore code. Questo identificativo viene definito *handle di collegamento*, a volte indicato come *Hconn*. Per programmi di CICS , l'handle di collegamento è sempre zero. Per tutte le altre piattaforme o stili di programmi, l'handle di connessione viene restituito dalla chiamata MQCONN o MQCONNX quando il programma si connette al gestore code. I programmi passano l'handle di collegamento come un parametro di input quando utilizzano le altre chiamate.

Perché un programma possa utilizzare un oggetto WebSphere MQ , il programma deve avere un identificativo univoco con cui riconosce tale oggetto. Questo identificativo è denominato *handle dell'oggetto*, a volte indicato come *Hobj*. L'handle viene restituito dalla chiamata MQOPEN quando il programma apre l'oggetto per gestirlo. I programmi passano la gestione dell'oggetto come un parametro di input quando utilizzano chiamate MQPUT, MQGET, MQINQ, MQSET o MQCLOSE successive.

Allo stesso modo, la chiamata MQSUB restituisce un *handle della sottoscrizione* o *Hsub*, utilizzato per identificare la sottoscrizione nelle successive chiamate MQGET, MQCB o MQSUBRQ e alcune chiamate che elaborano le proprietà del messaggio utilizzano un *handle del messaggio* o *Hmsg*.

Informazioni sui codici di ritorno

Un codice di completamento e un codice motivo vengono restituiti come parametri di output da ogni chiamata. Questi sono noti collettivamente come *codici di ritorno*.

Per mostrare se una chiamata ha esito positivo, ogni chiamata restituisce un *codice di completamento* quando la chiamata è completa. Il codice di completamento è in genere MQCC_OK che indica l'esito positivo o MQCC_FAILED che indica l'errore. Alcune chiamate possono restituire uno stato intermedio, MQCC_WARNING, che indica un esito positivo parziale.

Inoltre, ogni chiamata restituisce un *codice di errore* che mostra il motivo dell'errore o dell'esito positivo parziale della chiamata. Ci sono molti codici motivo, che coprono circostanze come una coda piena, operazioni di richiamo non consentite per una coda e una particolare coda non definita per il gestore code. I programmi possono utilizzare il codice di errore per decidere come procedere. Ad esempio, possono richiedere agli utenti di modificare i propri dati di input, quindi effettuare di nuovo la chiamata oppure possono restituire un messaggio di errore all'utente.

Quando il codice di completamento è MQCC_OK, il codice di errore è sempre MQRC_NONE.

I codici di completamento e motivo per ogni chiamata sono elencati con la descrizione di tale chiamata. Consultare Descrizioni delle chiamate e selezionare la chiamata appropriata dall'elenco.

Per informazioni più dettagliate, incluse le idee per un'azione correttiva, consultare:

- Codici motivo per tutte le altre piattaforme WebSphere MQ

Specifica dei buffer

Il gestore code fa riferimento ai buffer solo se sono richiesti. Se non è richiesto un buffer su una chiamata o se il buffer è di lunghezza zero, è possibile utilizzare un puntatore null su un buffer.

Utilizzare sempre la lunghezza dati quando si specifica la dimensione del buffer richiesta.

Quando si utilizza un buffer per conservare l'emissione da una chiamata (ad esempio, per conservare i dati del messaggio per una chiamata MQGET o i valori degli attributi sottoposti a query dalla chiamata MQINQ), il gestore code tenta di restituire un codice motivo se il buffer specificato non è valido o si trova nella memoria di sola lettura. Tuttavia, potrebbe non essere sempre in grado di restituire un codice motivo.

UNIX and Linux Considerazioni

Considerazioni di cui è necessario essere consapevoli.

Prendere nota dei seguenti punti quando si sviluppano applicazioni UNIX and Linux .

La chiamata di sistema fork nei sistemi UNIX and Linux

Tenere presenti queste considerazioni quando si utilizza una chiamata di sistema fork nelle applicazioni IBM WebSphere MQ .

Se l'applicazione desidera utilizzare `fork`, il processo parent di tale applicazione deve richiamare `fork` prima di effettuare qualsiasi chiamata IBM WebSphere MQ , ad esempio, `MQCONN`o creare un oggetto IBM WebSphere MQ utilizzando **`ImqQueueManager`**.

Se l'applicazione desidera creare un processo secondario dopo aver eseguito le chiamate IBM WebSphere MQ , il codice dell'applicazione deve utilizzare un `fork()` con `exec()` per garantire che l'elemento secondario sia una nuova istanza e non una copia esatta dell'elemento principale.

Se l'applicazione non utilizza `exec()`, la chiamata API IBM WebSphere MQ effettuata all'interno del processo child restituisce MQRC_ENVIRONMENT_ERROR.

Gestione del segnale UNIX and Linux

Ciò non si applica a WebSphere MQ per z/OS o WebSphere MQ per Windows.

In generale, i sistemi UNIX, Linux e IBM i sono stati spostati da un ambiente non thread (processo) a un ambiente a più thread. Nell'ambiente non thread, alcune funzioni potevano essere implementate solo utilizzando segnali, sebbene la maggior parte delle applicazioni non avesse bisogno di essere a conoscenza dei segnali e della loro gestione. Nell'ambiente a più thread, le primitive basate su thread supportano alcune delle funzioni che erano implementate negli ambienti non thread utilizzando i segnali.

In molti casi, i segnali e la gestione dei segnali, sebbene supportati, non si adattano bene all'ambiente multithread e esistono varie limitazioni. Ciò può essere problematico quando si integra il codice dell'applicazione con diverse librerie middleware (in esecuzione come parte dell'applicazione) in un ambiente a più thread in cui ciascuno tenta di gestire i segnali. L'approccio tradizionale di salvare e ripristinare i gestori di segnale (definiti per processo), che funzionava quando c'era un solo thread di esecuzione all'interno di un processo, non funziona in un ambiente a più thread. Questo perché molti thread di esecuzione potrebbero tentare di salvare e ripristinare una risorsa a livello di processo, con risultati imprevedibili.

Applicazioni senza thread

Non applicabile su Solaris poiché tutte le applicazioni sono considerate con thread anche se utilizzano solo un singolo thread.

Ogni funzione MQI configura il proprio gestore di segnali per i segnali:

SIGALRM
SIGBUS
SIGFPE
SIGSEGV
SIGILL

I gestori degli utenti per questi vengono sostituiti per la durata della funzione MQI. Altri segnali possono essere catturati in modo normale dai gestori scritti dall'utente. Se non si installa un gestore, le azioni predefinite (ad esempio, ignora, core dump o esci) vengono lasciate al loro posto.

Dopo WebSphere MQ gestisce un segnale sincrono (SIGSEGV, SIGBUS, SIGFPE, SIGILL), tenta di passare il segnale a qualsiasi gestore del segnale registrato prima di effettuare la chiamata della funzione MQI.

Applicazioni con thread

Un thread è considerato connesso a WebSphere MQ da MQCONN (o MQCONNX) fino a MQDISC.

Segnali sincroni

I segnali sincroni si verificano in un thread specifico.

I sistemi UNIX and Linux consentono l'impostazione di un gestore di segnali per tali segnali per l'intero processo. Tuttavia, WebSphere MQ configura il proprio gestore per i seguenti segnali, nel processo dell'applicazione, mentre qualsiasi thread è connesso a WebSphere MQ:

SIGBUS
SIGFPE
SIGSEGV
SIGILL

Se si stanno scrivendo applicazioni a più thread, esiste un solo gestore di segnali a livello di processo per ogni segnale. Quando WebSphere MQ imposta i propri gestori di segnali sincroni, salva tutti i gestori precedentemente registrati per ciascun segnale. Dopo che WebSphere MQ gestisce uno dei segnali elencati, WebSphere MQ tenta di richiamare il gestore del segnale attivo al momento della prima connessione WebSphere MQ all'interno del processo. I gestori precedentemente registrati vengono ripristinati quando tutti i thread dell'applicazione si sono disconnessi da WebSphere MQ.

Poiché i gestori di segnali vengono salvati e ripristinati da WebSphere MQ, i thread dell'applicazione non devono stabilire i gestori di segnali per tali segnali, mentre è possibile che un altro thread dello stesso processo sia connesso anche a WebSphere MQ.

Nota: Quando un'applicazione, o una libreria middleware (in esecuzione come parte di un'applicazione), stabilisce un gestore del segnale mentre un thread è connesso a WebSphere MQ, il gestore del segnale dell'applicazione deve richiamare il gestore WebSphere MQ corrispondente durante l'elaborazione di tale segnale.

Quando si stabiliscono e si ripristinano gestori di segnali, il principio generale è che l'ultimo gestore di segnali da salvare deve essere il primo ad essere ripristinato:

- Quando un'applicazione stabilisce un gestore del segnale dopo la connessione a WebSphere MQ, il precedente gestore del segnale deve essere ripristinato prima che l'applicazione si disconnetta da WebSphere MQ.
- Quando un'applicazione stabilisce un gestore del segnale prima di connettersi a WebSphere MQ, l'applicazione deve disconnettersi da WebSphere MQ prima di ripristinare il gestore del segnale.

Nota: La mancata osservanza del principio generale secondo cui l'ultimo gestore del segnale da salvare deve essere il primo da ripristinare può comportare una gestione del segnale non prevista nell'applicazione e, potenzialmente, la perdita di segnali da parte dell'applicazione.

Segnali asincroni

WebSphere MQ non utilizza segnali asincroni nelle applicazioni con thread a meno che non si tratti di applicazioni client.

Ulteriori considerazioni per le applicazioni client con thread

WebSphere MQ gestisce i seguenti segnali durante l'I/O su un server. Questi segnali sono definiti dallo stack di comunicazioni. L'applicazione non deve stabilire un gestore segnali per questi segnali mentre un thread è connesso a un gestore code:

SIGPIPE (per TCP/IP)

Ulteriori considerazioni
Tenere presenti queste considerazioni quando si utilizza la gestione del segnale UNIX .

Applicazioni Fastpath (attendibili)

Le applicazioni Fastpath vengono eseguite nello stesso processo di WebSphere MQ e quindi vengono eseguite in un ambiente a più thread.

In questo ambiente WebSphere MQ gestisce i segnali sincroni SIGSEGV, SIGBUS, SIGFPE e SIGILL. Tutti gli altri segnali non devono essere consegnati all'applicazione Fastpath mentre è connessa a WebSphere MQ. Devono invece essere bloccati o gestiti dall'applicazione. Se un'applicazione Fastpath intercetta un evento di questo tipo, il gestore code deve essere arrestato e riavviato oppure può essere lasciato in uno stato non definito. Per un elenco completo delle limitazioni per le applicazioni Fastpath in MQCONNX, consultare “Connessione a un gestore code utilizzando la chiamata MQCONNX” a pagina 209.

Chiamate della funzione MQI all'interno dei gestori di segnali

Mentre ci si trova in un gestore segnali, non richiamare una funzione MQI.

Se si tenta di richiamare una funzione MQI da un gestore del segnale mentre è attiva un'altra funzione MQI, viene restituito MQRC_CALL_IN_PROGRESS. Se si tenta di richiamare una funzione MQI da un gestore segnali mentre non è attiva nessun' altra funzione MQI, è probabile che abbia esito negativo durante l'operazione a causa delle limitazioni del sistema operativo in cui solo le chiamate selettive possono essere emesse da o all'interno di un gestore.

Per i metodi del distruttore C++, che potrebbero essere richiamati automaticamente durante l'uscita del programma, potrebbe non essere possibile arrestare la chiamata delle funzioni MQI. Ignorare eventuali errori relativi a MQRC_CALL_IN_PROGRESS. Se un programma di gestione dei segnali richiama exit (), WebSphere MQ esegue il backout dei messaggi non sottoposti a commit nel punto di sincronizzazione come al solito e chiude le code aperte.

Segnali durante le chiamate MQI

Le funzioni MQI non restituiscono il codice EINTR o qualsiasi equivalente ai programmi applicativi.

Se un segnale si verifica durante una chiamata MQI e il gestore richiama *return*, la chiamata continua ad essere eseguita come se il segnale non si fosse verificato. In particolare, MQGET non può essere interrotto da un segnale per restituire immediatamente il controllo all'applicazione. Se si desidera uscire da un MQGET, impostare la coda su GET_DISABLED; in alternativa, utilizzare un loop intorno a una chiamata a MQGET con una scadenza a tempo finito (MQGMO_WAIT con gmo.WaitInterval impostato) e utilizzare il gestore del segnale (in un ambiente senza thread) o una funzione equivalente in un ambiente con thread per impostare un indicatore che interrompe il loop.

Nell'ambiente AIX , WebSphere MQ richiede che le chiamate di sistema interrotte dai segnali vengano riavviate. Quando si stabilisce il proprio gestore di segnali con sigaction (2), impostare l'indicatore SA_RESTART nel campo sa_flags della nuova struttura di azioni altrimenti WebSphere MQ potrebbe non essere in grado di completare una chiamata interrotta da un segnale.

Uscite utente e servizi installabili

Le uscite utente e i servizi installabili che vengono eseguiti come parte di un processo WebSphere MQ in un ambiente a più sottoprocessi hanno le stesse limitazioni delle applicazioni fastpath. Considerare che questi siano collegati permanentemente a WebSphere MQ e quindi non utilizzano segnali o chiamate di sistema operativo non thread - safe.

Gestori exit VMS

Gli utenti possono installare i gestori di uscita per un'applicazione WebSphere MQ utilizzando il servizio di sistema **SYS$DCLEXH** .

Il gestore di uscita riceve il controllo quando un'immagine esce. Un'uscita immagine si verifica normalmente quando si richiama il servizio Uscita ($EXIT) o Uscita forzata ($FORCEX). $FORCEX interrompe il processo di destinazione in modalità utente. Quindi tutti gli handler di uscita in modalità utente (stabiliti da $DCLEXH) iniziano ad essere eseguiti in ordine inverso di creazione. Per ulteriori dettagli sui gestori di uscita e $FORCEX, consultare il *VMS Programming Concepts Manual* e il *VMS System Services Manual*.

Se si richiama una funzione MQI dall'interno di un gestore di uscita, il comportamento della funzione dipende dal modo in cui l'immagine è stata terminata. Se l'immagine è stata terminata mentre un'altra funzione MQI è attiva, viene restituito un MQRC_CALL_IN_PROGRESS .

È possibile richiamare una funzione MQI dall'interno di un gestore di uscita se nessun' altra funzione MQI è attiva e le chiamate di aggiornamento sono disabilitate per l'applicazione WebSphere MQ . Se le chiamate di aggiornamento sono abilitate per l'applicazione WebSphere MQ , l'operazione ha esito negativo con il codice di errore MQRC_HCONN_ERROR.

L'ambito di una chiamata MQCONN o MQCONNX è in genere il thread che l'ha emessa. Se le chiamate di aggiornamento sono abilitate, il gestore di uscita viene eseguito come un thread separato e gli handle di connessione non possono essere condivisi.

I gestori di uscita vengono avviati nel contesto interrotto del processo di destinazione. È compito dell'applicazione garantire che le azioni intraprese da un gestore siano sicure e affidabili, per il contesto interrotto in modo asincrono da cui vengono chiamate.

Connessione e disconnessione da un gestore code

Per utilizzare i servizi di programmazione WebSphere MQ , un programma deve disporre di una connessione a un gestore code. Utilizzare queste informazioni per informazioni su come connettersi e disconnettersi da un gestore code.

Il modo in cui viene effettuata questa connessione dipende dalla piattaforma e dall'ambiente in cui il programma sta operando:

z/OS batch, WebSphere MQ per IBM i, WebSphere MQ su sistemi UNIX , WebSphere MQ su sistemi Linux e WebSphere MQ per Windows

I programmi in esecuzione in questi ambienti possono utilizzare la chiamata MQCONN MQI per connettersi a un gestore code e la chiamata MQDISC per disconnettersi da un gestore code. In alternativa, i programmi possono utilizzare la chiamata MQCONNX.

I programmi batch z/OS possono connettersi, consecutivamente o simultaneamente, a più gestori code sullo stesso TCB.

IMS

La control region IMS è connessa a uno o più gestori code quando viene avviata. Questa connessione è controllata dai comandi IMS . Tuttavia, i programmi di scrittura dei messaggi in coda IMS devono utilizzare la chiamata MQCONN MQI per specificare il gestore code a cui si desidera connettersi. Possono utilizzare la chiamata MQDISC per disconnettersi da tale gestore code.

Dopo una chiamata IMS che stabilisce un punto di sincronizzazione e prima di elaborare un messaggio per un altro utente, l'adattatore IMS garantisce che l'applicazione chiuda gli handle e si disconnetta dal gestore code.

I programmi IMS possono connettersi, consecutivamente o simultaneamente, a più gestori code sullo stesso TCB.

CICS Transaction Server per z/OS e CICS per MVS/ESA

I programmi CICS non hanno bisogno di eseguire alcun lavoro per connettersi a un gestore code perché il sistema CICS è connesso. Questa connessione viene generalmente effettuata automaticamente all'inizializzazione, ma è anche possibile utilizzare la transazione CKQC, che èfornito con WebSphere MQ per z/OS.

I task CICS possono connettersi solo al gestore code a cui è connessa la region CICS .

Nota: I programmi CICS possono inoltre utilizzare le chiamate MQI connect e disconnect (MQCONN e MQDISC). È possibile eseguire questa operazione in modo da poter trasferire queste applicazioni in ambienti non CICS con un minimo di ricodifica. Tuttavia, queste chiamate *sempre* vengono completate correttamente in ambiente CICS . Ciò significa che il codice di ritorno potrebbe non riflettere lo stato reale della connessione al gestore code.

TXSeries per Windows e Open Systems
Questi programmi non devono eseguire alcun lavoro per connettersi a un gestore code poiché lo stesso sistema CICS è connesso. Pertanto, è supportata solo una connessione alla volta. Le applicazioni CICS devono emettere una chiamata MQCONN per ottenere un handle di connessione e una chiamata MQDISC prima di uscire.

Utilizzare i seguenti link per ulteriori informazioni sulla connessione e la disconnessione da un gestore code:

- “Connessione a un gestore code utilizzando la chiamata MQCONN” a pagina 208
- “Connessione a un gestore code utilizzando la chiamata MQCONNX” a pagina 209
- “Disconnessione di programmi da un gestore code utilizzando MQDISC” a pagina 214

Concetti correlati
“Panoramica su Message Queue Interface” a pagina 195
Informazioni sui componenti MQI (Message Queue Interface).

“Apertura e chiusura di oggetti” a pagina 215
Queste informazioni forniscono informazioni sull'apertura e la chiusura degli oggetti WebSphere MQ .

“Inserimento di messaggi in una coda” a pagina 225
Utilizzare queste informazioni per informazioni su come inserire i messaggi su una coda.

“Richiamo dei messaggi da una coda” a pagina 241
Utilizzare queste informazioni per ottenere messaggi da una coda.

“Richiesta di informazioni e impostazione degli attributi dell'oggetto” a pagina 322
Gli attributi sono proprietà che definiscono le caratteristiche di un oggetto WebSphere MQ .

“Commit e backout delle unità di lavoro” a pagina 325
Queste informazioni descrivono come eseguire il commit e il backout di tutte le operazioni get e put recuperabili che si sono verificate in un'unità di lavoro.

“Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger” a pagina 331
Informazioni sui trigger e su come avviare le applicazioni IBM WebSphere MQ utilizzando i trigger.

“Utilizzo di MQI e cluster” a pagina 349
Esistono opzioni speciali sulle chiamate e sui codici di ritorno relativi al cluster.

Connessione a un gestore code utilizzando la chiamata MQCONN

Utilizzare queste informazioni per informazioni su come connettersi a un gestore code utilizzando la chiamata MQCONN.

In generale, è possibile connettersi a un gestore code specifico o al gestore code predefinito:

- Per IBM WebSphere MQ per z/OS, nell'ambiente batch, il gestore code predefinito è specificato nel modulo CSQBDEFV.
- Per sistemi IBM WebSphere MQ per Windows, IBM i, UNIXe Linux , nel file mqs.ini è specificato il gestore code predefinito.

In alternativa, negli ambienti z/OS MVS batch, TSO e RRS è possibile connettersi a qualsiasi gestore code all'interno di un gruppo di condivisione code. La richiesta MQCONN o MQCONNX seleziona uno qualsiasi dei membri attivi del gruppo.

Quando ci si connette a un gestore code, deve essere locale per l'attività. Deve appartenere allo stesso sistema dell'applicazione IBM WebSphere MQ .

Nell'ambiente IMS , il gestore code deve essere connesso alla control region IMS e alla region dipendente utilizzata dal programma. Il gestore code predefinito è specificato nel modulo CSQQDEFV quando è installato IBM WebSphere MQ per z/OS .

Con l'ambiente TXSeries CICS e TXSeries per Windows e AIX, il gestore code deve essere definito come una risorsa XA per CICS.

Per connettersi al gestore code predefinito, richiamare MQCONN, specificando un nome composto interamente da spazi vuoti o che inizia con un carattere null (X'00 ').

Un'applicazione deve essere autorizzata per connettersi correttamente a un gestore code. Per ulteriori informazioni, consultare Sicurezza.

L'output di MQCONN è:

- Un handle di collegamento (**Hconn**)
- Un codice di completamento
- Un codice di errore

Utilizzare l'handle di connessione nelle chiamate MQI successive.

Se il codice motivo indica che l'applicazione è già connessa a tale gestore code, l'handle di connessione restituito è uguale a quello restituito quando l'applicazione si è connessa per la prima volta. L'applicazione non deve emettere la chiamata MQDISC in questa situazione perché l'applicazione chiamante prevede di rimanere connessa.

L'ambito dell'handle di connessione è uguale all'ambito dell'handle di oggetto (consultare “Apertura di oggetti utilizzando la chiamata MQOPEN” a pagina 216).

Le descrizioni dei parametri sono fornite nella descrizione della chiamata MQCONN in MQCONN.

La chiamata MQCONN ha esito negativo se il gestore code è in uno stato di inattività quando si emette la chiamata o se il gestore code è in fase di arresto.

Ambito di MQCONN o MQCONNX

L'ambito di una chiamata MQCONN o MQCONNX è in genere il thread che l'ha emessa. Ovvero, l'handle di connessione restituito dalla chiamata è valido solo all'interno del thread che ha emesso la chiamata. È possibile effettuare una sola chiamata alla volta utilizzando l'handle. Se viene utilizzato da un thread differente, viene rifiutato come non valido. Se si dispone di più thread nell'applicazione e ciascuno desidera utilizzare le chiamate IBM WebSphere MQ , ciascuno deve emettere MQCONN o MQCONNX.

Non è necessario che ogni chiamata venga effettuata allo stesso gestore code quando un processo effettua più chiamate MQCONN. Tuttavia, è possibile stabilire solo una connessione WebSphere MQ da un thread alla volta. In alternativa, considerare “Connessioni condivise (indipendenti dal thread) con MQCONNX” a pagina 213 per consentire l'utilizzo di più connessioni WebSphere MQ da un singolo thread e di una connessione WebSphere MQ da qualsiasi thread.[1]

Se l'applicazione è in esecuzione come client, può connettersi a più di un gestore code all'interno di un thread.

Connessione a un gestore code utilizzando la chiamata MQCONNX

La chiamata MQCONNX è simile alla chiamata MQCONN, ma include opzioni per controllare il funzionamento della chiamata.

Come input per MQCONNX, è possibile fornire un nome gestore code o un nome gruppo di condivisione code su sistemi di code condivise z/OS . L'output da MQCONNX è:

- Un handle di connessione (Hconn)
- Un codice di completamento
- Un codice di errore

Utilizzare l'handle di connessione nelle chiamate MQI successive.

[1] Quando si utilizzano applicazioni a più thread con sistemi IBM WebSphere MQ su UNIX and Linux , è necessario verificare che le applicazioni abbiano una dimensione di stack sufficiente per i thread. Considerare l'utilizzo di una dimensione di stack pari o superiore a 256 KB, quando le applicazioni a più thread effettuano chiamate MQI, da sole o con altri gestori di segnali (ad esempio, CICS).

Una descrizione di tutti i parametri di MQCONNX viene fornita in MQCONNX. Il campo *Options* consente di impostare STANDARD_BINDING, FASTPATH_BINDING, SHARED_BINDING o ISOLATED_BINDING per qualsiasi versione di MQCNO. È anche possibile stabilire connessioni condivise (indipendenti dal thread) utilizzando una chiamata MQCONNX. Per ulteriori informazioni, consultare “Connessioni condivise (indipendenti dal thread) con MQCONNX” a pagina 213 .

MQCNO_STANDARD_BINDING

Per impostazione predefinita, MQCONNX (come MQCONN) implica due thread logici in cui l'applicazione WebSphere MQ e l'agent del gestore code locale vengono eseguiti in processi separati. L'applicazione WebSphere MQ richiede l'operazione WebSphere MQ e l'agent del gestore code locale esegue la richiesta. Questo è definito dall'opzione MQCNO_STANDARD_BINDING sulla chiamata MQCONNX.

se si specifica MQCNO_STANDARD_BINDING, la chiamata MQCONNX utilizza MQCNO_SHARED_BINDING o MQCNO_ISOLATED_BINDING, in base al valore dell'attributo di tipo DefaultBinddel gestore code, definito in qm.ini o nel registro Windows .

Questo è il valore predefinito.

Se si sta effettuando il collegamento alla libreria `mqm` , viene tentata prima una connessione al server standard utilizzando il tipo di collegamento predefinito. Se non è stato possibile caricare la libreria del server sottostante, viene tentata una connessione client.

- Se viene specificata la variabile di ambiente MQ_CONNECT_TYPE, è possibile fornire una delle seguenti opzioni per modificare il comportamento di MQCONN o MQCONNX se è specificato MQCNO_STANDARD_BINDING. (L'eccezione è se MQCNO_FASTPATH_BINDING viene specificato con MQ_CONNECT_TYPE impostato su LOCAL o STANDARD per consentire il downgrade delle connessioni fastpath da parte dell'amministratore senza una modifica correlata all'applicazione:

Valore	**Significato**
CLIENT	Viene tentato solo un collegamento client.
Percorso veloce	Questo valore era supportato nei release precedenti, ma viene ora ignorato se specificato.
LOCALE	Viene tentata solo una connessione server. Le connessioni Fastpath vengono ridotte a una connessione server standard.
STANDARD	Supportato per la compatibilità con le release precedenti. Questo valore viene ora considerato come LOCAL.

- Se la variabile di ambiente MQ_CONNECT_TYPE non è impostata quando viene richiamato MQCONN, viene tentata una connessione al server standard che utilizza il tipo di bind predefinito. Se il caricamento della libreria del server non riesce, viene tentata una connessione client.

MQCNO_FASTPATH_BINDING

Applicazioni attendibili implica che l'applicazione WebSphere MQ e l'agente del gestore code locale diventino lo stesso processo. Poiché il processo agent non ha più bisogno di utilizzare un'interfaccia per accedere al gestore code, queste applicazioni diventano un'estensione del gestore code. Ciò è definito dall'opzione MQCNO_FASTPATH_BINDING sulla chiamata MQCONNX.

È necessario collegare le applicazioni attendibili alle librerie WebSphere MQ con thread. Per istruzioni su come impostare un'applicazione WebSphere MQ da eseguire come sicura, consultare Opzioni MQCNO.

Questa opzione fornisce le prestazioni più elevate.

Nota: Questa opzione compromette l'integrità del gestore code: non vi è alcuna protezione dalla sovrascrittura della relativa memoria. Ciò si applica anche se l'applicazione contiene errori che

possono essere esposti ai messaggi e ad altri dati anche nel gestore code. Considerare questi problemi prima di utilizzare questa opzione.

MQCNO_SHARED_BINDING

Specificare questa opzione per eseguire l'applicazione e l'agente del gestore code locale in processi separati. Ciò mantiene l'integrità del gestore code, ossia protegge il gestore code da programmi erranti. Tuttavia, l'applicazione e l'agent gestore code locale condividono alcune risorse.

Questa opzione è intermedia tra MQCNO_FASTPATH_BINDING e MQCNO_ISOLATED_BINDING, sia in termini di protezione dell'integrità del gestore code, sia in termini di prestazioni delle chiamate MQI.

MQCNO_SHARED_BINDING viene ignorato se il gestore code non supporta questo tipo di bind. L'elaborazione continua come se l'opzione non fosse stata specificata.

Se un'applicazione si è connessa al gestore code locale utilizzando MQCNO_SHARED_BINDING, il gestore code può essere arrestato mentre l'applicazione è in esecuzione. Se si riavvia il gestore code mentre l'applicazione è ancora in esecuzione, il tentativo di avviare il gestore code ha esito negativo con l'errore AMQ7018 poiché l'applicazione è ancora in attesa delle risorse richieste dal gestore code.

Per avviare il gestore code, è necessario arrestare l'applicazione.

MQCNO_ISOLATED_BINDING

Specificare questa opzione per eseguire l'applicazione e l'agente del gestore code locale in processi separati, come per MQCNO_SHARED_BINDING. In questo caso, tuttavia, il processo dell'applicazione e l'agent del gestore code locale sono isolati l'un l'altro in quanto non condividono le risorse.

Questa è l'opzione più sicura per proteggere l'integrità del gestore code, ma fornisce le prestazioni più lente delle chiamate MQI.

MQCNO_ISOLATED_BINDING viene ignorato se il gestore code non supporta questo tipo di bind. L'elaborazione continua come se l'opzione non fosse stata specificata.

MQCNO_CLIENT_BINDING

Specificare questa opzione per rendere il tentativo dell'applicazione solo una connessione client. Questa opzione ha i seguenti limiti:

- MQCNO_CLIENT_BINDING è stato rifiutato in z/OS con MQRC_OPTIONS_ERROR.
- MQCNO_CLIENT_BINDING viene rifiutato con MQRC_OPTIONS_ERROR se è specificato con qualsiasi opzione di bind MQCNO diversa da MQCNO_STANDARD_BINDING.
- MQCNO_CLIENT_BINDING non è disponibile per Java poiché dispone di meccanismi propri per la selezione del tipo di bind.
- V 7.5.0.7 Prima di IBM WebSphere MQ Version 7.5.0, Fix Pack 7, MQCNO_CLIENT_BINDING non è disponibile per .NET poiché dispone di meccanismi propri per la scelta del tipo di bind. Da Version 7.5.0, Fix Pack 7, la limitazione all'utilizzo di .NET per MQCNO_CLIENT_BINDING viene rimossa.
- Se la variabile di ambiente MQ_CONNECT_TYPE non è impostata quando viene richiamato MQCONNX, viene tentata una connessione server standard che utilizza il tipo di collegamento predefinito. Se il caricamento della libreria del server non riesce, viene tentata una connessione client.

BINDING MQCNO_LOCAL_

Specificare questa opzione per fare in modo che l'applicazione tenti una connessione server. Se è stato specificato anche MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING o MQCNO_SHARED_BINDING, la connessione è di quel tipo ed è documentata in questa sezione. Altrimenti viene tentata una connessione al server standard utilizzando il tipo di bind predefinito. MQCNO_LOCAL_BINDING ha i seguenti limiti:

- MQCNO_LOCAL_BINDING viene ignorato su z/OS.
- MQCNO_LOCAL_BINDING viene rifiutato con MQRC_OPTIONS_ERROR se specificato con qualsiasi opzione di riconnessione MQCNO diversa da MQCNO_RECONNECT_AS_DEF.

- MQCNO_LOCAL_BINDING non è disponibile per Java poiché dispone di meccanismi propri per la scelta del tipo di bind.
- V 7.5.0.7 Prima di IBM WebSphere MQ Version 7.5.0, Fix Pack 7, MQCNO_LOCAL_BINDING non è disponibile per .NET poiché dispone di meccanismi propri per la scelta del tipo di bind. Da Version 7.5.0, Fix Pack 7, la limitazione sull'utilizzo di .NET per MQCNO_LOCAL_BINDING viene rimossa.
- Se la variabile di ambiente MQ_CONNECT_TYPE non è impostata quando viene richiamato MQCONNX, viene tentata una connessione server standard che utilizza il tipo di collegamento predefinito. Se il caricamento della libreria del server non riesce, viene tentata una connessione client.

Su z/OS queste opzioni sono tollerate, ma viene eseguita solo una connessione di collegamento standard. MQCNO Versione 3, per z/OS, consente quattro opzioni alternative:

MQCNO_SERIALIZE_CONN_TAG_QSG

Ciò consente a un'applicazione di richiedere che una sola istanza di un'applicazione venga eseguita contemporaneamente in un gruppo di condivisione code. Ciò si ottiene registrando l'utilizzo di una tag di connessione con un valore specificato o derivato dall'applicazione. La tag è una stringa di caratteri di 128 byte specificata in MQCNO Versione 3.

MQCNO_RESTRICT_CONN_TAG_QSG

Viene utilizzato quando un'applicazione è composta da più di un processo (o da un TCB), ognuno dei quali può connettersi a un gestore code. La connessione è consentita solo se non vi è alcun utilizzo corrente della tag o se l'applicazione richiedente si trova all'interno dello stesso ambito di elaborazione. Questo è lo spazio di indirizzo MVS all'interno dello stesso gruppo di condivisione code del proprietario della tag.

MQCNO_SERIALIZE_CONN_TAG_Q_MGR

È simile a MQCNO_SERIALIZE_CONN_TAG_QSG, ma viene interrogato solo il gestore code locale per verificare se la tag richiesta è già in uso.

MQCNO_RESTRICT_CONN_TAG_Q_MGR

È simile a MQCNO_RESTRICT_CONN_TAG_QSG, ma solo il gestore code locale viene interrogato per vedere se il tag richiesto è già in uso.

Limitazioni per le applicazioni attendibili

Le seguenti limitazioni si applicano alle applicazioni attendibili:

- È necessario disconnettere esplicitamente le applicazioni attendibili dal gestore code.
- È necessario arrestare le applicazioni attendibili prima di terminare il gestore code con il comando endmqm.
- Non utilizzare segnali asincroni e interruzioni timer (come `sigkill`) con MQCNO_FASTPATH_BINDING.
- Su tutte le piattaforme, un thread all'interno di un'applicazione attendibile non può connettersi a un gestore code mentre un altro thread nello stesso processo è connesso a un gestore code differente.
- Sui sistemi WebSphere MQ su UNIX and Linux è necessario utilizzare mqm come userID e groupID effettivi per tutte le chiamate MQI. È possibile modificare questi ID prima di effettuare una chiamata non MQI che richiede autenticazione (ad esempio, l'apertura di un file), ma è *necessario* modificarlo nuovamente in mqm prima di effettuare la successiva chiamata MQI.
- Su WebSphere MQ per HP-UX, è probabile che le applicazioni fast - path a più thread debbano impostare una dimensione di stack maggiore rispetto a quella predefinita. Utilizzare una dimensione di 256 KB.
- Su WebSphere MQ per Windows le applicazioni a 64 bit attendibili non sono supportate. Se si tenta di eseguire un'applicazione attendibile a 64 bit, verrà eseguito il downgrade a una connessione collegata standard.

- Su WebSphere MQ su sistemi UNIX and Linux , le applicazioni a 32 bit attendibili non sono supportate. Se si tenta di eseguire un'applicazione a 32 bit attendibile, verrà eseguito il downgrade a una connessione collegata standard.

Connessioni condivise (indipendenti dal thread) con MQCONNX

Utilizzare queste informazioni per informazioni sulle connessioni condivise con MQCONNX e alcune note di utilizzo da considerare.

Nota: Non supportato su WebSphere MQ per z/OS.

Su piattaforme WebSphere MQ diverse da WebSphere MQ for z/OS, una connessione effettuata con MQCONN è disponibile solo per il thread che ha effettuato la connessione. Le opzioni sulla chiamata MQCONNX consentono di creare una connessione che può essere condivisa da tutti i thread in un processo. Se l'applicazione è in esecuzione in un ambiente transazionale che richiede l'emissione di chiamate MQI sullo stesso thread, è necessario utilizzare la seguente opzione predefinita:

MQCNO_HANDLE_SHARE_NONE
Crea una connessione non condivisa.

Nella maggior parte degli altri ambienti, è possibile utilizzare una delle seguenti opzioni di connessione condivisa, indipendente dal thread:

MQCNO_HANDLE_SHARE_BLOCK
Crea una connessione condivisa. Su una connessione MQCNO_HANDLE_SHARE_BLOCK , se la connessione è attualmente utilizzata da una chiamata MQI su un altro thread, la chiamata MQI attende il completamento della chiamata MQI corrente.

MQCNO_HANDLE_SHARE_NO_BLOCK
Crea una connessione condivisa. Su una connessione MQCNO_HANDLE_SHARE_NO_BLOCK , se la connessione è attualmente utilizzata da una chiamata MQI su un altro thread, la chiamata MQI non riesce immediatamente con un motivo MQRC_CALL_IN_PROGRESS.

Tranne che per l'ambiente MTS (Microsoft Transaction Server), il valore predefinito è MQCNO_HANDLE_SHARE_NONE. Nell'ambiente MTS, il valore predefinito è MQCNO_HANDLE_SHARE_BLOCK.

Un handle di connessione viene restituito dalla chiamata MQCONNX . L'handle può essere utilizzato da chiamate MQI successive da qualsiasi thread nel processo, associando tali chiamate all'handle restituito da MQCONNX. Le chiamate MQI che utilizzano un singolo handle condiviso vengono serializzate tra i thread.

Ad esempio, la seguente sequenza di attività è possibile con un handle condiviso:

1. Il thread 1 emette MQCONNX e ottiene un handle condiviso *h1*
2. Il thread 1 apre una coda ed emette una richiesta get utilizzando *h1*
3. Il thread 2 emette una richiesta di inserimento utilizzando *h1*
4. Il thread 3 emette una richiesta di inserimento utilizzando *h1*
5. Il thread 2 emette MQDISC utilizzando *h1*

Mentre l'handle è utilizzato da qualsiasi thread, l'accesso alla connessione non è disponibile per altri thread. In circostanze in cui è accettabile che un thread attenda il termine di qualsiasi chiamata precedente da un altro thread, utilizzare MQCONNX con l'opzione MQCNO_HANDLE_SHARE_BLOCK.

Tuttavia, il blocco può causare difficoltà. Si supponga che nel passo “2” a pagina 213, il thread 1 emetti una richiesta get che attende i messaggi che potrebbero non essere ancora arrivati (un get con attesa). In questo caso, anche i thread 2 e 3 vengono lasciati in attesa (bloccati) per tutto il tempo richiesto dalla richiesta get sul thread 1. Se si preferisce che una chiamata MQI restituisca un messaggio di errore se un'altra chiamata MQI è già in esecuzione sull'handle, utilizzare MQCONNX con l'opzione MQCNO_HANDLE_SHARE_NO_BLOCK.

Note sull'utilizzo della connessione condivisa

1. Tutti gli handle di oggetto (Hobj) creati aprendo un oggetto sono associati a un Hconn; quindi per un Hconn condiviso, gli Hobj sono anche condivisi e utilizzabili da qualsiasi thread che utilizza l'Hconn. Allo stesso modo, qualsiasi unità di lavoro avviata sotto un Hconn è associata a tale Hconn; quindi anche questo è condiviso tra i thread con l'Hconn condiviso.
2. *Qualsiasi thread* può chiamare MQDISC per scollegare un Hconn condiviso, non solo il thread che ha richiamato il corrispondente MQCONNX. MQDISC termina l'Hconn rendendolo non disponibile per tutti i thread.
3. Un singolo thread può utilizzare più Hconn condivisi in modo seriale, ad esempio utilizzare MQPUT per inserire un messaggio in un Hconn condiviso, quindi inserire un altro messaggio utilizzando un altro Hconn condiviso, con ogni operazione in una diversa unità di lavoro locale.
4. Gli hconn condivisi non possono essere utilizzati all'interno di un'unità di lavoro globale.

Utilizzo delle opzioni di chiamata MQCONNX con MQ_CONNECT_TYPE

Utilizzare queste informazioni per comprendere le diverse opzioni di chiamata MQCONNX come vengono utilizzate con MQ_CONNECT_TYPE.

Su sistemi WebSphere MQ per IBM i, WebSphere MQ per Windowse WebSphere MQ su sistemi UNIX and Linux , è possibile utilizzare la variabile di ambiente MQ_CONNECT_TYPE in combinazione con il tipo di binding specificato nel campo *Options* della struttura MQCNO utilizzata su una chiamata MQCONNX.

Tabella 33. La variabile di ambiente MQ_CONNECT_TYPE

Opzione chiamata MQCONNX	**Variabile di ambiente MQ_CONNECT_TYPE**	**Risultato**
STANDARD	NON DEFINITO	STANDARD
STANDARD	STANDARD	STANDARD
STANDARD	Percorso veloce	STANDARD
STANDARD	CLIENTE	CLIENTE
STANDARD	LOCALE	STANDARD

Se MQCNO_STANDARD_BINDING non viene specificato, è possibile utilizzare MQCNO_NONE, che assume il valore predefinito MQCNO_STANDARD_BINDING.

Disconnessione di programmi da un gestore code utilizzando MQDISC

Utilizzare queste informazioni per informazioni sulla disconnessione dei programmi da un gestore code utilizzando MQDISC.

Quando un programma connesso a un gestore code utilizzando la chiamata MQCONN o MQCONNX ha terminato tutte le interazioni con il gestore code, interrompe la connessione utilizzando la chiamata MQDISC, tranne:

- Su applicazioni CICS Transaction Server for z/OS , dove la chiamata è facoltativa, a meno che non sia stato utilizzato MQCONNX e si desideri eliminare la tag di connessione prima del termine dell'applicazione.
- In WebSphere MQ per IBM i , dove, quando ci si scollega dal sistema operativo, viene effettuata una chiamata MQDISC implicita.

Come input per la chiamata MQDISC, è necessario fornire l'handle di connessione (Hconn) restituito da MQCONN o MQCONNX quando si è connessi al gestore code.

Tranne in CICS su z/OS, dopo che MQDISC viene richiamato, l'handle di connessione (Hconn) non è più valido e non è possibile emettere ulteriori chiamate MQI finché non si richiama nuovamente MQCONN o MQCONNX. MQDISC esegue un MQCLOSE implicito per tutti gli oggetti ancora aperti utilizzando questo handle.

Se si utilizza MQCONNX per connettersi a WebSphere MQ per z/OS, MQDISC termina anche l'ambito della tag di connessione stabilita da MQCONNX. Tuttavia, in un'applicazione CICS, IMSo RRS, se esiste un'unità di ripristino attiva associata a una tag di connessione, MQDISC viene rifiutato con un codice motivo di MQRC_CONN_TAG_NOT_RELEASED.

Le descrizioni dei parametri vengono fornite nella descrizione della chiamata MQDISC in MQDISC.

Quando non viene emesso alcun MQDISC

Una connessione standard non condivisa (Hconn) viene ripulita quando il thread di creazione termina. Una connessione condivisa viene solo implicitamente ripristinata e disconnessa quando termina l'intero processo. Se il thread che ha creato l'Hconn condiviso termina mentre l'Hconn esiste ancora, l'Hconn è ancora utilizzabile.

Controllo autorizzazione

Le chiamate MQCLOSE e MQDISC di solito non eseguono alcun controllo di autorizzazione.

Nel normale corso degli eventi, un lavoro che dispone dell'autorizzazione per aprire o connettersi a un oggetto WebSphere MQ si chiude o si disconnette da tale oggetto. Anche se l'autorizzazione di un lavoro che si è collegato o ha aperto un oggetto WebSphere MQ viene revocata, le chiamate MQCLOSE e MQDISC vengono accettate.

Apertura e chiusura di oggetti

Queste informazioni forniscono informazioni sull'apertura e la chiusura degli oggetti WebSphere MQ .

Per eseguire una delle operazioni riportate di seguito, è necessario prima *aprire* l'oggetto WebSphere MQ pertinente:

- Inserire i messaggi su una coda
- Richiamare (sfogliare o richiamare) i messaggi da una coda
- Impostare gli attributi di un oggetto
- Interroga sugli attributi di qualsiasi oggetto

Utilizzare la chiamata MQOPen per aprire l'oggetto, utilizzando le opzioni della chiamata per specificare cosa si desidera fare con l'oggetto. L'unica eccezione è se si desidera inserire un singolo messaggio in una coda, quindi chiudere immediatamente la coda. In questo caso, puoi ignorare lo stage *opening* utilizzando la chiamata MQPUT1 (vedi “Inserimento di un messaggio su una coda utilizzando la chiamata MQPUT1” a pagina 234).

Prima di aprire un oggetto utilizzando la chiamata MQOPEN, è necessario connettere il programma a un gestore code. Questo è spiegato in dettaglio, per tutti gli ambienti, in “Connessione e disconnessione da un gestore code” a pagina 207.

Ci sono quattro tipi di oggetti WebSphere MQ che possono essere aperti:

- Coda
- Elenco nomi
- Definizione di processo
- Gestore code

Tutti questi oggetti vengono aperti in modo simile utilizzando la chiamata MQOPEN. Per ulteriori informazioni sugli oggetti WebSphere MQ , consultare Oggetti.

È possibile aprire lo stesso oggetto più di una volta e ogni volta che si ottiene un nuovo handle di oggetto.È possibile esaminare i messaggi su una coda utilizzando un handle e rimuovere i messaggi dalla stessa coda utilizzando un altro handle. Ciò consente di risparmiare utilizzando le risorse per chiudere e riaprire lo stesso oggetto. È inoltre possibile aprire una coda per sfogliare *e* rimuovere i messaggi contemporaneamente.

Inoltre, è possibile aprire più oggetti con un singolo MQOPEN e chiuderli utilizzando MQCLOSE. Consultare “Liste di distribuzione” a pagina 235 per informazioni su come eseguire questa operazione.

Quando si tenta di aprire un oggetto, il gestore code verifica che l'utente sia autorizzato ad aprire tale oggetto per le opzioni specificate nella chiamata MQOPEN.

Gli oggetti vengono chiusi automaticamente quando un programma si disconnette dal gestore code. Nell'ambiente IMS , la disconnessione viene forzata quando un programma avvia l'elaborazione per un nuovo utente in seguito a una chiamata GU (get unique) IMS . Sulla piattaforma IBM i , gli oggetti vengono chiusi automaticamente al termine di un lavoro.

È buona prassi di programmazione chiudere gli oggetti aperti. Utilizzare la chiamata MQCLOSE per eseguire questa operazione.

Utilizzare i seguenti collegamenti per ulteriori informazioni sull'apertura e la chiusura di oggetti:

- “Apertura di oggetti utilizzando la chiamata MQOPEN” a pagina 216
- “Creazione di code dinamiche” a pagina 224
- “Apertura di code remote” a pagina 224
- “Chiusura di oggetti mediante la chiamata MQCLOSE” a pagina 225

Concetti correlati

“Panoramica su Message Queue Interface” a pagina 195
Informazioni sui componenti MQI (Message Queue Interface).

“Connessione e disconnessione da un gestore code” a pagina 207
Per utilizzare i servizi di programmazione WebSphere MQ , un programma deve disporre di una connessione a un gestore code. Utilizzare queste informazioni per informazioni su come connettersi e disconnettersi da un gestore code.

“Inserimento di messaggi in una coda” a pagina 225
Utilizzare queste informazioni per informazioni su come inserire i messaggi su una coda.

“Richiamo dei messaggi da una coda” a pagina 241
Utilizzare queste informazioni per ottenere messaggi da una coda.

“Richiesta di informazioni e impostazione degli attributi dell'oggetto” a pagina 322
Gli attributi sono proprietà che definiscono le caratteristiche di un oggetto WebSphere MQ .

“Commit e backout delle unità di lavoro” a pagina 325
Queste informazioni descrivono come eseguire il commit e il backout di tutte le operazioni get e put recuperabili che si sono verificate in un'unità di lavoro.

“Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger” a pagina 331
Informazioni sui trigger e su come avviare le applicazioni IBM WebSphere MQ utilizzando i trigger.

“Utilizzo di MQI e cluster” a pagina 349
Esistono opzioni speciali sulle chiamate e sui codici di ritorno relativi al cluster.

Apertura di oggetti utilizzando la chiamata MQOPEN

Utilizzare queste informazioni per informazioni sull'apertura di oggetti utilizzando la chiamata MQOPEN.

Come input della chiamata MQOPEN, è necessario fornire:

- Un handle di connessione. Per le applicazioni CICS su z/OS, è possibile specificare la costante MQHC_DEF_HCONN (che ha il valore zero) o utilizzare l'handle di connessione restituito dalla chiamata MQCONN o MQCONNX. Per altri programmi, utilizzare sempre l'handle di connessione restituito dalla chiamata MQCONN o MQCONNX.
- Una descrizione dell'oggetto che si desidera aprire, utilizzando MQOD (object descriptor structure).
- Una o più opzioni che controllano l'azione della chiamata.

L'output di MQOPEN è:

- Un handle di oggetto che rappresenta il proprio accesso all'oggetto. Utilizzare questa opzione per l'input a tutte le chiamate MQI successive.
- Una struttura descrittore oggetto modificata, se si sta creando una coda dinamica (ed è supportata sulla piattaforma).
- Un codice di completamento.
- Un codice di errore.

Ambito di un handle di oggetto

L'ambito di un handle di oggetto (Hobj) è uguale all'ambito di un handle di connessione (Hconn).

Ciò è trattato in "Ambito di MQCONN o MQCONNX" a pagina 209 e "Connessioni condivise (indipendenti dal thread) con MQCONNX" a pagina 213. Tuttavia, ci sono ulteriori considerazioni in alcuni ambienti:

CICS
In un programma CICS , è possibile utilizzare l'handle solo all'interno della stessa attività CICS da cui è stata effettuata la chiamata MQOPEN.

IMS e z/OS batch
Negli ambienti IMS e batch, è possibile utilizzare l'handle all'interno della stessa attività, ma non all'interno di alcuna attività secondaria.

Le descrizioni dei parametri della chiamata MQOPEN sono fornite in MQOPEN.

Le seguenti sezioni descrivono le informazioni che è necessario fornire come input per MQOPEN.

Identificazione degli oggetti (la struttura MQOD)

Utilizzare la struttura MQOD per identificare l'oggetto che si desidera aprire. Questa struttura è un parametro di immissione per la chiamata MQOPEN. (La struttura viene modificata dal gestore code quando si utilizza una chiamata MQOPEN per creare una coda dinamica).

Per i dettagli completi sulla struttura MQOD, consultare MQOD.

Per informazioni sull'utilizzo della struttura MQOD per gli elenchi di distribuzione, consultare "Utilizzo della struttura MQOD" a pagina 237 in "Liste di distribuzione" a pagina 235.

Risoluzione nomi

In che modo la chiamata MQOPEN risolve i nomi di code e gestori code.

Nota: Un alias gestore code è una definizione di coda remota senza un campo RNAME .

Quando si apre una coda WebSphere MQ , la chiamata MQOPEN esegue una funzione di risoluzione dei nomi sul nome della coda specificato. Ciò determina su quale coda il gestore code esegue operazioni successive. Ciò significa che quando si specifica il nome di una coda alias o di una coda remota nel descrittore oggetto (MQOD), la chiamata risolve il nome in una coda locale o in una coda di trasmissione. Se una coda viene aperta per qualsiasi tipo di input, ricerca o impostazione, si risolve in una coda locale se ne esiste una e non riesce se non ne esiste una. Si risolve in una coda non locale solo se è aperta solo per l'emissione, solo per l'interrogazione o solo per l'emissione e l'interrogazione. Consultare Tabella 34 a pagina 218 per una panoramica del processo di risoluzione dei nomi. Il nome fornito in *ObjectQMgrName* viene risolto *prima* di quello in *ObjectName*.

La Tabella 34 a pagina 218 mostra anche come utilizzare una definizione locale di una coda remota per definire un alias per il nome di un gestore code. Ciò consente di selezionare quale coda di trasmissione viene utilizzata quando si inseriscono i messaggi su una coda remota, in modo da poter, ad esempio, utilizzare una sola coda di trasmissione per i messaggi destinati a molti gestori code remoti.

Per utilizzare la seguente tabella, leggere prima le due colonne di sinistra, sotto l'intestazione **Input to MQOD**, e selezionare il caso appropriato. Quindi leggere la riga corrispondente, seguendo le istruzioni. Seguendo le istruzioni nelle colonne **Nomi risolti** , è possibile tornare alle colonne **Input to MQOD** e inserire i valori come indicato oppure è possibile uscire dalla tabella con i risultati forniti. Ad esempio, potrebbe essere necessario immettere *ObjectName*.

Tabella 34. Risoluzione dei nomi delle code quando si utilizza MQOPEN

Input in MQOD		**Nomi risolti**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Coda di trasmissione**
Gestore code locale o vuoto	Coda locale senza attributo CLUSTER	Gestore code locale	Immettere *ObjectName*	Non applicabile (coda locale utilizzata)
Gestore code vuoto	Coda locale con attributo CLUSTER	Gestore code cluster selezionato di gestione del carico di lavoro o gestore code cluster specifico selezionato in PUT	Immettere *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE e coda locale utilizzata SYSTEM.QSG.TRANSMIT.QUEUE (vedere nota)
Gestore code locale	Coda locale con attributo CLUSTER	Gestore code locale	Immettere *ObjectName*	Non applicabile (coda locale utilizzata)
Gestore code locale o vuoto	Coda modello	Gestore code locale	Nome generato	Non applicabile (coda locale utilizzata)
Gestore code locale o vuoto	Coda alias con o senza attributo CLUSTER	Eseguire di nuovo la risoluzione del nome con *ObjectQMgrNome* non modificato e immettere *ObjectName* impostato su *BaseQName* nell'oggetto definizione coda alias. Non deve risolversi in un alias definito localmente in cui è specificato *ObjectQMgrNome* , ma può risolversi in un alias cluster (ospitato su altri gestori code) in cui *ObjectQMgrNome* è vuoto.		
Gestore code locale	Coda alias con attributo CLUSTER	L'alias non deve risolversi in una coda cluster non definita localmente o in una coda cluster con lo stesso *ObjectName* dell'alias.		
Gestore code vuoto	Coda alias con attributo CLUSTER	L'alias può essere risolto in una coda cluster con lo stesso *ObjectName* dell'alias.		

Tabella 34. Risoluzione dei nomi delle code quando si utilizza MQOPEN (Continua)

Input in MQOD		**Nomi risolti**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Coda di trasmissione**
Gestore code locale o vuoto	definizione locale di coda remota	Eseguire di nuovo la risoluzione del nome con *ObjectQMgrName* impostato su *RemoteQMgrNamee* *ObjectName* impostato su *RemoteQName*. Non deve risolvere le code remote		Nome dell'attributo *XmitQName* , se non vuoto; altrimenti *RemoteQMgrName* nell'oggetto di definizione della coda remota. SYSTEM.QSG.TRANSMIT.QUEUE (vedere nota)
Gestore code vuoto	Nessun oggetto locale corrispondente ; coda cluster trovata	Gestore code cluster selezionato di gestione del carico di lavoro o gestore code cluster specifico selezionato in PUT	Immettere *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (vedere nota)
Gestore code locale o vuoto	Nessun oggetto locale corrispondente ; coda cluster non trovata		Errore, coda non trovata	Non applicabile
Nome del gestore code nello stesso gruppo di condivisione code del gestore code locale	Coda condivisa locale	Gestore code locale	Immettere *ObjectName*	Non applicabile
Nome di una coda di trasmissione locale	(Non risolto)	Immettere il nome *ObjectQMgr*	Immettere *ObjectName*	Immettere il nome *ObjectQMgr* SYSTEM.QSG.TRANSMIT.QUEUE (vedere nota)
Definizione alias del gestore code (*RemoteQMgrName* può essere il gestore code locale)	(Non risolto, coda remota)	Eseguire nuovamente la risoluzione dei nomi con *ObjectQMgrName* impostato su *RemoteQMgrName*. Non deve essere risolto in code remote	Immettere *ObjectName*	Nome dell'attributo *XmitQName* , se non vuoto; altrimenti *RemoteQMgrName* nell'oggetto di definizione della coda remota. SYSTEM.QSG.TRANSMIT.QUEUE (vedere nota)
Il gestore code non è il nome di alcun oggetto locale; sono stati trovati i gestori code del cluster o l'alias del gestore code	(Non risolto)	*ObjectQMgrNome* o gestore code cluster specifico selezionato in PUT	Immettere *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (vedere nota)

Tabella 34. Risoluzione dei nomi delle code quando si utilizza MQOPEN (Continua)				
Input in MQOD		**Nomi risolti**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Coda di trasmissione**
Il gestore code non è il nome di alcun oggetto locale; nessun oggetto cluster trovato	(Non risolto)	Immettere il nome *ObjectQMgr*	Immettere *ObjectName*	Attributo *DefXmitQName* del gestore code in cui è supportato *DefXmitQName* . SYSTEM.QSG.TRANSMIT.QUEUE (vedere nota)

Note:

1. *BaseQName* è il nome della coda di base dalla definizione della coda alias.
2. *RemoteQName* è il nome della coda remota dalla definizione locale della coda remota.
3. *RemoteQMgrName* è il nome del gestore code remoto dalla definizione locale della coda remota.
4. *XmitQName* è il nome della coda di trasmissione dalla definizione locale della coda remota.
5. Quando si utilizzano i gestori code WebSphere MQ per z/OS che fanno parte di un gruppo di condivisione code (QSG), è possibile utilizzare il nome del QSG invece del nome del gestore code locale in Tabella 34 a pagina 218.

   Se il gestore code locale non può aprire la coda di destinazione o inserire un messaggio nella coda, il messaggio viene trasferito al nome ObjectQMgrspecificato tramite l'accodamento all'interno del gruppo o un canale WebSphere MQ .
6. Nella colonna *ObjectName* della tabella, CLUSTER fa riferimento agli attributi CLUSTER e CLUSNL della coda.
7. Il SISTEMA SYSTEM.QSG.TRANSMIT.QUEUE viene utilizzato se i gestori code locali e remoti si trovano nello stesso gruppo di condivisione code; l'accodamento all'interno del gruppo è abilitato.
8. Se è stata assegnata una diversa coda di trasmissione cluster a ciascun canale mittente del cluster, SYSTEM.CLUSTER.TRANSMIT.QUEUE potrebbe non essere il nome della coda di trasmissione cluster. Per ulteriori informazioni su più code di trasmissione del cluster, consultare Cluster: Pianificazione della configurazione delle code di trasmissione del cluster .
9. Nella situazione in cui il gestore code non è il nome di alcun oggetto locale; sono stati trovati gestori code cluster o alias del gestore code.

   Quando è stato fornito un nome gestore code utilizzando **ObjectQMgrName**e sono presenti più canali cluster con nomi cluster differenti noti al gestore code locale che raggiungerebbero tale destinazione, è possibile che uno di questi canali venga utilizzato per spostare il messaggio, indipendentemente dal nome cluster della coda di destinazione.

   Ciò potrebbe essere imprevisto, se si stavano anticipando i messaggi per quella coda da inviare solo attraverso un canale che ha lo stesso nome cluster della coda.

   Tuttavia, **ObjectQMgrName** ha la precedenza in questo caso e il bilanciamento del carico di lavoro del cluster prende in considerazione tutti i canali che potrebbero raggiungere tale gestore code, indipendentemente dal nome del cluster in cui si trovano.

L'apertura di una coda alias apre anche la coda di base in cui l'alias viene risolto e l'apertura di una coda remota apre anche la coda di trasmissione. Pertanto, non è possibile eliminare la coda specificata o la coda in cui viene risolta mentre l'altra è aperta.

Mentre una coda alias non è in grado di risolvere un'altra coda alias definita localmente (condivisa o meno in un cluster), la risoluzione in una coda alias cluster definita in remoto è consentita e può quindi essere specificata come coda di base.

Il nome della coda risolta e il nome del gestore code risolto vengono memorizzati nei campi *ResolvedQName* e *ResolvedQMgrName* in MQOD.

Per ulteriori informazioni sulla risoluzione dei nomi in un ambiente di accodamento distribuito, consultare What is queue name resolution?.

Utilizzo delle opzioni della chiamata MQOPEN

Nel parametro *Options* della chiamata MQOPEN, è necessario scegliere una o più opzioni per controllare l'accesso fornito all'oggetto che si sta aprendo. Con queste opzioni è possibile:

- Aprire una coda e specificare che tutti i messaggi inseriti in tale coda devono essere indirizzati alla stessa istanza di essa
- Aprire una coda per consentire l'inserimento di messaggi su di essa
- Aprire una coda per consentire di sfogliare i messaggi su di essa
- Aprire una coda per consentire la rimozione dei messaggi da essa
- Aprire un oggetto per consentire di interrogare e impostare i suoi attributi (ma è possibile impostare solo gli attributi delle code)
- Aprire un argomento o una stringa di argomenti per pubblicare i messaggi
- Associa informazioni di contesto a un messaggio
- Designare un identificativo utente alternativo da utilizzare per i controlli di sicurezza
- Controlla la chiamata se il gestore code è in uno stato di inattività

Opzione MQOPEN per la coda cluster
Il bind utilizzato per l'handle della coda viene preso dall'attributo della coda *DefBind* , che può assumere il valore MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXEDo MQBND_BIND_ON_GROUP.

Per instradare tutti i messaggi inseriti in una coda utilizzando MQPUT allo stesso gestore code tramite lo stesso instradamento, utilizzare l'opzione MQOO_BIND_ON_OPEN sulla chiamata MQOPEN .

Per specificare che una destinazione deve essere selezionata al momento di MQPUT, ossia, su base messaggio per messaggio, utilizzare l'opzione MQOO_BIND_NOT_FIXED sulla chiamata MQOPEN .

Per specificare che tutti i messaggi in un gruppo di messaggi inseriti in una coda utilizzando MQPUT sono assegnati alla stessa istanza di destinazione, utilizzare l'opzione MQOO_BIND_ON_GROUP nella chiamata MQOPEN .

È necessario specificare MQOO_BIND_ON_OPEN o MQOO_BIND_ON_GROUP quando si utilizzano gruppi di messaggi con cluster per garantire che tutti i messaggi nel gruppo vengano elaborati alla stessa destinazione.

Se non si specifica alcuna di queste opzioni, viene utilizzato il valore predefinito, MQOO_BIND_AS_Q_DEF.

Se si specifica il nome di un gestore code in MQOD, viene selezionata la coda in tale gestore code. Se il nome del gestore code è vuoto, è possibile selezionare qualsiasi istanza. Per ulteriori informazioni, fare riferimento a “MQOPEN e cluster” a pagina 350.

Se si apre una coda cluster utilizzando una definizione QALIAS , alcuni attributi della coda vengono definiti dalla coda alias e non dalla coda di base. Gli attributi del cluster sono tra gli attributi della definizione della coda di base sovrascritti dalla coda alias. Ad esempio, nel seguente frammento, la coda cluster viene aperta con MQOO_BIND_NOT_FIXED e non con MQOO_BIND_ON_OPEN. La definizione della coda del cluster viene pubblicizzata in tutto il cluster, la definizione della coda alias è locale per il gestore code.

```
DEFINE QLOCAL(CLQ1)  CLUSTER(MYCLUSTER) DEFBIND(OPEN) REPLACE
DEFINE QALIAS(ACLQ1) TARGET(CLQ1) DEFBIND(NOTFIXED) REPLACE
```

Opzione MQOPEN per l'inserimento di messaggi
Per aprire una coda o un argomento in cui inserire i messaggi, utilizzare l'opzione MQOO_OUTPUT.

Opzione MQOPEN per la ricerca dei messaggi
Per aprire una coda in modo da poter *sfogliare* i messaggi su di essa, utilizzare la chiamata MQOPEN con l'opzione MQOO_BROWSE.

Questo crea un *cursore di esplorazione* che il gestore code utilizza per identificare il successivo messaggio sulla coda. Per ulteriori informazioni, vedere "Visualizzazione dei messaggi su una coda" a pagina 273.

Nota:

1. Non è possibile ricercare i messaggi su una coda remota; non aprire una coda remota utilizzando l'opzione MQOO_BROWSE.
2. Non è possibile specificare questa opzione quando si apre un elenco di distribuzione. Per ulteriori informazioni sugli elenchi di distribuzione, consultare "Liste di distribuzione" a pagina 235.
3. Utilizzare MQOO_CO_OP insieme a MQOO_BROWSE se si utilizza la navigazione cooperativa; consultare Opzioni

Opzioni MQOPEN per la rimozione dei messaggi
Tre opzioni controllano l'apertura di una coda per rimuovere i messaggi da essa.

È possibile utilizzarne solo uno in qualsiasi chiamata MQOPEN. Queste opzioni definiscono se il proprio programma ha accesso esclusivo o condiviso alla coda. *Accesso esclusivo* significa che, fino a quando non si chiude la coda,solo è possibile rimuovere i messaggi. Se un altro programma tenta di aprire la coda per rimuovere i messaggi, la chiamata MQOPEN ha esito negativo. *Accesso condiviso* significa che più di un programma può rimuoveredalla coda.

L'approccio più consigliato consiste nell'accettare il tipo di accesso previsto per la coda al momento della definizione della coda. La definizione della coda comportava l'impostazione di *Shareability* e*DefInputOpenOption* Attributi. Per accettarlo, utilizzare l'opzione MQOO_INPUT_AS_Q_DEF. Fare riferimento a Tabella 35 a pagina 222 per informazioni su come l'impostazione di questi attributi influisce sul tipo di accesso che verrà fornito quando si utilizza questa opzione.

Tabella 35. In che modo gli attributi e le opzioni della coda della chiamata MQOPEN influenzano l'accesso alle code

Attributi Coda		**Tipo di accesso con opzioni MQOPEN**		
Shareability	*DefInputOpenOption*	**DEF Q_AS**	**CONDIVISO**	**Esclusivo**
Condivisibile	CONDIVISO	condiviso	condiviso	esclusivo
Condivisibile	Esclusivo	esclusivo	condiviso	esclusivo
NON_CONDIVISIBILE *	SARA*	esclusivo	esclusivo	esclusivo
NON_CONDIVISIBILE	Esclusivo	esclusivo	esclusivo	esclusivo
Nota: * Anche se è possibile definire una coda per avere questa combinazione di attributi, l'opzione di apertura di input predefinita viene sovrascritta dall'attributo di condivisibilità.				

In alternativa:

- Se si sa che la propria applicazione può funzionare correttamente anche se altri programmi possono rimuovere i messaggi dalla coda contemporaneamente, utilizzare l'opzione MQOO_INPUT_SHARED. Tabella 35 a pagina 222 mostra come, in alcuni casi, si avrà accesso esclusivo alla coda, anche con questa opzione.
- Se si sa che l'applicazione può funzionare correttamente solo se ad altri programmi viene impedito di rimuovere i messaggi dalla coda contemporaneamente, utilizzare l'opzione MQOO_INPUT_EXCLUSIVE.

Nota:

1. Non è possibile rimuovere messaggi da una coda remota. Pertanto, non è possibile aprire una coda remota utilizzando le opzioni MQOO_INPUT_ *.

2. Non è possibile specificare questa opzione quando si apre un elenco di distribuzione. Per ulteriori informazioni, fare riferimento a “Liste di distribuzione” a pagina 235.

Opzioni MQOPEN per l'impostazione e la richiesta di informazioni sugli attributi
Per aprire una coda in modo da poterne impostare gli attributi, utilizzare l'opzione MQOO_SET.

Non è possibile impostare gli attributi di qualsiasi altro tipo di oggetto (consultare “Richiesta di informazioni e impostazione degli attributi dell'oggetto” a pagina 322).

Per aprire un oggetto in modo da poter richiedere informazioni sui relativi attributi, utilizzare l'opzione MQOO_INQUIRE.

Nota: Non è possibile specificare questa opzione quando si apre un elenco di distribuzione.

Opzioni MQOPEN relative al contesto del messaggio
Se si desidera poter associare le informazioni di contesto a un messaggio quando le si inserisce in una coda, è necessario utilizzare una delle opzioni di contesto del messaggio quando si apre la coda.

Le opzioni consentono di distinguere tra le informazioni di contesto relative all' *utente* che ha originato il messaggio e quelle relative all'applicazione che ha originato il messaggio. Inoltre, è possibile scegliere di impostare le informazioni di contesto quando si inserisce il messaggio nella coda oppure è possibile scegliere che il contesto venga acquisito automaticamente da un altro handle della coda.

Concetti correlati
“Contesto messaggio” a pagina 39
Le informazioni relative al *contesto del messaggio* consentono all'applicazione che richiama il messaggio di individuare il creatore del messaggio.

“Controllo delle informazioni di contesto” a pagina 232
Quando si utilizza la chiamata MQPUT o MQPUT1 per inserire un messaggio in una coda, è possibile specificare che il gestore code deve aggiungere alcune informazioni di contesto predefinite al descrittore del messaggio. Le applicazioni che dispongono del livello di autorizzazione appropriato possono aggiungere ulteriori informazioni di contesto. È possibile utilizzare il campo opzioni nella struttura MQPMO per controllare le informazioni di contesto.

Opzione MQOPEN per autorizzazione utente alternativa
Quando si tenta di aprire un oggetto utilizzando la chiamata MQOPEN, il gestore code verifica che si disponga dell'autorizzazione per aprire tale oggetto. Se non si dispone dell'autorizzazione, la chiamata ha esito negativo.

Tuttavia, i programmi server potrebbero richiedere al gestore code di controllare l'autorizzazione dell'utente per cui stanno lavorando, piuttosto che l'autorizzazione del server. Per effettuare questa operazione, è necessario utilizzare l'opzione MQOO_ALTERNATE_USER_AUTHORITY della chiamata MQOPEN e specificare l'ID utente alternativo nel campo *AlternateUserId* della struttura MQOD. Generalmente, il server ottiene l'ID utente dalle informazioni di contesto nel messaggio che sta elaborando.

L'opzione MQOPEN per la sospensione del gestore code
Nell'ambiente CICS su z/OS, se si utilizza la chiamata MQOPEN quando il gestore code è in uno stato di sospensione, la chiamata ha sempre esito negativo.

In altri ambienti z/OS , sistemi IBM i, Windows e in ambienti di sistemi UNIX and Linux , la chiamata ha esito negativo quando il gestore code è in fase di sospensione solo se si utilizza l'opzione MQOO_FAIL_IF_QUIESCING della chiamata MQOPEN.

Opzione MQOPEN per la risoluzione dei nomi delle code locali
Quando si apre una coda locale, alias o modello, viene restituita la coda locale.

Tuttavia, quando si apre una coda remota o una coda cluster, i campi *ResolvedQName* e *ResolvedQMgrName* della struttura MQOD vengono riempiti con i nomi della coda remota e del gestore code remoto trovati nella definizione della coda remota o con la coda cluster remota scelta.

Utilizzare l'opzione MQOO_RESOLVE_LOCAL_Q della chiamata MQOPEN per riempire il *ResolvedQName* nella struttura MQOD con il nome della coda locale aperta. *ResolvedQMgrName* è simile al nome del gestore code locale che ospita la coda locale. Questo campo è disponibile solo con la versione 3 della struttura MQOD; se la struttura è inferiore alla versione 3, MQOO_RESOLVE_LOCAL_Q viene ignorato senza che venga restituito un errore.

Se si specifica MQOO_RESOLVE_LOCAL_Q durante l'apertura, ad esempio, di una coda remota, *ResolvedQName* è il nome della coda di trasmissione in cui verranno inseriti i messaggi. *ResolvedQMgrName*è il nome del gestore code locale che ospita la coda di trasmissione.

Creazione di code dinamiche

Utilizzare una coda dinamica quando non è necessaria la coda dopo il termine dell'applicazione.

Ad esempio, è possibile utilizzare una coda dinamica per la coda di risposta. Specificare il nome della coda di risposta nel campo *ReplyToQ* della struttura MQMD quando si inserisce un messaggio in una coda (consultare "Definizione dei messaggi utilizzando la struttura MQMD" a pagina 227).

Per creare una coda dinamica, utilizzare un modello noto come coda modello, insieme alla chiamata MQOPEN. Creare una coda modello utilizzando i comandi WebSphere MQ o le operazioni e i pannelli di controllo. La coda dinamica creata prende gli attributi della coda modello.

Quando si richiama MQOPEN, specificare il nome della coda modello nel campo *ObjectName* della struttura MQOD. Una volta completata la chiamata, il campo *ObjectName* viene impostato sul nome della coda dinamica creata. Inoltre, il campo *ObjectQMgrName* è impostato sul nome del gestore code locale.

È possibile specificare il nome della coda dinamica creata in tre modi:

- Fornire il nome completo desiderato nel campo *DynamicQName* della struttura MQOD.
- Specificare un prefisso (meno di 33 caratteri) per il nome e consentire al gestore code di generare il resto del nome. Ciò significa che il gestore code genera un nome univoco, ma si dispone ancora di un certo controllo (ad esempio, è possibile che si desideri che ciascun utente utilizzi un determinato prefisso o che si desideri fornire una classificazione di sicurezza speciale alle code con un determinato prefisso nel proprio nome). Per utilizzare questo metodo, specificare un asterisco (*) per l'ultimo carattere non vuoto del campo *DynamicQName* . Non specificare un singolo asterisco (*) per il nome della coda dinamica.
- Consentire al gestore code di generare il nome completo. Per utilizzare questo metodo, specificare un asterisco (*) nella prima posizione del carattere del campo *DynamicQName* .

Per ulteriori informazioni su questi metodi, consultare la descrizione del campo DynamicQName .

Sono disponibili ulteriori informazioni sulle code dinamiche in Code dinamiche e modello .

Apertura di code remote

Una coda remota è una coda di proprietà di un gestore code diverso da quello a cui è connessa l'applicazione.

Per aprire una coda remota, utilizzare la chiamata MQOPEN come per una coda locale. È possibile specificare il nome della coda nel modo seguente:

1. Nel campo *ObjectName* della struttura MQOD, specificare il nome della coda remota come è noto al gestore code *locale* .

   Nota: Lasciare vuoto il campo *ObjectQMgrName* in questo caso.
2. Nel campo *ObjectName* della struttura MQOD, specificare il nome della coda remota, come noto al gestore code *remoto* . Nel campo *ObjectQMgrName* , specificare:
   - Il nome della coda di trasmissione con lo stesso nome del gestore code remoto. Il nome e il maiuscolo / minuscolo (maiuscolo, minuscolo o misto) devono corrispondere *esattamente*.
   - Il nome di un oggetto alias del gestore code che si risolve nel gestore code di destinazione o nella coda di trasmissione.

Ciò indica al gestore code la destinazione del messaggio e la coda di trasmissione su cui è necessario inserirlo per ottenerlo.

3. Se *DefXmitQname* è supportato, nel campo *ObjectName* della struttura MQOD, specificare il nome della coda remota come è noto al gestore code *remoto* .

   Nota: Impostare il campo *ObjectQMgrName* sul nome del gestore code remoto (non può essere lasciato vuoto in questo caso).

Solo i nomi locali vengono convalidati quando si richiama MQOPEN; l'ultimo controllo è relativo all'esistenza della coda di trasmissione da utilizzare.

Questi metodi sono riepilogati in Tabella 34 a pagina 218.

Chiusura di oggetti mediante la chiamata MQCLOSE

Per chiudere un oggetto, utilizzare la chiamata MQCLOSE.

Se l'oggetto è una coda, tenere presente quanto segue:

- Non è necessario svuotare una coda dinamica temporanea prima di chiuderla.

  Quando si chiude una coda dinamica temporanea, la coda viene eliminata, insieme a tutti i messaggi che potrebbero essere ancora su di essa. Questo è vero anche se ci sono chiamate MQGET, MQPUT o MQPUT1 non sottoposte a commit in sospeso rispetto alla coda.
- Su WebSphere MQ per z/OS, se si dispone di richieste MQGET con un'opzione MQGMO_SET_SIGNAL in sospeso per tale coda, queste vengono annullate.
- Se la coda è stata aperta utilizzando l'opzione MQOO_BROWSE, il cursore di ricerca viene distrutto.

La chiusura non è correlata al punto di sincronizzazione, quindi è possibile chiudere le code prima o dopo il punto di sincronizzazione.

Come input per la chiamata MQCLOSE, è necessario fornire:

- Un handle di connessione. Utilizzare lo stesso handle di collegamento utilizzato per aprirlo o, in alternativa, per le applicazioni CICS su z/OS, è possibile specificare la costante MQHC_DEF_HCONN (che ha il valore zero).
- L'handle dell'oggetto che si desidera chiudere. Acquisirlo dall'output della chiamata MQOPEN.
- MQCO_NONE nel campo *Options* (a meno che non si stia chiudendo una coda dinamica permanente).
- L'opzione di controllo per determinare se il gestore code deve eliminare la coda anche se contiene ancora messaggi (quando si chiude una coda dinamica permanente).

L'output di MQCLOSE è:

- Un codice di completamento
- Un codice di errore
- L'handle dell'oggetto, reimpostato sul valore MQHO_UNUSABLE_HOBJ

Le descrizioni dei parametri della chiamata MQCLOSE sono fornite in MQCLOSE.

Inserimento di messaggi in una coda

Utilizzare queste informazioni per informazioni su come inserire i messaggi su una coda.

Utilizzare la chiamata MQPUT per inserire i messaggi nella coda. È possibile utilizzare MQPUT ripetutamente per inserire molti messaggi nella stessa coda, seguendo la chiamata MQOPEN iniziale. Richiamare MQCLOSE una volta terminato l'inserimento di tutti i messaggi nella coda.

Se si desidera inserire un singolo messaggio in una coda e chiudere la coda immediatamente dopo, è possibile utilizzare la chiamata MQPUT1 . MQPUT1 esegue le stesse funzioni della seguente sequenza di chiamate:

- MQOPEN
- MQPUT
- MQCLOSE

In genere, tuttavia, se si dispone di più di un messaggio da inserire nella coda, è più efficiente utilizzare la chiamata MQPUT. Ciò dipende dalla dimensione del messaggio e dalla piattaforma su cui si sta lavorando.

Utilizzare i seguenti collegamenti per ulteriori informazioni sull'inserimento di messaggi in una coda:

- “Inserimento di messaggi su una coda locale utilizzando la chiamata MQPUT” a pagina 226
- “Inserimento di messaggi su una coda remota” a pagina 231
- “Impostazione delle proprietà di un messaggio” a pagina 231
- “Controllo delle informazioni di contesto” a pagina 232
- “Inserimento di un messaggio su una coda utilizzando la chiamata MQPUT1” a pagina 234
- “Liste di distribuzione” a pagina 235
- “Alcuni casi in cui le chiamate put hanno esito negativo” a pagina 240

Concetti correlati

“Panoramica su Message Queue Interface” a pagina 195
Informazioni sui componenti MQI (Message Queue Interface).

“Connessione e disconnessione da un gestore code” a pagina 207
Per utilizzare i servizi di programmazione WebSphere MQ , un programma deve disporre di una connessione a un gestore code. Utilizzare queste informazioni per informazioni su come connettersi e disconnettersi da un gestore code.

“Apertura e chiusura di oggetti” a pagina 215
Queste informazioni forniscono informazioni sull'apertura e la chiusura degli oggetti WebSphere MQ .

“Richiamo dei messaggi da una coda” a pagina 241
Utilizzare queste informazioni per ottenere messaggi da una coda.

“Richiesta di informazioni e impostazione degli attributi dell'oggetto” a pagina 322
Gli attributi sono proprietà che definiscono le caratteristiche di un oggetto WebSphere MQ .

“Commit e backout delle unità di lavoro” a pagina 325
Queste informazioni descrivono come eseguire il commit e il backout di tutte le operazioni get e put recuperabili che si sono verificate in un'unità di lavoro.

“Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger” a pagina 331
Informazioni sui trigger e su come avviare le applicazioni IBM WebSphere MQ utilizzando i trigger.

“Utilizzo di MQI e cluster” a pagina 349
Esistono opzioni speciali sulle chiamate e sui codici di ritorno relativi al cluster.

Inserimento di messaggi su una coda locale utilizzando la chiamata MQPUT

Utilizzare queste informazioni per informazioni sull'inserimento di messaggi su una coda locale utilizzando la chiamata MQPUT.

Come input per la chiamata MQPUT, è necessario fornire:

- Un handle di connessione (Hconn).
- Un gestore code (Hobj).
- Una descrizione del messaggio che si desidera inserire nella coda. È sotto forma di una struttura del descrittore del messaggio (MQMD).
- Informazioni di controllo, sotto forma di MQPMO (put - message options structure).
- La lunghezza dei dati contenuti nel messaggio (MQLONG).
- I dati del messaggio.

L'output della chiamata MQPUT è il seguente:

- Un codice motivo (MQLONG)
- Un codice di completamento (MQLONG)

Se la chiamata viene completata correttamente, restituisce anche la struttura delle opzioni e la struttura del descrittore del messaggio. La chiamata modifica la propria struttura di opzioni per visualizzare il nome della coda e il gestore code a cui è stato inviato il messaggio. Se si richiede che il gestore code generi un valore univoco per l'identificativo del messaggio che si sta immettendo (specificando zero binario nel campo *MsgId* della struttura MQMD), la chiamata inserisce il valore nel campo *MsgId* prima di restituire questa struttura all'utente. Reimposta questo valore prima di emettere un altro MQPUT.

Esiste una descrizione della chiamata MQPUT in MQPUT.

Per ulteriori informazioni sulle informazioni necessarie come input per la chiamata MQPUT, consultare i seguenti link:

- “Specifica di handle” a pagina 227
- “Definizione dei messaggi utilizzando la struttura MQMD” a pagina 227
- “Specifica delle opzioni utilizzando la struttura MQPMO” a pagina 227
- “I dati nel messaggio” a pagina 230
- “Inserimento dei messaggi: utilizzo degli handle dei messaggi” a pagina 231

Specifica di handle

Per l'handle di connessione (*Hconn*) in CICS su applicazioni z/OS , è possibile specificare la costante MQHC_DEF_HCONN (che ha il valore zero) oppure è possibile utilizzare l'handle di collegamento restituito dalla chiamata MQCONN o MQCONNX. Per altre applicazioni, utilizzare sempre l'handle di connessione restituito dalla chiamata MQCONN o MQCONNX.

Indipendentemente dall'ambiente in cui si sta lavorando, utilizzare lo stesso handle di coda (*Hobj*) restituito dalla chiamata MQOPEN.

Definizione dei messaggi utilizzando la struttura MQMD

MQMD (message descriptor structure) è un parametro di input / output per le chiamate MQPUT e MQPUT1 . Utilizzarlo per definire il messaggio che si sta inserendo in una coda.

Se MQPRI_PRIORITY_AS_Q_DEF o MQPER_PERSISTENCE_AS_Q_DEF è specificato per il messaggio e la coda è una coda cluster, i valori utilizzati sono quelli della coda in cui MQPUT si risolve. Se la coda è disabilitata per MQPUT, la chiamata avrà esito negativo. Per ulteriori informazioni, consultare Configurazione di un cluster di gestori code .

Nota: Utilizzare MQPMO_NEW_MSG_ID e MQPMO_NEW_CORREL_ID prima di inserire un nuovo messaggio per assicurarsi che *MsgId* e *CorrelId* siano univoci. I valori in questi campi vengono restituiti in un MQPUT riuscito.

È disponibile un'introduzione alle proprietà del messaggio descritte da MQMD in “IBM WebSphere MQ messaggi” a pagina 9e una descrizione della struttura stessa in MQMD.

Specifica delle opzioni utilizzando la struttura MQPMO

Utilizzare la struttura MQPMO (Put Message Option) per passare opzioni alle chiamate MQPUT e MQPUT1 .

Le seguenti sezioni forniscono assistenza per compilare i campi di questa struttura. C'è una descrizione della struttura in MQPMO.

La struttura include i seguenti campi:

- *StrucId*
- *Version*
- *Options*

- *Context*
- *ResolvedQName*
- *ResolvedQMgrName*
- *RecsPresent*
- *PutMsgRecsFields*
- *ResponseRecOffset and ResponseRecPtr*
- *OriginalMsgHandle*
- *NewMsgHandle*
- *Action*
- *PubLevel*

Il contenuto di questi campi è il seguente:

StrucId
Identifica la struttura come una struttura di opzioni put - message. Questo è un campo di 4 caratteri. Specificare sempre MQPMO_STRUC_ID.

Versione
Descrive il numero di versione della struttura. Il valore predefinito è MQPMO_VERSION_1. Se si immette MQPMO_VERSION_2, è possibile utilizzare gli elenchi di distribuzione (consultare "Liste di distribuzione" a pagina 235). Se si immette MQPMO_VERSION_3, è possibile utilizzare gli handle del messaggio e le proprietà del messaggio. Se si immette MQPMO_CURRENT_VERSION, l'applicazione è sempre impostata per utilizzare il livello più recente.

Opzioni
Ciò controlla quanto segue:

- Se l'operazione di inserimento è inclusa in un'unità di lavoro
- Quante informazioni di contesto sono associate a un messaggio
- Da dove vengono prese le informazioni di contesto
- Indica se la chiamata ha esito negativo se il gestore code è in stato di sospensione
- Se il raggruppamento o la segmentazione sono consentiti
- Creazione di un nuovo identificativo di messaggio e di correlazione
- L'ordine in cui i messaggi e i segmenti vengono inseriti in coda
- Indica se risolvere i nomi delle code locali

Se si lascia il campo *Options* impostato sul valore predefinito (MQPMO_NONE), al messaggio inserito sono associate informazioni di contesto predefinite.

Inoltre, il modo in cui la chiamata opera con i punti di sincronizzazione è determinato dalla piattaforma. Il valore predefinito del controllo del punto di sincronizzazione è yes in z/OS; per altre piattaforme, è no.

Contesto
Indica il nome dell'handle della coda da cui si desidera copiare le informazioni di contesto (se richiesto nel campo *Options*).

Per un'introduzione al contesto del messaggio, consultare "Contesto messaggio" a pagina 39. Per informazioni sull'utilizzo della struttura MQPM per controllare le informazioni di contesto in un messaggio, consultare "Controllo delle informazioni di contesto" a pagina 232.

ResolvedQName
Contiene il nome (dopo la risoluzione di qualsiasi nome alias) della coda che è stata aperta per ricevere il messaggio. Questo è un campo di output.

Nome ResolvedQMgr
Contiene il nome (dopo la risoluzione di qualsiasi nome alias) del gestore code proprietario della coda in *ResolvedQName*. Questo è un campo di output.

MQPMO può contenere anche i campi richiesti per gli elenchi di distribuzione (consultare “Liste di distribuzione” a pagina 235). Se si desidera utilizzare questa funzionalità, viene utilizzata la versione 2 della struttura MQPMO. Sono inclusi i seguenti campi:

RecsPresent
Questo campo contiene il numero di code nell'elenco di distribuzione, ossia il numero di MQPMR (Put Message Records) e MQRR (Response Records) corrispondenti.

Il valore immesso può essere lo stesso del numero di record oggetto fornito in MQOPEN. Tuttavia, se il valore è inferiore al numero di record oggetto forniti nella chiamata MQOPEN o se non si fornisce alcun record di inserimento messaggi, i valori delle code non definite vengono presi dai valori predefiniti forniti dal descrittore del messaggio. Inoltre, se il valore è maggiore del numero di record oggetto forniti, i record di messaggi immessi in eccesso vengono ignorati.

Si consiglia di effettuare una delle seguenti operazioni:

- Se si desidera ricevere un report o una risposta da ciascuna destinazione, immettere lo stesso valore visualizzato nella struttura MQOR e utilizzare MQPMR contenenti campi *MsgId* . Inizializzare questi campi *MsgId* su zeri oppure specificare MQPMO_NEW_MSG_ID.

  Una volta inserito il messaggio nella coda, i valori *MsgId* che il gestore code ha creato diventano disponibili in MQPMR; è possibile utilizzarli per identificare quale destinazione è associata a ciascun report o risposta.
- Se non si desidera ricevere report o risposte, scegliere una delle seguenti opzioni:

  1. Se si desidera identificare le destinazioni che hanno esito negativo immediatamente, è comunque possibile immettere lo stesso valore nel campo *RecsPresent* visualizzato nella struttura MQOR e fornire MQRR per identificare tali destinazioni. Non specificare alcun MQPMR.
  2. Se non si desidera identificare le destinazioni non riuscite, immettere zero nel campo *RecsPresent* e non fornire MQPMR né MQRR.

Nota: Se si sta utilizzando MQPUT1, il numero di Puntatori del record di risposta e Offset del record di risposta deve essere zero.

Per una descrizione completa di MQPMR (Put Message Records) e MQRR (Response Records), consultare MQPM e MQRR.

PutMsgRecFields
Indica quali campi sono presenti in ciascun record MQPMR (Put Message Record). Per un elenco di questi campi, consultare “Utilizzo della struttura MQPMR” a pagina 239.

PutMsgRecOffset* e *PutMsgRecPtr
I puntatori (di solito in C) e gli offset (di solito in COBOL) vengono utilizzati per indirizzare i record Put Message (vedere “Utilizzo della struttura MQPMR” a pagina 239 per una panoramica della struttura MQPMR).

Utilizzare il campo *PutMsgRecPtr* per specificare un puntatore al primo record di inserimento messaggi o il campo *PutMsgRecOffset* per specificare lo scostamento del primo record di inserimento messaggi. Questo è l'offset dall'inizio di MQPM. A seconda del campo *PutMsgRecFields* , immettere un valore non null per *PutMsgRecOffset* o *PutMsgRecPtr*.

Offset ResponseRece Ptr ResponseRec
È inoltre possibile utilizzare puntatori e offset per indirizzare i record di risposta (vedere “Utilizzo della struttura MQRR” a pagina 238 per ulteriori informazioni sui record di risposta).

Utilizzare il campo *ResponseRecPtr* per specificare un puntatore al primo record di risposta oppure il campo *ResponseRecOffset* per specificare lo scostamento del primo record di risposta. Questo è l'offset dall'inizio della struttura MQPMO. Immettere un valore non null per *ResponseRecOffset* o *ResponseRecPtr*.

Nota: Se si utilizza MQPUT1 per inserire i messaggi in un elenco di distribuzione, *ResponseRecPtr* deve essere null o zero e *ResponseRecOffset* deve essere zero.

La Versione 3 della struttura di MQPMO include anche i seguenti campi:

Handle OriginalMsg
L'utilizzo di questo campo dipende dal valore del campo *Azione* . Se si sta inserendo un nuovo messaggio con le proprietà del messaggio associate, impostare questo campo sull'handle del messaggio precedentemente creato e impostare le proprietà su. Se si sta inoltrando, rispondendo o generando un report in risposta a un messaggio precedentemente richiamato, questo campo contiene l'handle del messaggio di tale messaggio.

NewMsgHandle
Se si specifica un *NewMsgHandle*, tutte le proprietà associate all'handle sovrascrivono le proprietà associate all' *OriginalMsgHandle*. Per ulteriori informazioni, consultare Azione (MQLONG).

Azione
Utilizzare questo campo per specificare il tipo di immissione da eseguire. I valori possibili e i relativi significati sono i seguenti:

NUOVO
Questo è un nuovo messaggio non correlato ad altri.

FORWARD MQACTP
Questo messaggio è stato richiamato in precedenza ed è ora in fase di inoltro.

MQACTP_REPLY
Questo messaggio è una risposta ad un messaggio precedentemente richiamato.

PROSPETTO MQACTP_REPORT
Questo messaggio è un report generato come risultato di un messaggio precedentemente richiamato.

Per ulteriori informazioni, consultare Azione (MQLONG).

PubLevel
Se questo messaggio è una pubblicazione, è possibile impostare questo campo per determinare quali sottoscrizioni lo ricevono. Solo le sottoscrizioni con un *SubLevel* inferiore o uguale a questo valore riceveranno questa pubblicazione. Il valore predefinito è 9, che è il livello più alto e significa che le sottoscrizioni con qualsiasi *SubLevel* possono ricevere questa pubblicazione.

I dati nel messaggio

Fornire l'indirizzo del buffer che contiene i dati nel parametro *Buffer* della chiamata MQPUT. È possibile includere qualsiasi cosa nei dati nei messaggi. La quantità di dati nei messaggi, tuttavia, influenza le prestazioni dell'applicazione che li sta elaborando.

La dimensione massima dei dati è determinata da:

- L'attributo *MaxMsgLength* del gestore code
- L'attributo *MaxMsgLength* della coda in cui si sta inserendo il messaggio
- La dimensione di qualsiasi intestazione di messaggio aggiunta da WebSphere MQ (incluse l'intestazione del messaggio non instradabile, MQDLH e l'intestazione dell'elenco di distribuzione, MQDH)

L'attributo *MaxMsgLength* del gestore code contiene la dimensione del messaggio che il gestore code può elaborare. Questo ha un valore predefinito di 100 MB per tutti i prodotti WebSphere MQ alla versione V6 o superiore.

Per determinare il valore di questo attributo, utilizzare la chiamata MQINQ sull'oggetto gestore code. Per messaggi di grandi dimensioni, è possibile modificare questo valore.

L'attributo *MaxMsgLength* di una coda determina la dimensione massima del messaggio che è possibile inserire sulla coda. Se si tenta di inserire un messaggio con una dimensione maggiore del valore di questo attributo, la chiamata MQPUT ha esito negativo. Se si sta inserendo un messaggio su una coda remota, la dimensione massima del messaggio che è possibile inserire correttamente è determinata dall'attributo *MaxMsgLength* della coda remota, da tutte le code di trasmissione intermedie su cui il messaggio viene inserito lungo l'instradamento alla relativa destinazione e dai canali utilizzati.

Per un'operazione MQPUT, la dimensione del messaggio deve essere inferiore o uguale all'attributo *MaxMsgLength* della coda e del gestore code. I valori di questi attributi sono indipendenti, ma si consiglia di impostare il *MaxMsgLength* della coda su un valore inferiore o uguale a quello del gestore code.

WebSphere MQ aggiunge le informazioni di intestazione ai messaggi nelle seguenti circostanze:

- Quando si inserisce un messaggio su una coda remota, WebSphere MQ aggiunge una struttura di intestazione di trasmissione (MQXQH) al messaggio. Questa struttura comprende il nome della coda di destinazione e il suo gestore code proprietario.
- Se WebSphere MQ non è in grado di consegnare un messaggio a una coda remota, tenta di inserire il messaggio nella coda dei messaggi non recapitabili (messaggi non recapitati). Aggiunge una struttura MQDLH al messaggio. Questa struttura comprende il nome della coda di destinazione e il motivo per cui il messaggio è stato inserito nella coda di messaggi non instradabili.
- Se si desidera inviare un messaggio a più code di destinazione, WebSphere MQ aggiunge un'intestazione MQDH al messaggio. Descrive i dati presenti in un messaggio, appartenenti a un elenco di distribuzione, su una coda di trasmissione. Considerare questa opzione quando si sceglie un valore ottimale per la lunghezza massima del messaggio.
- Se il messaggio è un segmento o un messaggio in un gruppo, WebSphere MQ potrebbe aggiungere un MQMDE.

Queste strutture sono descritte in MQDH e MQMDE.

Se i messaggi sono della dimensione massima consentita per queste code, l'aggiunta di queste intestazioni significa che le operazioni di inserimento hanno esito negativo perché i messaggi sono ora troppo grandi. Per ridurre la possibilità che le operazioni di inserimento non funzionino:

- Ridurre la dimensione dei messaggi rispetto all'attributo *MaxMsgLength* delle code di trasmissione e di messaggi non recapitabili. Consentire almeno il valore della costante MQ_MSG_HEADER_LENGTH (più per elenchi di distribuzione di grandi dimensioni).
- Assicurarsi che l'attributo *MaxMsgLength* della coda di messaggi non instradabili sia impostato sullo stesso valore di *MaxMsgLength* del gestore code proprietario della coda di messaggi non instradabili.

Gli attributi per il gestore code e le costanti di accodamento dei messaggi sono descritti in Attributi per il gestore code.

Inserimento dei messaggi: utilizzo degli handle dei messaggi

Sono disponibili due handle del messaggio nella struttura MQPM, *OriginalMsgHandle* e *NewMsgHandle*. La relazione tra questi handle del messaggio viene definita dal valore del campo *Azione* MQPMO.

Per dettagli completi, consultare Azione (MQLONG). Un handle del messaggio non è necessariamente richiesto per inserire un messaggio. Il suo scopo è quello di associare le proprietà a un messaggio, quindi è richiesto solo se si utilizzano le proprietà del messaggio.

Inserimento di messaggi su una coda remota

Quando si desidera inserire un messaggio su una coda remota (ossia, una coda di un gestore code diverso da quello a cui è connessa l'applicazione) piuttosto che su una coda locale, l'unica considerazione aggiuntiva è il modo in cui si specifica il nome della coda quando viene aperta.Ciò è descritto in “Apertura di code remote” a pagina 224. Non vi sono modifiche al modo in cui si utilizza la chiamata MQPUT o MQPUT1 per una coda locale.

Per ulteriori informazioni sull'utilizzo delle code remote e di trasmissione, consultare WebSphere MQ distributed - messaging techniques .

Impostazione delle proprietà di un messaggio

Richiamare MQSETMP per ogni proprietà che si desidera impostare. Quando si inserisce la serie di messaggi, l'handle del messaggio e i relativi campi di azione della struttura MQPMO.

Per associare le proprietà ad un messaggio, il messaggio deve disporre di un handle del messaggio. Creare un handle del messaggio utilizzando la chiamata alla funzione MQCRTMH. Richiamare MQSETMP specificando questo handle del messaggio per ciascuna proprietà che si desidera impostare. Un programma di esempio, amqsstma.c, viene fornito per illustrare l'utilizzo di MQSETMP.

Se si tratta di un nuovo messaggio, quando lo si inserisce in una coda, utilizzando MQPUT o MQPUT1, impostare il campo Handle OriginalMsgin MQPMO sul valore di questo handle del messaggio e impostare il campo Azione MQPMO su MQACTP_NEW (questo è il valore predefinito).

Se si tratta di un messaggio richiamato in precedenza e si sta ora inoltrando o rispondendo ad esso o inviando un report in risposta ad esso, inserire l'handle del messaggio originale nel campo Handle OriginalMsgdi MQPMO e il nuovo handle del messaggio nel campo Handle NewMsg. Impostare il campo Azione su MQACTP_FORWARD, MQACTP_REPLY o MQACTP_REPORT, come appropriato.

Se si dispone di proprietà in un'intestazione MQRFH2 da un messaggio precedentemente richiamato, è possibile convertirle in proprietà di gestione messaggi utilizzando la chiamata MQBUFMH.

Se si sta inserendo il messaggio in una coda su un gestore code a un livello precedente a WebSphere MQ Versione 7.0, che non può elaborare le proprietà del messaggio, è possibile impostare il parametro PropertyControl nella definizione del canale per specificare il modo in cui devono essere trattate le proprietà.

Controllo delle informazioni di contesto

Quando si utilizza la chiamata MQPUT o MQPUT1 per inserire un messaggio in una coda, è possibile specificare che il gestore code deve aggiungere alcune informazioni di contesto predefinite al descrittore del messaggio. Le applicazioni che dispongono del livello di autorizzazione appropriato possono aggiungere ulteriori informazioni di contesto. È possibile utilizzare il campo opzioni nella struttura MQPMO per controllare le informazioni di contesto.

Per controllare le informazioni di contesto, utilizzare il campo *Options* nella struttura MQPMO.

In caso contrario, il gestore code sovrascrive le informazioni di contesto che potrebbero già essere presenti nel descrittore del messaggio con le informazioni di identità e di contesto generate per il messaggio. Ciò equivale a specificare l'opzione MQPMO_DEFAULT_CONTEXT.Si potrebbero desiderare queste informazioni di contesto predefinite quando si crea un nuovo messaggio (ad esempio, quando si elabora l'input dell'utente da una schermata di interrogazione).

Se non si desidera che le informazioni di contesto siano associate al messaggio, utilizzare l'opzione MQPMO_NO_CONTEXT. Quando si inserisce un messaggio senza contesto, i controlli di autorizzazione eseguiti da IBM WebSphere MQ vengono eseguiti utilizzando un ID utente vuoto. Un ID utente vuoto non può essere assegnato all'autorizzazione esplicita per le risorse IBM WebSphere MQ , ma viene considerato come membro del gruppo speciale 'nobody'. Per ulteriori dettagli sul gruppo speciale nobody, consultare Installable services interface reference information.

Se non si desidera che le informazioni di contesto siano associate al messaggio, utilizzare l'opzione MQPMO_NO_CONTEXT.

Le seguenti sezioni di questo argomento spiegano l'uso del contesto identità, del contesto utente e di tutto il contesto.

- “Passaggio contesto identità” a pagina 233
- “Passaggio del contesto utente” a pagina 233
- “Passaggio di tutti i contesti” a pagina 233
- “Impostazione contesto identità” a pagina 233
- “Impostazione contesto utente” a pagina 234
- “Impostazione di tutti i contesti” a pagina 234

Passaggio contesto identità

In generale, i programmi devono trasmettere le informazioni sul contesto di identità da messaggio a messaggio intorno a una applicazione fino a quando i dati non raggiungono la destinazione finale.

I programmi devono modificare le informazioni sul contesto di origine ogni volta che modificano i dati. Tuttavia, le applicazioni che desiderano modificare o impostare le informazioni di contesto devono disporre del livello di autorizzazione appropriato. Il gestore code controlla questa autorizzazione quando le applicazioni aprono le code; devono disporre dell'autorizzazione per utilizzare le opzioni di contesto appropriate per la chiamata MQOPEN.

Se l'applicazione riceve un messaggio, elabora i dati dal messaggio, quindi inserisce i dati modificati in un altro messaggio (possibilmente per l'elaborazione da parte di un'altra applicazione), l'applicazione deve passare le informazioni del contesto di identità dal messaggio originale al nuovo messaggio. È possibile consentire al gestore code di creare le informazioni sul contesto di origine.

Per salvare le informazioni di contesto dal messaggio originale, utilizzare l'opzione MQOO_SAVE_ALL_CONTEXT quando si apre la coda per ottenere il messaggio. Questo è in aggiunta a tutte le altre opzioni che si utilizzano con la chiamata MQOPEN. Notare, tuttavia, che non è possibile salvare le informazioni di contesto se si sfoglia solo il messaggio.

Quando si crea il secondo messaggio:

- Aprire la coda utilizzando l'opzione MQOO_PASS_IDENTITY_CONTEXT (in aggiunta all'opzione MQOO_OUTPUT).
- Nel campo *Context* della struttura di opzioni di inserimento del messaggio, fornire l'handle della coda da cui sono state salvate le informazioni di contesto.
- Nel campo *Options* della struttura delle opzioni put - message, specificare l'opzione MQPMO_PASS_IDENTITY_CONTEXT.

Passaggio del contesto utente

Non è possibile scegliere di passare solo il contesto utente. Per passare il contesto utente durante l'inserimento di un messaggio, specificare MQPMO_PASS_ALL_CONTEXT. Tutte le proprietà nel contesto utente vengono trasmesse nello stesso modo del contesto di origine.

Quando si verifica un MQPUT o MQPUT1 e il contesto viene passato, tutte le proprietà nel contesto utente vengono trasmesse dal messaggio richiamato al messaggio di inserimento. Tutte le proprietà del contesto utente che l'applicazione di inserimento ha modificato vengono inserite con i relativi valori originali. Tutte le proprietà del contesto utente che l'applicazione di inserimento ha eliminato vengono ripristinate nel messaggio di inserimento. Tutte le proprietà del contesto utente che l'applicazione di inserimento ha aggiunto al messaggio vengono conservate.

Passaggio di tutti i contesti

Se l'applicazione riceve un messaggio e inserisce i dati del messaggio (non modificati) in un altro messaggio, l'applicazione deve trasmettere tutte le informazioni di contesto (identità, origine e utente) dal messaggio originale al nuovo messaggio. Un esempio di un'applicazione che potrebbe eseguire questa operazione è un programma di spostamento messaggi, che sposta i messaggi da una coda all'altra.

Seguire la stessa procedura utilizzata per il trasferimento del contesto di identità, ad eccezione del fatto che si utilizzano le opzioni MQOPEN MQOO_PASS_ALL_CONTEXT e put - message MQPMO_PASS_ALL_CONTEXT.

Impostazione contesto identità

Se si desidera impostare le informazioni sul contesto di identità per un messaggio:

- Aprire la coda utilizzando l'opzione MQOO_SET_IDENTITY_CONTEXT.
- Inserire il messaggio nella coda, specificando l'opzione MQPMO_SET_IDENTITY_CONTEXT. Nel descrittore del messaggio, specificare le informazioni di contesto di identità richieste.

Nota: Quando si impostano alcuni (ma non tutti) dei campi del contesto di identità utilizzando le opzioni MQOO_SET_IDENTITY_CONTEXT e MQPMO_SET_IDENTITY_CONTEXT, è importante rendersi conto che il gestore code non imposta nessuno degli altri campi.

Per modificare le opzioni di contesto del messaggio, è necessario disporre delle autorizzazioni appropriate per emettere la chiamata. Ad esempio, per utilizzare MQOO_SET_IDENTITY_CONTEXT o MQPMO_SET_IDENTITY_CONTEXT, è necessario disporre dell'autorizzazione `+setid` .

Impostazione contesto utente

Per impostare una proprietà nel contesto utente, impostare il campo Contesto di MQPD (message property descriptor) su MQPD_USER_CONTEXT quando si effettua la chiamata MQSETMP.

Non è necessaria alcuna autorizzazione speciale per impostare una proprietà nel contesto utente. Il contesto utente non ha opzioni di contesto MQOO_SET_ * o MQPMO_SET_ *.

Impostazione di tutti i contesti

Se si desidera impostare sia l'identità che le informazioni sul contesto di origine per un messaggio:

1. Aprire la coda utilizzando l'opzione MQOO_SET_ALL_CONTEXT.
2. Inserire il messaggio nella coda, specificando l'opzione MQPMO_SET_ALL_CONTEXT. Nel descrittore del messaggio, specificare le informazioni di identità e contesto di origine richieste.

Per ogni tipo di impostazione di contesto è necessaria l'autorizzazione appropriata.

Concetti correlati

“Contesto messaggio” a pagina 39
Le informazioni relative al *contesto del messaggio* consentono all'applicazione che richiama il messaggio di individuare il creatore del messaggio.

Riferimenti correlati

“Opzioni MQOPEN relative al contesto del messaggio” a pagina 223
Se si desidera poter associare le informazioni di contesto a un messaggio quando le si inserisce in una coda, è necessario utilizzare una delle opzioni di contesto del messaggio quando si apre la coda.

Inserimento di un messaggio su una coda utilizzando la chiamata MQPUT1

Utilizzare la chiamata MQPUT1 quando si desidera chiudere la coda immediatamente dopo aver inserito un messaggio singolo. Ad esempio, un'applicazione server probabilmente utilizzerà la chiamata MQPUT1 quando invia una risposta a ognuna delle diverse code.

MQPUT1 è funzionalmente equivalente alla chiamata MQOPEN seguita da MQPUT, seguita da MQCLOSE. L'unica differenza nella sintassi delle chiamate MQPUT e MQPUT1 è che per MQPUT si specifica un handle dell'oggetto, mentre per MQPUT1 si specifica una struttura del descrittore dell'oggetto (MQOD) come definito in MQOPEN (consultare “Identificazione degli oggetti (la struttura MQOD)” a pagina 217). Ciò è dovuto al fatto che è necessario fornire informazioni alla chiamata MQPUT1 sulla coda che deve aprire, mentre quando si richiama MQPUT, la coda deve essere già aperta.

Come input per la chiamata MQPUT1 , è necessario fornire:

- Un handle di connessione.
- Una descrizione dell'oggetto che si desidera aprire. È sotto forma di MQOD (object descriptor structure).
- Una descrizione del messaggio che si desidera inserire nella coda. È sotto forma di una struttura del descrittore del messaggio (MQMD).
- Controllare le informazioni sotto forma di MQPMO (put - message options structure).
- La lunghezza dei dati contenuti nel messaggio (MQLONG).
- L'indirizzo dei dati del messaggio.

L'output da MQPUT1 è:

- Un codice di completamento
- Un codice di errore

Se la chiamata viene completata correttamente, restituisce anche la struttura delle opzioni e la struttura del descrittore del messaggio. La chiamata modifica la propria struttura di opzioni per visualizzare il nome della coda e il gestore code a cui è stato inviato il messaggio. Se si richiede che il gestore code generi un valore univoco per l'identificativo del messaggio che si sta immettendo (specificando zero binario nel campo *MsgId* della struttura MQMD), la chiamata inserisce il valore nel campo *MsgId* prima di restituire questa struttura all'utente.

Nota: Non è possibile utilizzare MQPUT1 con un nome coda modello; tuttavia, una volta aperta una coda modello, è possibile immettere MQPUT1 per la coda dinamica.

I sei parametri di input per MQPUT1 sono:

Hconn
Questo è un handle di connessione. Per le applicazioni CICS , è possibile specificare la costante MQHC_DEF_HCONN (che ha il valore zero) o utilizzare l'handle di connessione restituito dalla chiamata MQCONN o MQCONNX. Per altri programmi, utilizzare sempre l'handle di connessione restituito dalla chiamata MQCONN o MQCONNX.

ObjDesc
Questa è una struttura descrittore oggetto (MQOD).

Nei campi *ObjectName* e *ObjectQMgrName* , specificare il nome della coda in cui si desidera inserire un messaggio e il nome del gestore code proprietario di questa coda.

Il campo *DynamicQName* viene ignorato per la chiamata MQPUT1 perché non può utilizzare le code modello.

Utilizzare il campo *AlternateUserId* se si desidera denominare un identificativo utente alternativo da utilizzare per verificare l'autorizzazione ad aprire la coda.

MsgDesc
Questa è una struttura del descrittore del messaggio (MQMD). Come con la chiamata MQPUT, utilizzare questa struttura per definire il messaggio che si sta inserendo sulla coda.

PutMsgOpts
Si tratta di una struttura di opzioni di inserimento messaggi (MQPMO). Utilizzarlo come per la chiamata MQPUT (consultare “Specifica delle opzioni utilizzando la struttura MQPMO” a pagina 227).

Quando il campo *Options* è impostato a 0, il gestore code utilizza il proprio ID utente quando esegue verifiche per l'autorizzazione ad accedere alla coda. Inoltre, il gestore code ignora qualsiasi identificativo utente alternativo fornito nel campo *AlternateUserId* della struttura MQOD.

BufferLength
Questa è la lunghezza del messaggio.

Buffer
Questa è l'area di buffer che contiene il testo del messaggio.

Quando si utilizzano i cluster, MQPUT1 funziona come se MQOO_BIND_NOT_FIXED fosse attivo. Le applicazioni devono utilizzare i campi risolti nella struttura MQPMO piuttosto che la struttura MQOD per determinare dove è stato inviato il messaggio. Per ulteriori informazioni, consultare Configurazione di un cluster di gestori code .

Esiste una descrizione della chiamata MQPUT1 in MQPUT1.

Liste di distribuzione

Non supportato su WebSphere MQ per z/OS. Gli elenchi di distribuzione consentono di inserire un messaggio in più destinazioni in una singola chiamata MQPUT o MQPUT1 .Una singola chiamata MQOPEN può aprire più code e una singola chiamata MQPUT può quindi inserire un messaggio in ognuna di queste code. Alcune informazioni generiche dalle strutture MQI utilizzate per questo processo possono essere sostituite da informazioni specifiche relative alle singole destinazioni incluse nell'elenco di distribuzione.

V 7.5.0.8

Attenzione: Gli elenchi di distribuzione non supportano l'utilizzo di code alias che puntano agli oggetti argomento. Da Version 7.5.0, Fix Pack 8, se una coda alias punta a un oggetto argomento in un elenco di distribuzione, IBM WebSphere MQ restituisce MQRC_ALIAS_BASE_Q_TYPE_ERROR.

Quando viene emessa una chiamata MQOPEN, le informazioni generiche vengono prese da MQOD (Object Descriptor). Se si specifica MQOD_VERSION_2 nel campo *Version* e un valore maggiore di zero nel campo *RecsPresent* , *Hobj* può essere definito come un handle di un elenco (di una o più code) piuttosto che di una coda. In questo caso, le informazioni specifiche vengono fornite tramite i record oggetto (MQORs), che forniscono i dettagli della destinazione (ovvero, *ObjectName* e *ObjectQMgrName*).

L'handle dell'oggetto (*Hobj*) viene passato alla chiamata MQPUT, consentendo di inserire in un elenco piuttosto che in una coda singola.

Quando un messaggio viene inserito nelle code (MQPUT), le informazioni generiche vengono prese dalla struttura MQPMO (Put Message Option) e MQMD (Message Descriptor). Le informazioni specifiche vengono fornite sotto forma di MQPMR (Put Message Records).

I record di risposta (MQRR) possono ricevere un codice di completamento e un codice motivo specifici per ogni coda di destinazione.

La Figura 29 a pagina 236 mostra il funzionamento degli elenchi di distribuzione.

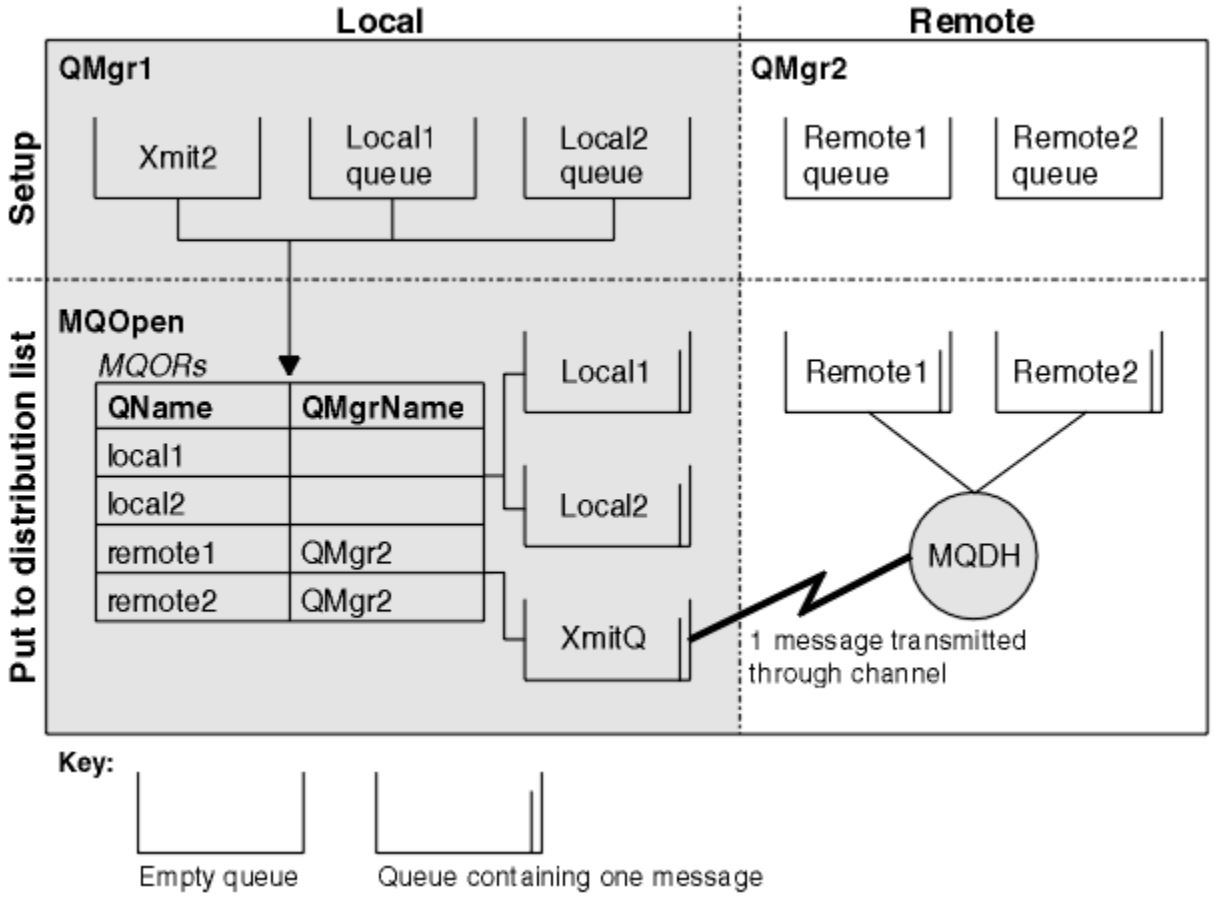


Figura 29. Funzionamento degli elenchi di distribuzione

Apertura degli elenchi di distribuzione

Utilizzare la chiamata MQOPEN per aprire un elenco di distribuzione e utilizzare le opzioni della chiamata per specificare cosa si desidera fare con l'elenco.

Come input per MQOPEN, è necessario fornire:

- Un handle di connessione (consultare “Inserimento di messaggi in una coda” a pagina 225 per una descrizione)
- Informazioni generiche in MQOD (Object Descriptor structure)
- Il nome di ciascuna coda che si desidera aprire, utilizzando MQOR (Object Record structure)

L'output di MQOPEN è:

- Una gestione oggetto che rappresenta l'accesso dell'utente all'elenco di distribuzione
- Un codice di completamento generico
- Un codice di errore generico
- Record di risposta (facoltativo), che contengono un codice di completamento e il motivo per ciascuna destinazione

Utilizzo della struttura MQOD

Utilizzare la struttura MQOD per identificare le code che si desidera aprire.

Per definire un elenco di distribuzione, è necessario specificare MQOD_VERSION_2 nel campo *Version* , un valore maggiore di zero nel campo *RecsPresent* e MQOT_Q nel campo *ObjectType* . Consultare MQOD per una descrizione di tutti i campi della struttura MQOD.

Utilizzo della struttura MQOR

Fornire una struttura MQOR per ogni destinazione.

La struttura contiene i nomi della coda di destinazione e del gestore code. I campi *ObjectName* e *ObjectQMgrName* in MQOD non vengono utilizzati per gli elenchi di distribuzione. Ci devono essere uno o più record oggetto. Se il *ObjectQMgrName* viene lasciato vuoto, viene utilizzato il gestore code locale. Per ulteriori informazioni su questi campi, consultare ObjectName e ObjectQMgrName .

È possibile specificare le code di destinazione in due modi:

- Utilizzando il campo offset *ObjectRecOffset*.

  In questo caso, l'applicazione deve dichiarare la sua struttura contenente una struttura MQOD, seguita dall'array di record MQOR (con tutti gli elementi dell'array necessari) e impostare *ObjectRecOffset* sull'offset del primo elemento dell'array dall'inizio dell'MQOD. Verificare che questo offset sia corretto.

  Si consiglia di utilizzare le funzioni integrate fornite dal linguaggio di programmazione, se sono disponibili in tutti gli ambienti in cui viene eseguita l'applicazione. Il seguente codice illustra questa tecnica per il linguaggio di programmazione 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.
```

  In alternativa, utilizzare la costante MQOD_CURRENT_LENGTH se il linguaggio di programmazione non supporta le funzioni integrate necessarie in tutti gli ambienti interessati. Il seguente codice illustra questa tecnica:

```
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.
```

Tuttavia, ciò funziona correttamente solo se la struttura MQOD e l'array dei record MQOR sono contigui; se il compilatore inserisce byte ignorati tra l'MQOD e l'array MQOR, questi devono essere aggiunti al valore memorizzato in *ObjectRecOffset*.

L'utilizzo di *ObjectRecOffset* è consigliato per i linguaggi di programmazione che non supportano il tipo di dati puntatore o che implementano il tipo di dati puntatore in un modo non portabile in ambienti differenti (ad esempio, il linguaggio di programmazione COBOL).

- Utilizzando il campo puntatore *ObjectRecPtr*.

  In questo caso, l'applicazione può dichiarare l'array delle strutture MQOR separatamente dalla struttura MQOD e impostare *ObjectRecPtr* sull'indirizzo dell'array. Il seguente codice illustra questa tecnica per il linguaggio di programmazione C:

```
MQOD MyMqod;
MQOR MyMqor[100];
MyMqod.ObjectRecPtr = MyMqor;
```

  L'uso di *ObjectRecPtr* è consigliato per i linguaggi di programmazione che supportano il tipo di dati del puntatore in un modo portabile in ambienti diversi (ad esempio, il linguaggio di programmazione C).

Indipendentemente dalla tecnica scelta, è necessario utilizzare uno tra *ObjectRecOffset* e *ObjectRecPtr*; la chiamata ha esito negativo con codice motivo MQRC_OBJECT_RECORDS_ERROR se entrambi sono zero o entrambi sono diversi da zero.

Utilizzo della struttura MQRR

Queste strutture sono specifiche della destinazione; ogni record di risposta contiene un campo *CompCode* e *Reason* per ogni coda di un elenco di distribuzione. È necessario utilizzare questa struttura per consentire di distinguere i problemi.

Ad esempio, se si riceve un codice di errore MQRC_MULTIPLE_REASON e l'elenco di distribuzione contiene cinque code di destinazione, non si sa a quali code si applicano i problemi se non si utilizza questa struttura. Tuttavia, se si dispone di un codice di completamento e di un codice motivo per ogni destinazione, è possibile individuare gli errori più facilmente.

Consultare MQRR per ulteriori informazioni sulla struttura MQRR.

Figura 30 a pagina 238 mostra come aprire un elenco di distribuzione in C.

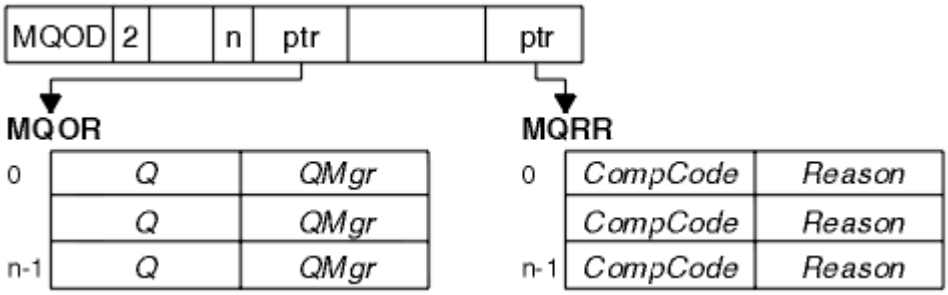


Figura 30. Apertura di un elenco di distribuzione in C

Figura 31 a pagina 238 mostra come è possibile aprire un elenco di distribuzione in COBOL.

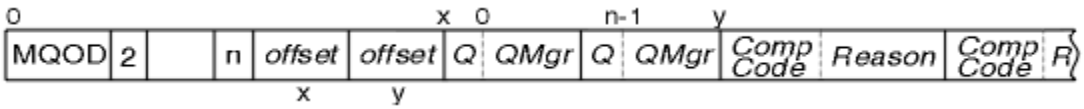


Figura 31. Apertura di un elenco di distribuzione in COBOL

Utilizzo delle opzioni MQOPEN

È possibile specificare le opzioni seguenti quando si apre un elenco di distribuzioni:

- OUTPUT MQOO
- MQOO_FAIL_IF_QUIESCING (facoltativo)
- MQOO_ALTERNATE_USER_AUTHORITY (facoltativo)
- MQOO_ * _CONTEXT (facoltativo)

Consultare “Apertura e chiusura di oggetti” a pagina 215 per una descrizione di queste opzioni.

Inserimento di messaggi in un elenco di distribuzione

Per inserire i messaggi in un elenco di distribuzione, è possibile utilizzare MQPUT o MQPUT1.

Come input, è necessario fornire:

- Un handle di connessione (consultare “Inserimento di messaggi in una coda” a pagina 225 per una descrizione).
- Un handle di oggetto. Se un elenco di distribuzione viene aperto utilizzando MQOPEN, *Hobj* consente solo di inserirlo nell'elenco.
- Una struttura descrittore del messaggio (MQMD). Consultare MQMD per una descrizione di questa struttura.
- Informazioni di controllo nel formato di una struttura di opzioni di inserimento messaggi (MQPMO). Consultare “Specifica delle opzioni utilizzando la struttura MQPMO” a pagina 227 per informazioni sul completamento dei campi della struttura MQPMO.
- Informazioni di controllo nel formato MQPMR (Put Message Records).
- La lunghezza dei dati contenuti nel messaggio (MQLONG).
- I dati del messaggio.

L'output è:

- Un codice di completamento
- Un codice di errore
- Record di risposta (facoltativo)

Utilizzo della struttura MQPMR

Questa struttura è facoltativa e fornisce informazioni specifiche di destinazione per alcuni campi che è possibile identificare in modo diverso da quelli già identificati in MQMD.

Per una descrizione di questi campi, consultare MQPMR.

Il contenuto di ciascun record dipende dalle informazioni fornite nel campo *PutMsgRecFields* di MQPM. Ad esempio, nel programma di esempio AMQSPTL0.C (consultare “Programma di esempio Elenco di distribuzione” a pagina 128 per una descrizione) che mostra l'utilizzo degli elenchi di distribuzione, l'esempio sceglie di fornire valori per *MsgId* e *CorrelId* in MQPMR. Questa sezione del programma di esempio è simile alla seguente:

```
typedef struct
{
MQBYTE24 MsgId;
MQBYTE24 CorrelId;
} PutMsgRec;
...
/**********************
MQLONG PutMsgRecFields=MQPMRF_MSG_ID | MQPMRF_CORREL_ID;
```

Ciò implica che *MsgId* e *CorrelId* vengono forniti per ciascuna destinazione di un elenco di distribuzione. I record di inserimento messaggio vengono forniti come un array.

Figura 32 a pagina 240 mostra come è possibile inserire un messaggio in un elenco di distribuzione in C.

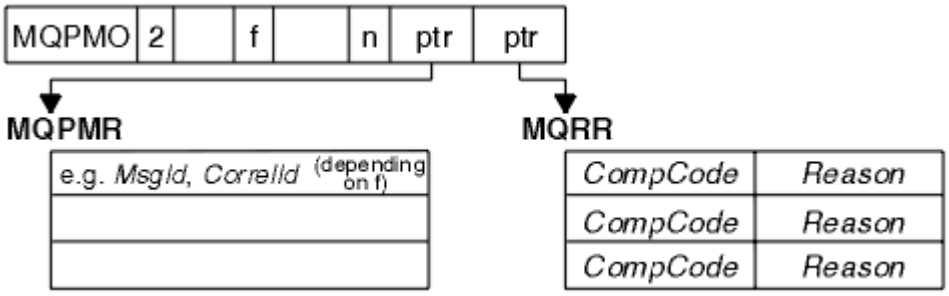


Figura 32. Inserimento di un messaggio in un elenco di distribuzione in C

Figura 33 a pagina 240 mostra come è possibile inserire un messaggio in un elenco di distribuzione in COBOL.

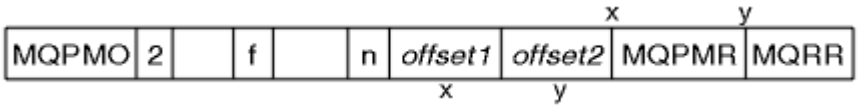


Figura 33. Inserimento di un messaggio in un elenco di distribuzione in COBOL

Utilizzo di MQPUT1

Se si sta utilizzando MQPUT1, considerare i seguenti punti:

1. I valori dei campi *ResponseRecOffset* e *ResponseRecPtr* devono essere null o zero.
2. I record di risposta, se richiesti, devono essere indirizzati da MQOD.

Alcuni casi in cui le chiamate put hanno esito negativo

Se alcuni attributi di una coda vengono modificati utilizzando l'opzione FORCE su un comando durante l'intervallo tra l'emissione di una chiamata MQOPEN e MQPUT, la chiamata MQPUT ha esito negativo e restituisce il codice motivo MQRC_OBJECT_CHANGED.

Il gestore code contrassegna l'handle dell'oggetto come non più valido. Ciò si verifica anche se le modifiche vengono effettuate durante l'elaborazione di una chiamata MQPUT1 o se le modifiche si applicano a qualsiasi coda in cui si risolve il nome della coda. Gli attributi che influenzano l'handle in questo modo sono riportati nella descrizione della chiamata MQOPEN in MQOPEN. Se la chiamata restituisce il codice motivo MQRC_OBJECT_CHANGED, chiudere la coda, riaprirla, quindi provare a inserire nuovamente un messaggio.

Se le operazioni di inserimento sono inibite per una coda in cui si sta tentando di inserire i messaggi (o per qualsiasi coda in cui il nome della coda si risolve), la chiamata MQPUT o MQPUT1 ha esito negativo e restituisce il codice di errore MQRC_PUT_INIITED. Potrebbe essere possibile inserire un messaggio correttamente se si tenta la chiamata in un secondo momento, se la progettazione dell'applicazione è tale che altri programmi modificano regolarmente gli attributi delle code.

Inoltre, se la coda in cui si sta tentando di inserire il messaggio è piena, la chiamata MQPUT o MQPUT1 ha esito negativo e restituisce MQRC_Q_FULL.

Se una coda dinamica (temporanea o permanente) è stata eliminata, le chiamate MQPUT che utilizzano un handle di oggetto precedentemente acquisito hanno esito negativo e restituiscono il codice motivo MQRC_Q_DELETED. In questa situazione, si consiglia di chiudere l'handle dell'oggetto poiché non è più utile.

Nel caso di elenchi di distribuzione, più codici di completamento e codici motivo possono verificarsi in una singola richiesta. Non possono essere gestiti utilizzando solo i campi di output *CompCode* e *Reason* su MQOPEN e MQPUT.

Quando si utilizzano gli elenchi di distribuzione per inserire i messaggi in più destinazioni, i record di risposta contengono gli specifici *CompCode* e *Reason* per ciascuna destinazione. Se si riceve un codice di completamento di MQCC_FAILED, nessun messaggio viene inserito correttamente in

una coda di destinazione. Se il codice di completamento è MQCC_WARNING, il messaggio viene inserito correttamente su una o più code di destinazione. Se si riceve un codice di ritorno MQRC_MULTIPLE_REASON, i codici di errore non sono tutti uguali per ogni destinazione. Si consiglia pertanto di utilizzare la struttura MQRR in modo da poter determinare la coda o le code che hanno causato un errore e le relative cause.

Richiamo dei messaggi da una coda

Utilizzare queste informazioni per ottenere messaggi da una coda.

È possibile richiamare i messaggi da una coda in due modi:

1. È possibile eliminare un messaggio dalla coda in modo che altri programmi non possano più visualizzarlo.
2. È possibile copiare un messaggio, lasciando il messaggio originale sulla coda. Questa operazione è nota come *esplorazione*. È possibile rimuovere il messaggio una volta esplorato.

In entrambi i casi, si utilizza la chiamata MQGET, ma prima l'applicazione deve essere connessa al gestore code e si deve utilizzare la chiamata MQOPEN per aprire la coda (per l'input, la ricerca o entrambi). Queste operazioni sono descritte in “Connessione e disconnessione da un gestore code” a pagina 207 e “Apertura e chiusura di oggetti” a pagina 215.

Una volta aperta la coda, è possibile utilizzare la chiamata MQGET ripetutamente per sfogliare o rimuovere i messaggi sulla stessa coda. Richiamare MQCLOSE una volta terminato di richiamare tutti i messaggi desiderati dalla coda.

Utilizzare i seguenti collegamenti per ulteriori informazioni sul richiamo dei messaggi da una coda:

- “Come ottenere i messaggi da una coda utilizzando la chiamata MQGET” a pagina 242
- “L'ordine in cui i messaggi vengono richiamati da una coda” a pagina 246
- “Ricezione di un messaggio particolare” a pagina 257
- “Miglioramento delle prestazioni dei messaggi non persistenti” a pagina 259
- “Gestione dei messaggi con lunghezza superiore a 4 MB” a pagina 263
- “In attesa di messaggi” a pagina 269
-
- “Backout ignorato” a pagina 269
- “Conversione dati applicazione” a pagina 272
- “Visualizzazione dei messaggi su una coda” a pagina 273
- “Alcuni casi in cui la chiamata MQGET non riesce” a pagina 279

Concetti correlati

“Panoramica su Message Queue Interface” a pagina 195
Informazioni sui componenti MQI (Message Queue Interface).

“Connessione e disconnessione da un gestore code” a pagina 207
Per utilizzare i servizi di programmazione WebSphere MQ , un programma deve disporre di una connessione a un gestore code. Utilizzare queste informazioni per informazioni su come connettersi e disconnettersi da un gestore code.

“Apertura e chiusura di oggetti” a pagina 215
Queste informazioni forniscono informazioni sull'apertura e la chiusura degli oggetti WebSphere MQ .

“Inserimento di messaggi in una coda” a pagina 225
Utilizzare queste informazioni per informazioni su come inserire i messaggi su una coda.

“Richiesta di informazioni e impostazione degli attributi dell'oggetto” a pagina 322
Gli attributi sono proprietà che definiscono le caratteristiche di un oggetto WebSphere MQ .

“Commit e backout delle unità di lavoro” a pagina 325

Queste informazioni descrivono come eseguire il commit e il backout di tutte le operazioni get e put recuperabili che si sono verificate in un'unità di lavoro.

“Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger” a pagina 331
Informazioni sui trigger e su come avviare le applicazioni IBM WebSphere MQ utilizzando i trigger.

“Utilizzo di MQI e cluster” a pagina 349
Esistono opzioni speciali sulle chiamate e sui codici di ritorno relativi al cluster.

Come ottenere i messaggi da una coda utilizzando la chiamata MQGET

La chiamata MQGET richiama un messaggio da una coda locale aperta. Non può richiamare un messaggio da una coda su un altro sistema.

Come input per la chiamata MQGET, è necessario fornire:

- Un handle di connessione.
- Un handle di coda.
- Una descrizione del messaggio che si desidera ottenere dalla coda. Si tratta di una struttura MQMD (Message Descriptor).
- Controllare le informazioni sotto forma di una struttura MQGMO (Get Message Options).
- La dimensione del buffer che è stato assegnato per conservare il messaggio (MQLONG).
- L'indirizzo della memoria in cui inserire il messaggio.

L'output di MQGET è:

- Un codice di errore
- Un codice di completamento
- Il messaggio nell'area di buffer specificato, se la chiamata viene completata correttamente
- La struttura delle opzioni, modificata per mostrare il nome della coda da cui è stato richiamato il messaggio
- La struttura del descrittore del messaggio, con il contenuto dei campi modificati per descrivere il messaggio richiamato
- La lunghezza del messaggio (MQLONG)

C'è una descrizione della chiamata MQGET in MQGET.

Le seguenti sezioni descrivono le informazioni che è necessario fornire come input alla chiamata MQGET.

- “Specifica degli handle di connessione” a pagina 242
- “Descrizione dei messaggi utilizzando la struttura di MQMD e la chiamata MQGET” a pagina 242
- “Specifica delle opzioni MQGET utilizzando la struttura MQGMO” a pagina 243
- “Specifica della dimensione dell'area buffer” a pagina 245

Specifica degli handle di connessione

Per applicazioni CICS su z/OS , è possibile specificare la costante MQHC_DEF_HCONN (con valore zero) oppure utilizzare l'handle di connessione restituito dalla chiamata MQCONN o MQCONNX. Per altre applicazioni, utilizzare sempre l'handle di connessione restituito dalla chiamata MQCONN o MQCONNX.

Utilizzare il gestore code (*Hobj*) restituito quando si richiama MQOPEN.

Descrizione dei messaggi utilizzando la struttura di MQMD e la chiamata MQGET

Per identificare il messaggio che si desidera ottenere da una coda, utilizzare la struttura del descrittore del messaggio (MQMD).

Si tratta di un parametro di input / output per la chiamata MQGET. È disponibile un'introduzione alle proprietà del messaggio descritte da MQMD in “IBM WebSphere MQ messaggi” a pagina 9e una descrizione della struttura stessa in MQMD.

Se si conosce il messaggio che si desidera richiamare dalla coda, consultare “Ricezione di un messaggio particolare” a pagina 257.

Se non si specifica un particolare messaggio, MQGET richiama il *primo* messaggio nella coda. “L'ordine in cui i messaggi vengono richiamati da una coda” a pagina 246 descrive in che modo la priorità di un messaggio, l'attributo *MsgDeliverySequence* della coda e l'opzione MQGMO_LOGICAL_ORDER determinano l'ordine dei messaggi nella coda.

Nota: Se si desidera utilizzare MQGET più di una volta (ad esempio, per esaminare i messaggi nella coda), è necessario impostare i campi *MsgId* e *CorrelId* di questa struttura su null dopo ogni chiamata. Questo elimina questi campi degli identificatori del messaggio che è stato richiamato.

Tuttavia, se si desidera raggruppare i messaggi, il *GroupId* deve essere lo stesso per i messaggi nello stesso gruppo, in modo che la chiamata ricerca un messaggio con gli stessi identificativi del messaggio precedente per costituire l'intero gruppo.

Specifica delle opzioni MQGET utilizzando la struttura MQGMO

La struttura MQGMO è una variabile di input / output per passare le opzioni alla chiamata MQGET. Le seguenti sezioni consentono di completare alcuni dei campi di questa struttura.

È presente una descrizione della struttura MQGMO in MQGMO.

StrucId
StrucId è un campo di 4 caratteri utilizzato per identificare la struttura come una struttura di opzioni get - message. Specificare sempre MQGMO_STRUC_ID.

Version
*Version*descrive il numero di versione della struttura. MQGMO_VERSION_1 è il valore predefinito. Se si desidera utilizzare i campi della Versione 2 o richiamare i messaggi in ordine logico, specificare MQGMO_VERSION_2. Se si desidera utilizzare i campi della Versione 3 o richiamare i messaggi in ordine logico, specificare MQGMO_VERSION_3. MQGMO_CURRENT_VERSION imposta l'applicazione per utilizzare il livello più recente.

Options
All'interno del codice, è possibile selezionare le opzioni in qualsiasi ordine; ogni opzione è rappresentata da un bit nel campo *Options* .

Il campo *Options* controlla:

- Se la chiamata MQGET attende l'arrivo di un messaggio sulla coda prima del completamento (consultare “In attesa di messaggi” a pagina 269)
- Se l'operazione di acquisizione è inclusa in un'unità di lavoro.
- Se un messaggio non persistente viene richiamato al di fuori del punto di sincronizzazione, consentendo la messaggistica rapida
- Su WebSphere MQ per z/OS, se il messaggio richiamato è contrassegnato come backout ignorato (consultare “Backout ignorato” a pagina 269)
- Se il messaggio viene rimosso dalla coda o semplicemente esplorato
- Indica se selezionare un messaggio utilizzando un cursore di ricerca o altri criteri di selezione
- Se la chiamata ha esito positivo anche se il messaggio è più lungo del buffer
- Su WebSphere MQ per z/OS, indica se consentire il completamento della chiamata. Questa opzione imposta anche un segnale per indicare che si desidera ricevere una notifica quando arriva un messaggio
- Indica se la chiamata ha esito negativo se il gestore code è in stato di sospensione
- Su WebSphere MQ per z/OS, se la chiamata ha esito negativo se la connessione è in uno stato di sospensione

- Se è richiesta la conversione dei dati dei messaggi dell'applicazione (consultare "Conversione dati applicazione" a pagina 272)
- L'ordine in cui i messaggi e i segmenti (tranne per WebSphere MQ per z/OS) vengono richiamati da una coda
- Tranne in WebSphere MQ per z/OS, se i messaggi logici completi sono richiamabili
- Se i messaggi in un gruppo possono essere richiamati solo quando *tutti* i messaggi nel gruppo sono disponibili
- Ad eccezione di WebSphere MQ per z/OS, se i segmenti in un messaggio logico possono essere richiamati solo quando sono disponibili *tutti* i segmenti nel messaggio logico

Se si lascia il campo *Options* impostato sul valore predefinito (MQGMO_NO_WAIT), la chiamata MQGET funziona in questo modo:

- Se non c'è alcun messaggio che corrisponde ai criteri di selezione sulla coda, la chiamata non attende l'arrivo di un messaggio, ma viene completata immediatamente. Inoltre, in WebSphere MQ per z/OS, la chiamata non imposta un segnale che richiede la notifica quando arriva tale messaggio.
- Il modo in cui la chiamata opera con i punti di sincronizzazione è determinato dalla piattaforma:

Piattaforma	**Sotto il controllo del punto di sincronizzazione**
IBM i	No
Sistemi UNIX and Linux	No
z/OS	Sì
Sistemi Windows	No

- In WebSphere MQ per z/OS, il messaggio richiamato non è contrassegnato come backout ignorato.
- Il messaggio selezionato viene rimosso dalla coda (non esplorato).
- Non è richiesta alcuna conversione dei dati del messaggio dell'applicazione.
- La chiamata ha esito negativo se il messaggio è più lungo del buffer.

WaitInterval
Il campo *WaitInterval* specifica il tempo massimo (in millesimi di secondo) che la chiamata MQGET attende per l'arrivo di un messaggio sulla coda quando si utilizza l'opzione MQGMO_WAIT. Se nessun messaggio arriva entro il periodo di tempo specificato in *WaitInterval*, la chiamata viene completata e restituisce un codice motivo che indica che non c'era alcun messaggio che corrispondesse ai criteri di selezione sulla coda.

Su WebSphere MQ per z/OS, se si utilizza l'opzione MQGMO_SET_SIGNAL, il campo *WaitInterval* specifica l'ora per cui è impostato il segnale.

Per ulteriori informazioni su queste opzioni, consultare "In attesa di messaggi" a pagina 269.

Signal1
Signal1 è supportato solo su WebSphere MQ per z/OS e MQSeries per HP Integrity NonStop Server.

Se si utilizza l'opzione MQGMO_SET_SIGNAL per richiedere che l'applicazione venga avvisata quando arriva un messaggio adatto, specificare il tipo di segnale nel campo *Signal1* . In WebSphere MQ su tutte le altre piattaforme, il campo *Signal1* è riservato e il suo valore non è significativo.

Signal2
Il campo *Signal2* è riservato su tutte le piattaforme e il suo valore non è significativo.

ResolvedQName
ResolvedQName è un campo di output in cui il gestore code restituisce il nome della coda (dopo la risoluzione di qualsiasi alias) da cui è stato richiamato il messaggio.

MatchOptions
MatchOptions controlla i criteri di selezione per MQGET.

GroupStatus
GroupStatus indica se il messaggio richiamato si trova in un gruppo.

SegmentStatus
SegmentStatus indica se l'elemento richiamato è un segmento di un messaggio logico.

Segmentation
Segmentation indica se la segmentazione è consentita per il messaggio richiamato.

MsgToken

MsgToken identifica in modo univoco un messaggio.

ReturnedLength
ReturnedLength è un campo di output in cui il gestore code restituisce la lunghezza dei dati del messaggio restituiti (in byte).

MsgHandle
L'handle per un messaggio che deve essere popolato con le proprietà del messaggio richiamato dalla coda. L'handle è stato precedentemente creato da una chiamata MQCRTMH. Tutte le proprietà già associate all'handle vengono eliminate prima di richiamare un messaggio.

Specifica della dimensione dell'area buffer

Nel parametro *BufferLength* della chiamata MQGET, specificare le dimensioni dell'area di buffer per contenere i dati del messaggio che si richiamano. Si decide quanto grande dovrebbe essere in tre modi:

1. È possibile che si conosca già la lunghezza dei messaggi da prevedere da questo programma. In tal caso, specificare un buffer di questa dimensione.

   Tuttavia, è possibile utilizzare l'opzione MQGMO_ACCEPT_TRUNCATED_MSG nella struttura MQGMO se si desidera che la chiamata MQGET venga completata anche se il messaggio è troppo grande per il buffer. In questo caso:

   - Il buffer è riempito con la maggior parte del messaggio che può contenere
   - La chiamata restituisce un codice di completamento di avvertenza
   - Il messaggio viene rimosso dalla coda (eliminando il resto del messaggio) oppure il cursore di ricerca è avanzato (se si sta sfogliando la coda)
   - La lunghezza reale del messaggio viene restituita in *DataLength*

   Senza questa opzione, la chiamata viene ancora completata con un'avvertenza, ma non rimuove il messaggio dalla coda (o non fa avanzare il cursore di ricerca).
2. Stimare una dimensione per il buffer (o anche specificare una dimensione di zero byte) e *non* utilizzare l'opzione MQGMO_ACCEPT_TRUNCATED_MSG. Se la chiamata MQGET ha esito negativo (ad esempio, perché il buffer è troppo piccolo), la lunghezza del messaggio viene restituita nel parametro *DataLength* della chiamata. (Il buffer è ancora riempito con la quantità di messaggi che può contenere, ma l'elaborazione della chiamata non viene completata.) Memorizzare il *MsgId* di questo messaggio, quindi ripetere la chiamata MQGET, specificando un'area buffer della dimensione corretta e il *MsgId* annotato dalla prima chiamata.

   Se il programma sta servendo una coda che è servita anche da altri programmi, uno di questi programmi potrebbe rimuovere il messaggio che si desidera prima che il programma possa emettere un'altra chiamata MQGET. Il programma potrebbe perdere tempo nella ricerca di un messaggio che non esiste più. Per evitare ciò, sfogliare prima la coda fino a trovare il messaggio desiderato, specificando un *BufferLength* di zero e utilizzando l'opzione MQGMO_ACCEPT_TRUNCATED_MSG. Questo posiziona il cursore di ricerca sotto il messaggio desiderato. È quindi possibile richiamare il messaggio richiamando nuovamente MQGET, specificando l'opzione MQGMO_MSG_UNDER_CURSOR. Se un altro programma rimuove il messaggio tra le chiamate di ricerca e rimozione, il secondo MQGET

non riesce immediatamente (senza ricercare l'intera coda), perché non c'è alcun messaggio sotto il cursore di ricerca.

3. L'attributo *MaxMsgLength* Coda determina la lunghezza massima dei messaggi accettati per tale coda; l'attributo *MaxMsgLength* Gestore code determina la lunghezza massima dei messaggi accettati per tale gestore code. Se non si conosce la lunghezza del messaggio prevista, è possibile richiedere informazioni sull'attributo *MaxMsgLength* (utilizzando la chiamata MQINQ), quindi specificare un buffer di questa dimensione.

   Provare a rendere la dimensione del buffer il più possibile vicina alla dimensione reale del messaggio per evitare prestazioni ridotte.

   Per ulteriori informazioni sull'attributo *MaxMsgLength* , consultare "Aumento della lunghezza massima del messaggio" a pagina 263.

L'ordine in cui i messaggi vengono richiamati da una coda

È possibile controllare l'ordine in cui richiamare i messaggi da una coda. Questa sezione esamina le opzioni.

Priorit...

Un programma può assegnare una priorità a un messaggio quando inserisce il messaggio su una coda (consultare "Priorità dei messaggi" a pagina 17). I messaggi di uguale priorità vengono memorizzati in una coda in ordine di arrivo, non in ordine di commit.

Il gestore code gestisce le code nella sequenza FIFO (first in, first out) o in FIFO all'interno della sequenza di priorità. Ciò dipende dall'impostazione dell'attributo *MsgDeliverySequence* della coda.Quando un messaggio arriva su una coda, viene inserito immediatamente dopo l'ultimo messaggio con la stessa priorità.

I programmi possono richiamare il primo messaggio da una coda oppure possono richiamare un determinato messaggio da una coda, ignorando la priorità di tali messaggi. Ad esempio, un programma potrebbe voler elaborare la risposta a un particolare messaggio inviato in precedenza. Per ulteriori informazioni, vedere "Ricezione di un messaggio particolare" a pagina 257.

Se un'applicazione inserisce una sequenza di messaggi in una coda, un'altra applicazione può richiamare tali messaggi nello stesso ordine in cui sono stati inseriti, a condizione che:

- I messaggi hanno tutti la stessa priorità
- I messaggi sono stati tutti inseriti all'interno della stessa unità di lavoro o all'esterno di un'unità di lavoro
- La coda è locale per l'applicazione di inserimento

Se queste condizioni non vengono soddisfatte e le applicazioni dipendono dai messaggi richiamati in un certo ordine, le applicazioni devono includere le informazioni sulla sequenza nei dati del messaggio o stabilire un mezzo per confermare la ricezione di un messaggio prima che venga inviato il messaggio successivo.

Ordinamento logico e fisico

I messaggi sulle code possono verificarsi (all'interno di ogni livello di priorità) in ordine *fisico* o *logico* .

L'ordine fisico è l'ordine in cui i messaggi arrivano su una coda. L'ordine logico è quando tutti i messaggi e i segmenti all'interno di un gruppo si trovano nella loro sequenza logica, uno accanto all'altro, nella posizione determinata dalla posizione fisica del primo elemento appartenente al gruppo.

Per una descrizione di gruppi, messaggi e segmenti, consultare "Gruppi di messaggi" a pagina 36. Questi ordini fisici e logici possono essere diversi perché:

- I gruppi possono arrivare a una destinazione in momenti simili da applicazioni diverse, perdendo quindi qualsiasi ordine fisico distinto.
- Anche all'interno di un singolo gruppo, i messaggi possono essere disordinati a causa del reinstradamento o del ritardo di alcuni dei messaggi nel gruppo.

Ad esempio, l'ordine logico potrebbe essere simile alla figura Figura 34 a pagina 247:

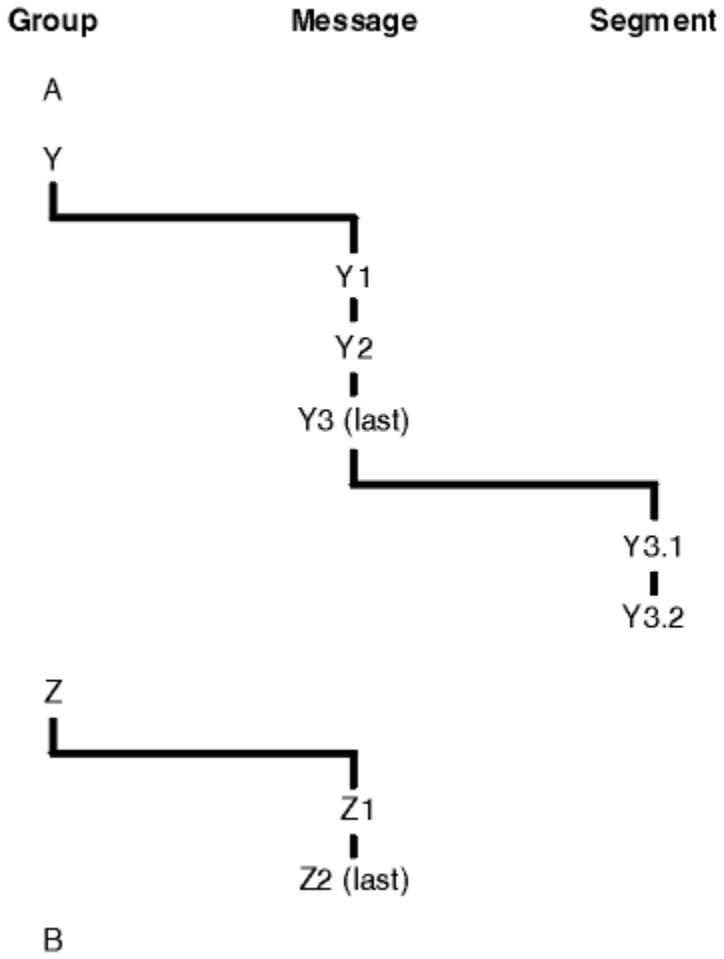


Figura 34. Ordine logico su una coda

Questi messaggi si verificano nel seguente ordine logico su una coda:

1. Messaggio A (non in un gruppo)
2. Messaggio logico 1 del gruppo Y
3. Messaggio logico 2 del gruppo Y
4. Segmento 1 di (ultimo) messaggio logico 3 del gruppo Y
5. (Ultimo) segmento 2 del (ultimo) messaggio logico 3 del gruppo Y
6. Messaggio logico 1 del gruppo Z
7. (Ultimo) messaggio logico 2 del gruppo Z
8. Messaggio B (non in un gruppo)

L'ordine fisico, tuttavia, potrebbe essere completamente diverso. La posizione fisica del *primo* elemento all'interno di ciascun gruppo determina la posizione logica dell'intero gruppo. Ad esempio, se i gruppi Y e Z sono arrivati in tempi simili e il messaggio 2 del gruppo Z ha superato il messaggio 1 dello stesso gruppo, l'ordine fisico sarà simile alla figura Figura 35 a pagina 248:

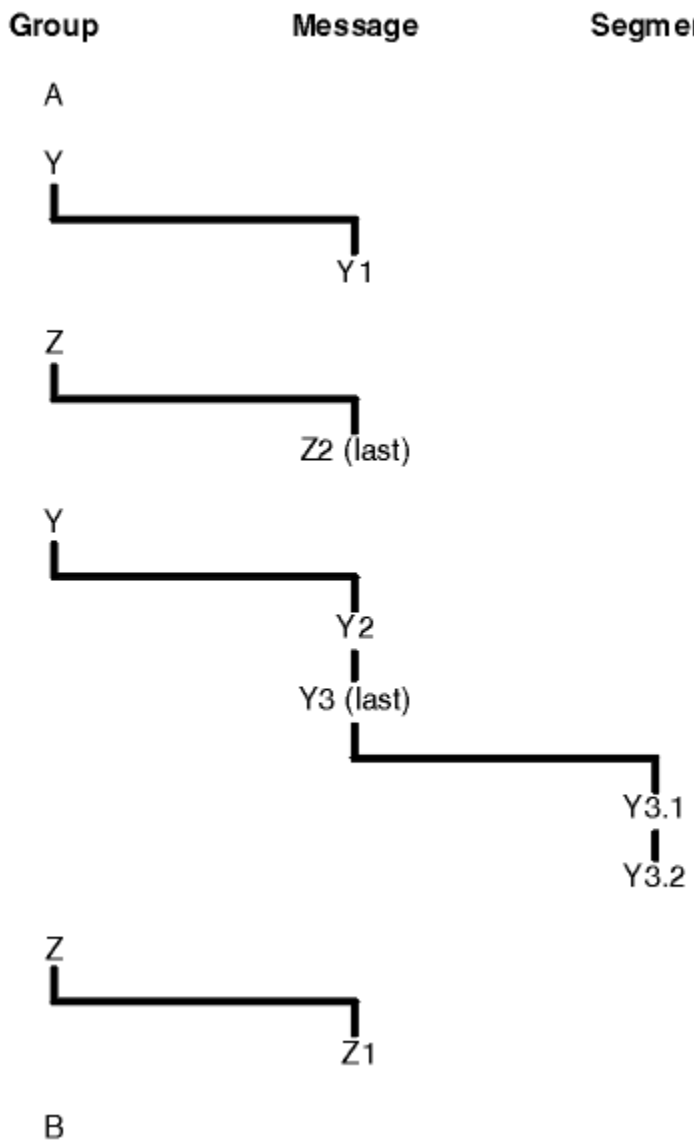


Figura 35. Ordine fisico su una coda

Questi messaggi si verificano nel seguente ordine fisico sulla coda:

1. Messaggio A (non in un gruppo)
2. Messaggio logico 1 del gruppo Y
3. Messaggio logico 2 del gruppo Z
4. Messaggio logico 2 del gruppo Y
5. Segmento 1 di (ultimo) messaggio logico 3 del gruppo Y
6. (Ultimo) segmento 2 del (ultimo) messaggio logico 3 del gruppo Y
7. Messaggio logico 1 del gruppo Z
8. Messaggio B (non in un gruppo)

Nota: Su IBM WebSphere MQ for z/OS, l'ordine fisico dei messaggi sulla coda non è garantito se la coda è indicizzata da GROUPID.

Quando si ricevono i messaggi, è possibile specificare MQGMO_LOGICAL_ORDER per richiamare i messaggi in ordine logico piuttosto che in ordine fisico.

Se si immette una chiamata MQGET con MQGMO_BROWSE_FIRST e MQGMO_LOGICAL_ORDER, le successive chiamate MQGET con MQGMO_BROWSE_NEXT devono specificare anche

MQGMO_LOGICAL_ORDER. Al contrario, se MQGET con MQGMO_BROWSE_FIRST non specifica MQGMO_LOGICAL_ORDER, né le seguenti MQGET con MQGMO_BROWSE_NEXT.

Le informazioni sul gruppo e sul segmento che il gestore code conserva per le chiamate MQGET che sfogliano i messaggi sulla coda sono separate dal gruppo e le informazioni sul segmento che il gestore code conserva per le chiamate MQGET che rimuovono i messaggi dalla coda. Quando si specifica MQGMO_BROWSE_FIRST, il gestore code ignora le informazioni sul gruppo e sul segmento per la ricerca e esegue la scansione della coda come se non vi fossero gruppi correnti e messaggi logici correnti.

Nota: non utilizzare una chiamata MQGET per esplorare *oltre la fine* di un gruppo di messaggi (o di un messaggio logico non presente in un gruppo) senza specificare MQGMO_LOGICAL_ORDER. Ad esempio, se l'ultimo messaggio nel gruppo *precede* il primo messaggio nel gruppo sulla coda, utilizzando MQGMO_BROWSE_NEXT per esplorare oltre la fine del gruppo, specificando MQMO_MATCH_MSG_SEQ_NUMBER con *MsgSeqNumber* impostato su 1 (per trovare il primo messaggio del gruppo successivo) restituisce nuovamente il primo messaggio nel gruppo già esplorato. Ciò potrebbe verificarsi immediatamente o in un numero di chiamate MQGET successive (se sono presenti gruppi che intervengono).

Evitare la possibilità di un loop infinito aprendo la coda *due volte* per la ricerca:

- Utilizzare il primo handle per ricercare solo il primo messaggio in ciascun gruppo.
- Utilizzare il secondo handle per ricercare solo i messaggi all'interno di un determinato gruppo.
- Utilizzare le opzioni MQMO_ * per spostare il secondo cursore di ricerca nella posizione del primo cursore di ricerca, prima di esaminare i messaggi nel gruppo.
- Non utilizzare la ricerca MQGMO_BROWSE_NEXT oltre la fine di un gruppo.

Per ulteriori informazioni su questo argomento, consultare MQGET, MQMDe Regole per la convalida delle opzioni MQI.

Per la maggior parte delle applicazioni si sceglierà probabilmente l'ordinamento logico o fisico durante la navigazione. Tuttavia, se si desidera passare da una modalità all'altra, tenere presente che quando si emette per la prima volta una ricerca con MQGMO_LOGICAL_ORDER, viene stabilita la propria posizione all'interno della sequenza logica.

Se il primo elemento all'interno del gruppo non è presente in questo momento, il gruppo in cui ci si trova non viene considerato come parte della sequenza logica.

Una volta che il cursore di ricerca si trova all'interno di un gruppo, è possibile continuare all'interno dello stesso gruppo, anche se il primo messaggio viene rimosso. Inizialmente, tuttavia, non è possibile spostarsi in un gruppo utilizzando MQGMO_LOGICAL_ORDER in cui il primo elemento non è presente.

ORDER MQPMO_LOGICAL_

L'opzione MQPMO indica al gestore code in che modo l'applicazione inserisce i messaggi in gruppi e segmenti di messaggi logici. Può essere specificato solo nella chiamata MQPUT; non è valido nella chiamata MQPUT1 .

Se MQPMO_LOGICAL_ORDER è specificato, indica che l'applicazione utilizza le chiamate MQPUT successive per:

1. Inserire i segmenti in ciascun segmento logico nell'ordine di offset crescente dei segmenti, a partire da 0, senza intervalli.
2. Inserire tutti i segmenti in un unico messaggio logico prima di inserirli nel successivo messaggio logico.
3. Inserire i messaggi logici in ciascun gruppo di messaggi nell'ordine crescente del numero di sequenza dei messaggi, a partire da 1, senza intervalli.IBM WebSphere MQ incrementa automaticamente il numero di sequenza del messaggio.
4. Inserire tutti i messaggi logici in un unico gruppo di messaggi prima di inserirli nel successivo gruppo di messaggi.

Poiché l'applicazione ha indicato al gestore code il modo in cui inserisce i messaggi in gruppi e segmenti di messaggi logici, non è necessario che l'applicazione mantenga e aggiorni le informazioni sul gruppo e sul segmento relative a ciascuna chiamata MQPUT, poiché il gestore code conserva e

aggiorna queste informazioni. In particolare, significa che l'applicazione non deve impostare i campi *GroupId*, *MsgSeqNumbe*r e *Offset* in MQMD, poiché il gestore code imposta questi campi sui valori appropriati. L'applicazione deve impostare solo il campo *MsgFlags* in MQMD, per indicare quando i messaggi appartengono a gruppi o sono segmenti di messaggi logici e per indicare l'ultimo messaggio in un gruppo o l'ultimo segmento di un messaggio logico.

Una volta avviato un gruppo di messaggi o un messaggio logico, le successive chiamate MQPUT devono specificare gli indicatori MQMF_ * appropriati in *MsgFlags* in MQMD. Se l'applicazione tenta di inserire un messaggio che non si trova in un gruppo quando è presente un gruppo di messaggi non terminato o un messaggio che non è un segmento quando è presente un messaggio logico non terminato, la chiamata non riesce con il codice di errore MQRC_INCOMPLETE_GROUP o MQRC_INCOMPLETE_MSG, come appropriato. Tuttavia, il gestore code conserva le informazioni sul gruppo di messaggi corrente o sul messaggio logico corrente e l'applicazione può terminarli inviando un messaggio (possibilmente senza dati del messaggio dell'applicazione) specificando MQMF_LAST_MSG_IN_GROUP o MQMF_LAST_SEGMENT come appropriato, prima di emettere nuovamente la chiamata MQPUT per inserire il messaggio che non si trova nel gruppo o non in un segmento.

Figura 35 a pagina 248 mostra le combinazioni di opzioni e indicatori validi e i valori dei campi *GroupId*, *MsgSeqNumbe*r e *Offset* utilizzati dal gestore code in ciascun caso. Le combinazioni di opzioni e indicatori non mostrate nella tabella non sono valide. Le colonne nella tabella hanno i seguenti significati; significa Sì o No:

ORD LOG
Se l'opzione MQPMO_LOGICAL_ORDER è specificata sulla chiamata.

MIG
Se l'opzione MQMF_MSG_IN_GROUP o MQMF_LAST_MSG_IN_GROUP è specificata nella chiamata.

SEG
Se l'opzione MQMF_SEGMENT o MQMF_LAST_SEGMENT è specificata nella chiamata.

SEG OK
Indica se l'opzione MQMF_SEGMENTATION_ALLOWED è specificata nella chiamata.

Grp Cur
Se esiste un gruppo di messaggi corrente prima della chiamata.

Messaggio log Cur
Se esiste un messaggio logico corrente prima della chiamata.

Altre colonne
Mostra i valori utilizzati dal gestore code. Precedente indica il valore utilizzato per il campo nel precedente messaggio per la gestione della coda.

Tabella 36. Opzioni MQPUT relative a messaggi in gruppi e segmenti di messaggi logici

Opzioni specificate				**Stato messaggio di log e gruppo prima della chiamata**		**Valori utilizzati dal gestore code**		
ORD LOG	**MIG**	**SEG**	**SEG - OK**	**Grp corrente**	**Messaggio di log corrente**	***GroupId***	***MsgSeqNumber***	***Offset***
Sì	No	No	No	No	No	MQGI_NONE	1	0
Sì	No	No	Sì	No	No	Nuovo ID gruppo	1	0

Tabella 36. Opzioni MQPUT relative a messaggi in gruppi e segmenti di messaggi logici (Continua)

Opzioni specificate				**Stato messaggio di log e gruppo prima della chiamata**		**Valori utilizzati dal gestore code**		
ORD LOG	**MIG**	**SEG**	**SEG - OK**	**Grp corrente**	**Messaggio di log corrente**	***GroupId***	***MsgSeqNumber***	***Offset***
Sì	No	Sì	Entrambi	No	No	Nuovo ID gruppo	1	0
Sì	No	Sì	Entrambi	No	Sì	ID gruppo precedente	1	Offset precedente + lunghezza segmento precedente
Sì	Sì	Entrambi	Entrambi	No	No	Nuovo ID gruppo	1	0
Sì	Sì	Entrambi	Entrambi	Sì	No	ID gruppo precedente	Numero di sequenza precedente + 1	0
Sì	Sì	Sì	Entrambi	Sì	Sì	ID gruppo precedente	Numero sequenza precedente	Offset precedente + lunghezza segmento precedente
No	No	No	No	Entrambi	Entrambi	MQGI_NONE	1	0
No	No	No	Sì	Entrambi	Entrambi	Nuovo ID gruppo se MQGI_NONE, altrimenti valore nel campo	1	0
No	No	Sì	Entrambi	Entrambi	Entrambi	Nuovo ID gruppo se MQGI_NONE, altrimenti valore nel campo	1	Valore nel campo
No	Sì	No	Entrambi	Entrambi	Entrambi	Nuovo ID gruppo se MQGI_NONE, altrimenti valore nel campo	Valore nel campo	0
No	Sì	Sì	Entrambi	Entrambi	Entrambi	Nuovo ID gruppo se MQGI_NONE, altrimenti valore nel campo	Valore nel campo	Valore nel campo

Tabella 36. Opzioni MQPUT relative a messaggi in gruppi e segmenti di messaggi logici (Continua)

Opzioni specificate				Stato messaggio di log e gruppo prima della chiamata		Valori utilizzati dal gestore code		
ORD LOG	**MIG**	**SEG**	**SEG - OK**	**Grp corrente**	**Messaggio di log corrente**	***GroupId***	***MsgSeqNumber***	***Offset***

Nota:

- MQPMO_LOGICAL_ORDER non è valido nella chiamata MQPUT1 .
- Per il campo *MsgId* , il gestore code genera un nuovo identificativo del messaggio se MQPMO_NEW_MSG_ID o MQMI_NONE è specificato e utilizza il valore nel campo in caso contrario.
- Per il campo *CorrelId* , il gestore code genera un nuovo identificativo di correlazione se viene specificato MQPMO_NEW_CORREL_ID e utilizza il valore nel campo in caso contrario.

Quando si specifica MQPMO_LOGICAL_ORDER, il gestore code richiede che tutti i messaggi in un gruppo e i segmenti in un messaggio logico vengano inseriti con lo stesso valore nel campo *Persistence* in MQMD, ossia tutti devono essere persistenti o tutti devono essere non persistenti. Se questa condizione non viene soddisfatta, la chiamata MQPUT ha esito negativo con codice motivo MQRC_INCONSISTENT_PERSISTENCE.

L'opzione MQPMO_LOGICAL_ORDER influisce sulle unità di lavoro come segue:

- Se il primo messaggio fisico in un gruppo o in un messaggio logico viene inserito all'interno di un'unità di lavoro, tutti gli altri messaggi fisici nel gruppo o nel messaggio logico devono essere inseriti all'interno di un'unità di lavoro, se viene utilizzato lo stesso gestore code. Tuttavia, non è necessario inserirli all'interno della stessa unità di lavoro, consentendo a un gruppo di messaggi o a un messaggio logico costituito da molti messaggi fisici di essere suddiviso in due o più unità di lavoro consecutive per l'handle della coda.
- Se il primo messaggio fisico in un gruppo o un messaggio logico non viene inserito all'interno di un'unità di lavoro, nessuno degli altri messaggi fisici nel gruppo o messaggio logico può essere inserito all'interno di un'unità di lavoro, se viene utilizzato lo stesso gestore code.

Se queste condizioni non vengono soddisfatte, la chiamata MQPUT ha esito negativo con codice motivo MQRC_INCONSISTENT_UOW.

Quando viene specificato MQPMO_LOGICAL_ORDER, l'MQMD fornito nella chiamata MQPUT non deve essere minore di MQMD_VERSION_2. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice motivo MQRC_WRONG_MD_VERSION.

Se MQPMO_LOGICAL_ORDER non è specificato, i messaggi in gruppi e segmenti di messaggi logici possono essere inseriti in qualsiasi ordine e non è necessario inserire gruppi di messaggi completi o messaggi logici completi. È responsabilità dell'applicazione assicurarsi che i campi *GroupId*, *MsgSeqNumber*, *Offset*e *MsgFlags* abbiano valori appropriati.

Utilizzare questa tecnica per riavviare un gruppo di messaggi o un messaggio logico nel mezzo, dopo che si è verificato un malfunzionamento del sistema. Quando il sistema viene riavviato, l'applicazione può impostare i campi *GroupId*, *MsgSeqNumber*, *Offset*, *MsgFlags*e *Persistence* sui valori appropriati, quindi emettere la chiamata MQPUT con MQPMO_SYNCPOINT o MQPMO_NO_SYNCPOINT impostati come richiesto, ma senza specificare MQPMO_LOGICAL_ORDER. Se questa chiamata ha esito positivo, il gestore code conserva le informazioni sul gruppo e sul segmento e le successive chiamate MQPUT che utilizzano tale handle di coda possono specificare MQPMO_LOGICAL_ORDER come normale.

Le informazioni sul gruppo e sul segmento che il gestore code conserva per la chiamata MQPUT sono separate dal gruppo e le informazioni sul segmento che conserva per la chiamata MQGET.

Per qualsiasi handle di coda fornito, l'applicazione può combinare chiamate MQPUT che specificano MQPMO_LOGICAL_ORDER con chiamate MQPUT che non lo fanno, ma notare i seguenti punti:

- Se MQPMO_LOGICAL_ORDER non viene specificato, ogni chiamata MQPUT eseguita correttamente fa sì che il gestore code imposti le informazioni sul gruppo e sul segmento per l'handle di coda sui valori specificati dall'applicazione, sostituendo le informazioni sul segmento e sul gruppo esistenti conservate dal gestore code per l'handle di coda.
- Se MQPMO_LOGICAL_ORDER non è specificato, la chiamata non ha esito negativo se è presente un messaggio logico o un gruppo di messaggi corrente; la chiamata potrebbe avere esito positivo con un codice di completamento MQCC_WARNING. Tabella 37 a pagina 253 mostra i vari casi che possono verificarsi. In questi casi, se il codice di completamento non è MQCC_OK, il codice di errore è uno dei seguenti (come appropriato):
  - GRUPPO_INCOMPLE_MQRC
  - MQRC_INCOMPLETE_MSG
  - MQRC_INCONSIST_PERSISTENZA
  - UOW MQRC_INCONSISTENT_

  Nota: Il gestore code non controlla le informazioni sul gruppo e sul segmento per la chiamata MQPUT1 .

Tabella 37. Risultato quando la chiamata MQPUT o MQCLOSE non è congruente con le informazioni sul gruppo e sul segmento

La chiamata corrente è	La chiamata precedente era MQPUT con MQPMO_LOGICAL_ORDER	La chiamata precedente era MQPUT senza MQPMO_LOGICAL_ORDER
MQPUT con MQPMO_LOGICAL_ORDER	MQCC_NON RIUSCITO	MQCC_NON RIUSCITO
MQPUT senza MQPMO_LOGICAL_ORDER	MQCC_AVVERTENZA	MQCC_OK
MQCLOSE con un gruppo non terminato o un messaggio logico	MQCC_AVVERTENZA	MQCC_OK

Per le applicazioni che inserano i messaggi e i segmenti in ordine logico, specificare MQPMO_LOGICAL_ORDER, poiché è l'opzione più semplice da utilizzare. Questa opzione allevia l'applicazione della necessità di gestire le informazioni sul gruppo e sul segmento, poiché il gestore code gestisce tali informazioni. Tuttavia, le applicazioni specializzate potrebbero richiedere un controllo maggiore rispetto a quello fornito dall'opzione MQPMO_LOGICAL_ORDER, che può essere ottenuto non specificando tale opzione; in caso contrario, è necessario assicurarsi che i campi*GroupId*, *MsgSeqNumber*, *Offset*e *MsgFlags* in MQMD siano impostati correttamente, prima di ogni chiamata MQPUT o MQPUT1 .

Ad esempio, un'applicazione che desidera inoltrare i messaggi fisici che riceve, indipendentemente dal fatto che tali messaggi si trovino in gruppi o segmenti di messaggi logici, non deve specificare MQPMO_LOGICAL_ORDER, per due motivi:

- Se i messaggi vengono richiamati e messi in ordine, specificando MQPMO_LOGICAL_ORDER si assegna un nuovo identificativo di gruppo ai messaggi, il che potrebbe rendere difficile o impossibile per il creatore dei messaggi correlare eventuali messaggi di risposta o di report che risultano dal gruppo di messaggi.
- In una rete complessa con più percorsi tra i gestori code di invio e di ricezione, i messaggi fisici potrebbero arrivare fuori ordine. Non specificando MQPMO_LOGICAL_ORDER e MQGMO_LOGICAL_ORDER nella chiamata MQGET, l'applicazione di inoltro può richiamare e inoltrare ogni messaggio fisico non appena arriva, senza attendere il successivo in ordine logico.

Le applicazioni che generano messaggi di report per messaggi in gruppi o segmenti di messaggi logici non devono specificare MQPMO_LOGICAL_ORDER durante l'inserimento del messaggio di report.

MQPMO_LOGICAL_ORDER può essere specificato con una delle altre opzioni MQPMO_ *.

Inserimento di gruppi ordinati in modo logico in una coda cluster (MQOO_BIND_ON_GROUP)

L'opzione MQOO_BIND_ON_OPEN garantisce che tutti i messaggi da questa applicazione, e quindi tutti i gruppi, vengano instradati a una singola istanza. Questo ha lo svantaggio che il traffico dell'applicazione non è bilanciato sul carico tra più istanze di una coda cluster. Per abilitare il bilanciamento del carico di lavoro mantenendo intatti i gruppi di messaggi, è necessario impostare le seguenti opzioni:

- La chiamata MQPUT deve specificare MQPMO_LOGICAL_ORDER
- La chiamata MQOPEN deve specificare una delle seguenti opzioni:
  - Gruppo_BIND_MQOO
  - MQOO_BIND_AS_Q_DEF e la definizione della coda devono specificare DEFBIND (GROUP)

Il bilanciamento del carico di lavoro viene quindi gestito *tra gruppi* di messaggi senza richiedere MQCLOSE e MQOPEN della coda. *Tra gruppi* significa che MQMF_MSG_IN_GROUP è impostato in MQMD (v2) o MQMDE e che non vi è alcun gruppo parzialmente completo in corso. Quando un gruppo è in corso, il gestore code risolto e il nome coda nell'handle dell'oggetto vengono riutilizzati.

Se il messaggio precedente era MQPMO_LOGICAL_ORDER e / o MQMF_MSG_IN_GROUP era impostato, ma il messaggio corrente non fa parte del gruppo, la chiamata PUT ha esito negativo con MQRC_INCOMPLETE_GROUP.

Se un singolo MQPUT non specifica MQPMO_LOGICAL_ORDER e non è attivo alcun gruppo corrente, il bilanciamento del carico di lavoro viene guidato per quel messaggio (come se la chiamata MQOPEN avesse specificato MQOO_BIND_NOT_FIXED).

Non viene eseguita alcuna riassegnazione per i messaggi collegati a una destinazione utilizzando MQOO_BIND_ON_GROUP. Per ulteriori informazioni sulla riallocazione, consultare “Gruppi di messaggi” a pagina 36.

Raggruppamento di messaggi logici

Esistono due motivi principali per utilizzare i messaggi logici in un gruppo:

- Potrebbe essere necessario elaborare i messaggi in un ordine particolare.
- Potrebbe essere necessario elaborare ogni messaggio in un gruppo in modo correlato.

In entrambi i casi, richiamare l'intero gruppo con la stessa istanza dell'applicazione di richiamo.

Ad esempio, si supponga che il gruppo sia composto da quattro messaggi logici. L'applicazione di inserimento è simile alla seguente:

```
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
```

L'applicazione di richiamo specifica l'opzione MQGMO_ALL_MSGS_AVAILABLE per il primo messaggio nel gruppo. Ciò garantisce che l'elaborazione non venga avviata fino all'arrivo di tutti i messaggi all'interno del gruppo. L'opzione MQGMO_ALL_MSGS_AVAILABLE viene ignorata per i messaggi successivi all'interno del gruppo.

Quando viene richiamato il primo messaggio logico del gruppo, è possibile utilizzare MQGMO_LOGICAL_ORDER per garantire che i rimanenti messaggi logici del gruppo vengano richiamati in ordine.

Quindi, l'applicazione di acquisizione si presenta come segue:

```
/* 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
```

Per ulteriori esempi di raggruppamento dei messaggi, consultare "Segmentazione dell'applicazione dei messaggi logici" a pagina 266 e "Mettere e ottenere un gruppo che si estende alle unità di lavoro" a pagina 255.

Per informazioni su come consentire a un'applicazione di richiedere che un gruppo di messaggi sia assegnato alla stessa istanza di destinazione per code cluster, consultare DefBind.

Mettere e ottenere un gruppo che si estende alle unità di lavoro

Nel caso precedente, i messaggi o i segmenti non possono iniziare a lasciare il nodo (se la sua destinazione è remota) o ad essere richiamati fino a quando l'intero gruppo non è stato inserito e non è stato eseguito il commit dell'unità di lavoro. Questo potrebbe non essere quello che si desidera se si impiega molto tempo per inserire l'intero gruppo o se lo spazio della coda è limitato sul nodo. Per superare questo problema, mettere il gruppo in diverse unità di lavoro.

Se il gruppo viene inserito all'interno di più unità di lavoro, è possibile che parte del gruppo esegua il commit anche quando l'applicazione di inserimento non riesce. L'applicazione deve quindi salvare le informazioni sullo stato, sottoposte a commit con ogni unità di lavoro, che può utilizzare dopo un riavvio per riprendere un gruppo incompleto. La posizione più semplice per registrare queste informazioni si trova in una coda STATUS. Se un gruppo completo è stato inserito correttamente, la coda STATUS è vuota.

Se la segmentazione è coinvolta, la logica è simile. In questo caso, StatusInfo deve includere *Offset* .

Di seguito viene riportato un esempio di inserimento del gruppo in diverse unità di lavoro:

```
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
```

Se tutte le unità di lavoro sono state sottoposte a commit, l'intero gruppo è stato inserito correttamente e la coda STATUS è vuota. In caso contrario, il gruppo deve essere ripreso nel punto indicato dalle

informazioni sullo stato. MQPMO_LOGICAL_ORDER non può essere utilizzato per la prima immissione, ma può essere utilizzato successivamente.

L'elaborazione del riavvio è simile alla seguente:

```
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
```

Dall'applicazione di acquisizione, è possibile che si desideri avviare l'elaborazione dei messaggi in un gruppo prima dell'arrivo dell'intero gruppo. Ciò migliora i tempi di risposta sui messaggi all'interno del gruppo e significa anche che la memoria non è richiesta per l'intero gruppo. Per realizzare i vantaggi, utilizzare diverse unità di lavoro per ciascun gruppo di messaggi. Per motivi di correzione, è necessario richiamare ogni messaggio all'interno di un'unità di lavoro.

Come per la corrispondente applicazione di inserimento, ciò richiede che le informazioni di stato vengano registrate automaticamente quando viene eseguito il commit di ciascuna unità di lavoro. Anche in questo caso, la posizione più semplice per registrare queste informazioni è in una coda STATUS. Se un gruppo completo è stato elaborato correttamente, la coda STATUS è vuota.

Nota: Per le unità di lavoro intermedie, è possibile evitare le richiami MQGET dalla coda STATUS specificando che ogni MQPUT nella coda di stato è un segmento di un messaggio (ovvero, impostando l'indicatore MQMF_SEGMENT), invece di inserire un nuovo messaggio completo per ogni unità di lavoro. Nell'ultima unità di lavoro, un segmento finale viene inserito nella coda di stato specificando MQMF_LAST_SEGMENT e le informazioni di stato vengono cancellate con un MQGET che specifica MQGMO_COMPLETE_MSG.

Durante l'elaborazione del riavvio, anziché utilizzare un singolo MQGET per ottenere un possibile messaggio di stato, sfogliare la coda di stato con MQGMO_LOGICAL_ORDER fino a raggiungere l'ultimo segmento (ossia, fino a quando non viene restituito alcun ulteriore segmento). Nella prima unità di lavoro dopo il riavvio, specificare esplicitamente l'offset quando si colloca il segmento di stato.

Nel seguente esempio, vengono considerati solo i messaggi all'interno di un gruppo, supponendo che il buffer dell'applicazione sia sempre abbastanza grande da contenere l'intero messaggio, indipendentemente dal fatto che il messaggio sia stato segmentato o meno. MQGMO_COMPLETE_MSG viene quindi specificato su ogni MQGET. Gli stessi principi si applicano se è coinvolta la segmentazione (in questo caso, StatusInfo deve includere *Offset*).

Per semplicità, si presume che un massimo di 4 messaggi vengano richiamati all'interno di una singola UOW:

```
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
```

Se è stato eseguito il commit di tutte le unità di lavoro, l'intero gruppo è stato richiamato correttamente e la coda STATUS è vuota. In caso contrario, il gruppo deve essere ripreso nel punto indicato dalle informazioni sullo stato. MQGMO_LOGICAL_ORDER non può essere utilizzato per il primo richiamo, ma può essere utilizzato successivamente.

L'elaborazione del riavvio è simile alla seguente:

```
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
```

Ricezione di un messaggio particolare

Esistono diversi modi per ottenere un determinato messaggio da una coda. Questi sono: selezionando MsgId e CorrelId, selezionando GroupId, MsgSeqNumber e Offset e selezionando MsgToken. È anche possibile utilizzare una stringa di selezione quando si apre la coda.

Per ottenere un particolare messaggio da una coda, utilizzare il campo *MsgId* e *CorrelId* della struttura MQMD. Tuttavia, le applicazioni possono impostare esplicitamente questi campi, in modo che i valori

specificati potrebbero non identificare un messaggio univoco. Tabella 38 a pagina 258 mostra quale messaggio viene richiamato per le possibili impostazioni di questi campi. Questi campi vengono ignorati all'immissione se si specifica MQGMO_MSG_UNDER_CURSOR nel parametro *GetMsgOpts* della chiamata MQGET.

Tabella 38. Utilizzo degli identificatori di correlazione e di messaggio

Per richiamare ...	***MsgId***	***CorrelId***
Primo messaggio nella coda	MQMI_NONE	MQCI_NONE
Primo messaggio che corrisponde a *MsgId*	Diverso da zero	MQCI_NONE
Primo messaggio che corrisponde a *CorrelId*	MQMI_NONE	Diverso da zero
Primo messaggio che corrisponde a *MsgId* e *CorrelId*	Diverso da zero	Diverso da zero

In ogni caso, *primo* indica il primo messaggio che soddisfa il criterio di selezione (a meno che non venga specificato MQGMO_BROWSE_NEXT, quando indica il *successivo* messaggio nella sequenza che soddisfa il criterio di selezione).

Al ritorno, la chiamata MQGET imposta i campi *MsgId* e *CorrelId* sugli identificativi del messaggio e della correlazione del messaggio restituito, se presenti.

Se si imposta il campo *Version* della struttura MQMD su 2, è possibile utilizzare i campi *GroupId*, *MsgSeqNumbe*r e *Offset* . Tabella 39 a pagina 258 mostra quale messaggio viene richiamato per le possibili impostazioni di questi campi.

Tabella 39. Utilizzo dell'identificativo del gruppo

Per richiamare ...	**opzioni di corrispondenza**
Primo messaggio nella coda	MQMO_NONE
Primo messaggio che corrisponde a *MsgId*	ID MQMO_MATCH_MSG_ID
Primo messaggio che corrisponde a *CorrelId*	ID CORREL_MQMO_MATCH_
Primo messaggio che corrisponde a *GroupId*	ID_GROUP_MATCH_MQMO
Primo messaggio che corrisponde a *MsgSeqNumber*	NUMERO SEQ MQMO_MATCH_MSG_
Primo messaggio che corrisponde a *MsgToken*	MQMO_MATCH_MSG_TOKEN
Primo messaggio che corrisponde a *Offset*	MQMO_MATCH_OFFSET

Note:

1. MQMO_MATCH_XXX implica che il campo *XXX* nella struttura MQMD sia impostato sul valore da mettere in corrispondenza.
2. Gli indicatori MQMO possono essere utilizzati in combinazione. Ad esempio, MQMO_MATCH_GROUP_ID, MQMO_MATCH_MSG_SEQ_NUMBER e MQMO_MATCH_OFFSET possono essere utilizzati insieme per fornire il segmento identificato dai campi *GroupId*, *MsgSeqNumbe*r e *Offset* .
3. Se si specifica MQGMO_LOGICAL_ORDER, il messaggio che si sta tentando di richiamare è interessato perché l'opzione dipende dalle informazioni di stato controllate per l'handle della coda. Per ulteriori informazioni, consultare “Ordinamento logico e fisico” a pagina 246 e Opzioni.

La chiamata MQGET di solito richiama il primo messaggio da una coda. Se si specifica un particolare messaggio quando si utilizza la chiamata MQGET, il gestore code deve ricercare la coda fino a quando non trova quel messaggio. Ciò può influire sulle prestazioni dell'applicazione.

Se si utilizza la versione 2 o successiva della struttura MQGMO e non si specificano gli indicatori MQMO_MATCH_MSG_ID o MQMO_MATCH_CORREL_ID, non è necessario reimpostare i campi *MsgId* o *CorrelId* tra le MQGET.

È possibile ottenere un messaggio specifico da una coda specificando MsgToken e MatchOption MQMO_MATCH_MSG_TOKEN nella struttura MQGMO. Il MsgToken viene restituito dalla chiamata MQPUT che in origine ha inserito tale messaggio nella coda o dalle precedenti operazioni MQGET e rimane costante a meno che il gestore code non venga riavviato.

Se si è interessati solo a un sottoinsieme di messaggi sulla coda, è possibile specificare quali messaggi si desidera elaborare utilizzando una stringa di selezione con la chiamata MQOPEN o MQSUB. MQGET richiama quindi il messaggio successivo che soddisfa tale stringa di selezione. Per ulteriori informazioni sulle stringhe di selezione, consultare “Selettori” a pagina 22.

Miglioramento delle prestazioni dei messaggi non persistenti

Quando un client richiede un messaggio da un server, invia una richiesta al server. Invia una richiesta separata per ciascuno dei messaggi che utilizza. Per migliorare le prestazioni di un client che utilizza messaggi non persistenti evitando di dover inviare questi messaggi di richiesta, è possibile configurare un client per utilizzare la *lettura anticipata*. La lettura anticipata consente l'invio di messaggi a un cliente senza che un'applicazione debba richiederli.

Quando la lettura anticipata è abilitata, i messaggi vengono inviati a un buffer di memoria sul client denominato *buffer di lettura anticipata*. Il client disporrà di un buffer di lettura anticipata per ogni coda aperta con lettura anticipata abilitata. I messaggi nel buffer di lettura anticipata non sono persistenti. Il client aggiorna periodicamente il server con informazioni sulla quantità di dati che ha utilizzato.

Quando si richiama MQOPEN con MQOO_READ_AHEAD, il client WebSphere MQ abilita la lettura anticipata solo se vengono soddisfatte determinate condizioni. Queste condizioni includono:

- Sia il client che il gestore code remoto devono trovarsi in WebSphere MQ Versione 7 o successive.
- L'applicazione client deve essere compilata e collegata rispetto alle librerie client MQI WebSphere MQ con thread.
- Il canale del client deve utilizzare il protocollo TCP/IP
- Il canale deve avere un'impostazione SharingConversations (SHARECNV) diversa da zero nelle definizioni di canale del client e del server.

L'utilizzo della lettura anticipata può migliorare le prestazioni quando si utilizzano messaggi non persistenti da un'applicazione client. Questo miglioramento delle performance è disponibile sia per le applicazioni MQI che JMS. Le applicazioni client che utilizzano MQGET o il consumo asincrono beneficeranno dei miglioramenti delle prestazioni quando si utilizzano messaggi non persistenti.

Non tutte le progettazioni di applicazioni client sono adatte per l'utilizzo della lettura anticipata poiché non tutte le opzioni sono supportate per l'utilizzo con la lettura anticipata e alcune opzioni sono richieste per essere congruenti tra le chiamate MQGET quando la lettura anticipata è abilitata. Se un client modifica i propri criteri di selezione tra le chiamate MQGET, i messaggi memorizzati nel buffer di lettura anticipata rimarranno bloccati nel buffer di lettura anticipata del client.

Se un backlog di messaggi bloccati con i criteri di selezione precedenti non è più richiesto, è possibile impostare un intervallo di eliminazione configurabile sul client per eliminare automaticamente tali messaggi dal client. L'intervallo di eliminazione è uno di un gruppo di opzioni di ottimizzazione lettura anticipata determinato dal client. È possibile regolare queste opzioni per soddisfare le vostre esigenze.

Se un'applicazione client viene riavviata, i messaggi nel buffer di lettura anticipata possono essere persi. Al contrario, un messaggio spostato in un buffer di lettura anticipata potrebbe essere eliminato dalla coda sottostante; ciò non ne comporta la rimozione dal buffer, quindi una chiamata MQGET che utilizza la lettura anticipata può restituire un messaggio che non esiste più.

La lettura anticipata viene eseguita solo per i collegamenti client. L'attributo viene ignorato per tutti gli altri bind.

La lettura anticipata non ha alcun effetto sull'attivazione. Nessun messaggio trigger viene generato quando un messaggio viene letto in anticipo dal client. La lettura anticipata non genera informazioni statistiche e account quando è abilitata.

Utilizzo della lettura anticipata con la messaggistica di pubblicazione - sottoscrizione

Quando un'applicazione di sottoscrizione specifica una coda di destinazione a cui vengono inviate le pubblicazioni, il valore DEFREADA della coda specificata viene utilizzato come valore di lettura anticipata predefinito.

Quando un'applicazione di sottoscrizione richiede che WebSphere MQ gestisca la destinazione a cui vengono inviate le pubblicazioni, una coda gestita viene creata come una coda dinamica basata su una coda modello predefinita. È il valore DEFREADA della coda modello utilizzato come valore di lettura anticipata predefinito. Le code modello predefinite SYSTEM.DURABLE.PUBLICATIONS.MODEL o SYSTEM.NONDURABLE.PUBLICATIONS.MODEL vengono utilizzati a meno che non venga definita una coda modello per questo o per un argomento principale.

Concetti correlati

“Ottimizzazione delle prestazioni per i messaggi non persistenti su AIX” a pagina 262
Se si utilizza AIX V5.3 o versioni successive, considerare l'impostazione del parametro di ottimizzazione per utilizzare le prestazioni complete per i messaggi non persistenti.

Attività correlate

“Abilitazione e disabilitazione della lettura anticipata” a pagina 261
Per impostazione predefinita la lettura anticipata è disabilitata. È possibile abilitare la lettura anticipata a livello di coda o di applicazione.

Riferimenti correlati

“Opzioni MQGET e lettura anticipata” a pagina 260
Non tutte le opzioni MQGET sono supportate quando la lettura anticipata è abilitata; alcune opzioni sono richieste per essere congruenti tra le chiamate MQGET.

Opzioni MQGET e lettura anticipata

Non tutte le opzioni MQGET sono supportate quando la lettura anticipata è abilitata; alcune opzioni sono richieste per essere congruenti tra le chiamate MQGET.

Quando si richiama MQOPEN con MQOO_READ_AHEAD, il client WebSphere MQ abilita la lettura anticipata solo se vengono soddisfatte determinate condizioni. Queste condizioni includono:

- Sia il client che il gestore code remoto devono trovarsi in WebSphere MQ Versione 7 o successive.
- L'applicazione client deve essere compilata e collegata rispetto alle librerie client MQI WebSphere MQ con thread.
- Il canale del client deve utilizzare il protocollo TCP/IP
- Il canale deve avere un'impostazione SharingConversations (SHARECNV) diversa da zero nelle definizioni di canale del client e del server.

La seguente tabella indica quali opzioni sono supportate per l'utilizzo con la lettura anticipata e se possono essere modificate tra le chiamate MQGET.

Tabella 40. Opzioni MQGET e lettura anticipata

	Consentito quando la lettura anticipata è abilitata e può essere modificato tra chiamate MQGET[5]	**Consentito quando la lettura anticipata è abilitata ma non può essere modificato tra chiamate MQGET[1]**	**Opzioni MQGET non consentite quando la lettura anticipata è abilitata[2]**
valori MQGET MQMD	ID messaggio[3] ID correlazione[3]	Codifica CodedCharSetId	

Tabella 40. Opzioni MQGET e lettura anticipata (Continua)			
	Consentito quando la lettura anticipata è abilitata e può essere modificato tra chiamate MQGET[5]	**Consentito quando la lettura anticipata è abilitata ma non può essere modificato tra chiamate MQGET[1]**	**Opzioni MQGET non consentite quando la lettura anticipata è abilitata[2]**
Opzioni MQGET MQGMO	• MQGMO_NO_WAIT • MQGMO_BROWSE_MESSAGE _SOTTO_CURSORE • MQGMO_BROWSE_FIRST • MQGMO_BROWSE_SUCCESSIVO • MQGMO_FAIL_IF_QUIESCING	• MQGMO_SYNCPOINT_IF _PERSISTENTE • MQGMO_NO_SYNCPOINT • MQGMO_ACCEPT_TRUNCATED _MSG • MQGMO_CONVERT	• MQGMO_SET_SIGNAL • SYNCPOINT MQGMO • MQGMO_MARK_SKIP _BACKOUT • MQGMO_MSG_UNDER _CURSORE[4] • LOCK_MQGMO • MQGMO_UNLOCK • ORDER LOGICAL_MQGMO_ • MQGMO_COMPLETE_MSG • MQGMO_ALL_MSGS_AVAILABLE • MQGMO_ALL_SEGMENTS_ DISPONIBILE

Note:

1. Se queste opzioni vengono modificate tra le chiamate MQGET, viene restituito un codice motivo MQRC_OPTIONS_CHANGED.
2. Se queste opzioni vengono specificate nella prima chiamata MQGET, il read ahead è disabilitato. Se queste opzioni vengono specificate in una successiva chiamata MQGET, viene restituito il codice motivo MQRC_OPTIONS_ERROR.
3. Se un'applicazione client altera i valori MsgId e CorrelId tra le chiamate MQGET, i messaggi con i valori precedenti potrebbero già essere stati inviati al client e rimarranno nel buffer di lettura anticipata del client fino a quando non saranno utilizzati (o eliminati automaticamente).
4. MQGMO_MSG_UNDER_CURSOR non è consentito con il read ahead. La lettura anticipata è disabilitata quando vengono specificate entrambe le opzioni MQOO_BROWSE e MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE durante l'apertura della coda.
5. Quando la lettura anticipata è abilitata, il primo MQGET determina se i messaggi devono essere esplorati o ricevuti da una coda. Se l'applicazione client utilizza MQGET con le opzioni modificate, come ad esempio il tentativo di esplorazione dopo una lettura iniziale o il tentativo di acquisizione dopo una lettura iniziale, viene restituito un codice motivo MQRC_OPTIONS_CHANGED.

Se un client modifica i propri criteri di selezione tra chiamate MQGET, i messaggi memorizzati nel buffer di lettura anticipata che corrispondono ai criteri di selezione iniziali non vengono utilizzati dall'applicazione client e rimangono bloccati nel buffer di lettura anticipata del client. In situazioni in cui il buffer di lettura anticipata del client contiene molti messaggi bloccati, i vantaggi associati alla lettura anticipata vengono persi e viene richiesta una richiesta separata al server per ogni messaggio utilizzato. Per determinare se la lettura anticipata viene utilizzata in modo efficiente, è possibile utilizzare il parametro di stato della connessione READA.

La lettura anticipata può essere inibita quando richiesta da un'applicazione a causa di opzioni incompatibili specificate sulla prima chiamata MQGET. In questa situazione lo stato della connessione mostra che la lettura anticipata è inibita.

Se, a causa di queste limitazioni su MQGET, si decide che una progettazione dell'applicazione client non è adatta per la lettura anticipata, specificare l'opzione MQOPEN MQOO_READ_AHEAD_NO. In alternativa, impostare il valore di lettura anticipata predefinito della coda che si sta aprendo su NO o su DISABLED.

Abilitazione e disabilitazione della lettura anticipata

Per impostazione predefinita la lettura anticipata è disabilitata. È possibile abilitare la lettura anticipata a livello di coda o di applicazione.

Informazioni su questa attività

Quando si richiama MQOPEN con MQOO_READ_AHEAD, il client WebSphere MQ abilita la lettura anticipata solo se vengono soddisfatte determinate condizioni. Queste condizioni includono:

- Sia il client che il gestore code remoto devono trovarsi in WebSphere MQ Versione 7 o successive.

- L'applicazione client deve essere compilata e collegata rispetto alle librerie client MQI WebSphere MQ con thread.
- Il canale del client deve utilizzare il protocollo TCP/IP
- Il canale deve avere un'impostazione SharingConversations (SHARECNV) diversa da zero nelle definizioni di canale del client e del server.

Per abilitare la lettura anticipata:

- Per configurare la lettura anticipata a livello della coda, impostare l'attributo della coda DEFREADA su YES.
- Per configurare la lettura anticipata a livello di applicazione:
  - per utilizzare la lettura anticipata laddove possibile, utilizzare l'opzione MQOO_READ_AHEAD sulla chiamata della funzione MQOPEN. Non è possibile per l'applicazione client utilizzare la lettura anticipata se l'attributo della coda DEFREADA è stato impostato su DISABLED.
  - per utilizzare la sola lettura anticipata quando la lettura anticipata è abilitata su una coda, utilizzare l'opzione MQOO_READ_AHEAD_AS_Q_DEF sulla chiamata alla funzione MQOPEN.

Se la progettazione di un'applicazione client non è adatta per la lettura anticipata, è possibile disabilitarla:

- al livello della coda impostando l'attributo della coda, DEFREADA su NO se non si desidera che venga utilizzata la lettura anticipata a meno che non sia richiesta da un'applicazione client o DISABLED se non si desidera che venga utilizzata la lettura anticipata indipendentemente dal fatto che la lettura anticipata sia richiesta da una applicazione client.
- a livello dell'applicazione utilizzando l'opzione MQOO_NO_READ_AHEAD sulla chiamata della funzione MQOPEN.

Due opzioni MQCLOSE consentono di configurare cosa accade ai messaggi memorizzati nel buffer di lettura anticipata se la coda è chiusa.

- Utilizzare MQCO_IMMEDIATE per eliminare i messaggi nel buffer di lettura anticipata.
- Utilizzare MQCO_QUIESCE per garantire che i messaggi nel buffer di lettura anticipata vengano utilizzati dall'applicazione prima che la coda venga chiusa. Quando viene emesso MQCLOSE con MQCO_QUIESCE e ci sono messaggi rimanenti nel buffer di lettura anticipata, MQRC_READ_AHEAD_MSGS restituisce MQCC_WARNING.

Ottimizzazione delle prestazioni per i messaggi non persistenti su AIX

Se si utilizza AIX V5.3 o versioni successive, considerare l'impostazione del parametro di ottimizzazione per utilizzare le prestazioni complete per i messaggi non persistenti.

Per impostare il parametro di ottimizzazione in modo che diventi immediatamente effettivo, immettere il seguente comando come utente root:

```
/usr/sbin/ioo -o j2_nPagesPerWriteBehindCluster=0
```

Per impostare il parametro di ottimizzazione in modo che abbia effetto immediato e che persista durante i riavvii, immettere il seguente comando come utente root:

```
/usr/sbin/ioo -p -o j2_nPagesPerWriteBehindCluster=0
```

Di solito, i messaggi non persistenti vengono conservati solo in memoria, ma ci sono circostanze in cui AIX può pianificare i messaggi non persistenti da scrivere sul disco. I messaggi pianificati per essere scritti su disco non sono disponibili per MQGET fino a quando non viene completata la scrittura su disco. Il comando di ottimizzazione suggerito varia questa soglia; invece di pianificare i messaggi da scrivere su disco quando vengono accodati 16 kilobyte di dati, la scrittura su disco si verifica solo quando la memoria reale sulla macchina si avvicina al riempimento. Si tratta di una modifica globale che potrebbe interessare altri componenti software.

Su AIX, quando si utilizzano applicazioni a più thread e soprattutto quando si utilizzano macchine con più processori, si consiglia di impostare AIXTHREAD_SCOPE=S nell'id mqm `.profile` o impostare AIXTHREAD_SCOPE=S nell'ambiente prima di avviare l'applicazione, per ottenere prestazioni migliori e una pianificazione più solida. Ad esempio:

```
export AIXTHREAD_SCOPE=S
```

L'impostazione di AIXTHREAD_SCOPE=S indica che i thread utente creati con gli attributi predefiniti vengono inseriti in un ambito di conflitto a livello di sistema. Se un thread utente viene creato con un ambito di conflitto a livello di sistema, è collegato a un thread kernel ed è pianificato dal kernel. Il thread del kernel sottostante non è condiviso con nessun altro thread utente.

Descrittori file

Quando si esegue un processo a più thread, ad esempio il processo agent, è possibile raggiungere il limite soft per i descrittori file. Questo limite fornisce il codice di errore IBM WebSphere MQ `MQRC_UNEXPECTED_ERROR (2195)` e, se ci sono descrittori di file sufficienti, un file IBM WebSphere MQ FFST™ .

Per evitare questo problema, è possibile aumentare il limite di processo per il numero di descrittori file. Per effettuare questa operazione, modificare l'attributo `nofiles` in `/etc/security/limits` in 10.000 per l'ID utente mqm o nella stanza predefinita.

Limiti risorse di sistema

Impostare il limite di risorse di sistema per il segmento dati e il segmento stack su illimitato utilizzando i seguenti comandi in un prompt dei comandi:

```
ulimit -d unlimited
ulimit -s unlimited
```

Gestione dei messaggi con lunghezza superiore a 4 MB

I messaggi possono essere troppo grandi per l'applicazione, la coda o il gestore code. In base all'ambiente, WebSphere MQ fornisce una serie di modi per gestire i messaggi più lunghi di 4 MB.

È possibile aumentare l'attributo *MaxMsgLength* fino a 100 MB su tutti i sistemi WebSphere MQ alla versione V6 o successiva. Impostare questo valore per riflettere la dimensione dei messaggi che utilizzano la coda. Su sistemi WebSphere MQ diversi da WebSphere MQ per z/OS, è anche possibile:

1. Utilizzare messaggi segmentati. (I messaggi possono essere segmentati dall'applicazione o dal gestore code.)
2. Utilizzare i messaggi di riferimento.

Ciascuno di questi approcci è descritto nel resto di questa sezione.

Aumento della lunghezza massima del messaggio

L'attributo Gestore code *MaxMsgLength* definisce la lunghezza massima di un messaggio che può essere gestito da un gestore code. Allo stesso modo, l'attributo della coda *MaxMsgLength* è la lunghezza massima di un messaggio che può essere gestita da una coda. La lunghezza massima del messaggio *predefinita* supportata dipende dall'ambiente in cui si sta lavorando.

Se si gestiscono messaggi di grandi dimensioni, è possibile modificare questi attributi in modo indipendente. È possibile impostare il valore dell'attributo del gestore code nell'intervallo tra 32768 byte e 100 MB; è possibile impostare il valore dell'attributo della coda nell'intervallo tra 0 e 100 MB.

Dopo aver modificato uno o entrambi gli attributi *MaxMsgLength* , riavviare le applicazioni e i canali per rendere effettive le modifiche.

Quando vengono apportate queste modifiche, la lunghezza del messaggio deve essere inferiore o uguale sia alla coda che agli attributi *MaxMsgLength* del gestore code. Tuttavia, i messaggi esistenti potrebbero essere più lunghi di entrambi gli attributi.

Se il messaggio è troppo grande per la coda, viene restituito MQRC_MSG_TOO_BIG_FOR_Q.
Allo stesso modo, se il messaggio è troppo grande per il gestore code, viene restituito MQRC_MSG_TOO_BIG_FOR_Q_MGR.

Questo metodo di gestione di messaggi di grandi dimensioni è facile e conveniente. Tuttavia, considerare i seguenti fattori prima di utilizzarlo:

- L'uniformità tra i gestori code è ridotta. La dimensione massima dei dati del messaggio è determinata dal *MaxMsgLength* per ogni coda (incluse le code di trasmissione) in cui verrà inserito il messaggio. Questo valore è spesso impostato sul valore predefinito *MaxMsgLength*del gestore code, specialmente per le code di trasmissione. Ciò rende difficile prevedere se un messaggio è troppo grande quando deve essere inviato a un gestore code remoto.
- L'utilizzo delle risorse di sistema è aumentato. Ad esempio, le applicazioni necessitano di buffer più grandi e, su alcune piattaforme, potrebbe verificarsi un maggiore utilizzo della memoria condivisa. La memoria della coda dovrebbe essere interessata solo se effettivamente richiesta per i messaggi più grandi.
- Il batch del canale è interessato. Un messaggio di grandi dimensioni conta ancora come un solo messaggio per il conteggio batch, ma ha bisogno di più tempo per la trasmissione, aumentando così i tempi di risposta per gli altri messaggi.

Segmentazione del messaggio

Utilizzare queste informazioni per informazioni sulla segmentazione dei messaggi.

Nota: Non supportato in IBM WebSphere MQ per z/OS o dalle applicazioni che utilizzano le classi IBM WebSphere MQ per JMS.

Aumentare la lunghezza massima del messaggio come descritto nell'argomento "Aumento della lunghezza massima del messaggio" a pagina 263 ha alcune implicazioni negative. Inoltre, è ancora possibile che il messaggio sia troppo grande per la coda o il gestore code. In questi casi, è possibile segmentare un messaggio. Per informazioni sui segmenti, consultare "Gruppi di messaggi" a pagina 36.

Le sezioni successive esaminano gli usi comuni per la segmentazione dei messaggi. Per l'inserimento e il richiamo in modo distruttivo, si presuppone che le chiamate MQPUT o MQGET *sempre* operino all'interno di un'unità di lavoro. Prendere sempre in considerazione l'utilizzo di questa tecnica per ridurre la possibilità che gruppi incompleti siano presenti nella rete. Il commit a fase singola viene assunto dal gestore code, ma altre tecniche di coordinamento sono ugualmente valide.

Inoltre, nelle applicazioni di richiamo, si presume che se più server elaborano la stessa coda, ogni server esegue un codice simile, in modo che un server non riesca mai a trovare un messaggio o un segmento che prevede di essere presente (poiché aveva specificato MQGMO_ALL_MSGS_AVAILABLE o MQGMO_ALL_SEGMENTS_AVAILABLE in precedenza).

Inserimento e ricezione di un messaggio segmentato che abbraccia le unità di lavoro

È possibile inserire e ottenere un messaggio segmentato che si estende a un'unità di lavoro in modo simile a "Mettere e ottenere un gruppo che si estende alle unità di lavoro" a pagina 255.

Tuttavia, non è possibile inserire o ottenere messaggi segmentati in un'unità di lavoro globale.

Segmentazione e riassemblaggio per gestore code

Questo è lo scenario più semplice, in cui un'applicazione inserisce un messaggio che deve essere richiamato da un altro. Il messaggio potrebbe essere grande: non troppo grande per l'applicazione di inserimento o di richiamo da gestire in un singolo buffer, ma troppo grande per il gestore code o una coda in cui deve essere inserito il messaggio.

Le uniche modifiche necessarie per queste applicazioni sono che l'applicazione di inserimento autorizzi il gestore code ad eseguire la segmentazione, se necessario:

```
PMO.Options = (existing options)
MD.MsgFlags = MQMF_SEGMENTATION_ALLOWED
memcpy(MD.GroupId, MQGI_NONE, MQ_GROUP_ID_LENGTH)
MQPUT
```

e per l'applicazione di richiamo per richiedere al gestore code di riassemblare il messaggio se è stato segmentato:

```
GMO.Options = MQGMO_COMPLETE_MSG | (existing options)
MQGET
```

In questo scenario più semplice, l'applicazione deve reimpostare il campo GroupId su MQGI_NONE prima della chiamata MQPUT, in modo che il gestore code possa generare un identificativo gruppo univoco per ogni messaggio. Se questa operazione non viene eseguita, i messaggi non correlati possono avere lo stesso identificativo del gruppo, che potrebbe successivamente portare a un'elaborazione non corretta.

Il buffer dell'applicazione deve essere abbastanza grande da contenere il messaggio riassemblato (a meno che non si includa l'opzione MQGMO_ACCEPT_TRUNCATED_MSG).

Se l'attributo MAXMSGLEN di una coda deve essere modificato per adattare la segmentazione del messaggio, considerare:

- Il segmento di messaggi minimo supportato su una coda locale è 16 byte.
- Per una coda di trasmissione, MAXMSGLEN deve includere anche lo spazio richiesto per le intestazioni. Considerare l'utilizzo di un valore di almeno 4000 byte superiore alla lunghezza massima prevista dei dati utente in qualsiasi segmento di messaggio che potrebbe essere inserito in una coda di trasmissione.

Se la conversione dei dati è necessaria, l'applicazione di richiamo potrebbe dover eseguire tale operazione specificando MQGMO_CONVERT. Ciò dovrebbe essere semplice perché l'uscita di conversione dati viene presentata con il messaggio completo. Non tentare di convertire i dati in un canale mittente se il messaggio è segmentato e il formato dei dati è tale che l'uscita di conversione dati non può eseguire la conversione su dati incompleti.

Segmentazione applicazione
La segmentazione dell'applicazione viene utilizzata quando la segmentazione del gestore code non è adeguata o quando le applicazioni richiedono la conversione dei dati con limiti di segmento specifici.

La segmentazione dell'applicazione viene utilizzata per due motivi principali:

1. La segmentazione del gestore code da sola non è adeguata perché il messaggio è troppo grande per essere gestito in un singolo buffer dalle applicazioni.
2. La conversione dei dati deve essere eseguita dai canali del mittente e il formato è tale che l'applicazione di inserimento deve stabilire dove devono essere i confini del segmento affinché sia possibile la conversione di un singolo segmento.

Tuttavia, se la conversione dei dati non è un problema o se l'applicazione di richiamo utilizza sempre MQGMO_COMPLETE_MSG, la segmentazione del gestore code può essere consentita anche specificando MQMF_SEGMENTATION_ALLOWED. In questo esempio, l'applicazione segmenta il messaggio in quattro segmenti:

```
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
```

Se non si utilizza MQPMO_LOGICAL_ORDER, l'applicazione deve impostare *Offset* e la lunghezza di ciascun segmento. In questo caso, lo stato logico non viene mantenuto automaticamente.

L'applicazione di richiamo non può garantire di avere un buffer abbastanza grande da contenere qualsiasi messaggio riassemblato. Deve quindi essere pronta a elaborare i segmenti singolarmente.

Per i messaggi che sono segmentati, questa applicazione non desidera avviare l'elaborazione di un segmento finché non sono presenti tutti i segmenti che costituiscono il messaggio logico. MQGMO_ALL_SEGMENTS_AVAILABLE è quindi specificato per il primo segmento. Se si specifica MQGMO_LOGICAL_ORDER ed è presente un messaggio logico corrente, MQGMO_ALL_SEGMENTS_AVAILABLE viene ignorato.

Dopo che il primo segmento di un messaggio logico è stato richiamato, utilizzare MQGMO_LOGICAL_ORDER per assicurarsi che i restanti segmenti del messaggio logico vengano richiamati in ordine.

Non viene data alcuna considerazione ai messaggi all'interno di gruppi differenti. Se tali messaggi si verificano, vengono elaborati nell'ordine in cui il primo segmento di ciascun messaggio si verifica sulla coda.

```
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
```

Segmentazione dell'applicazione dei messaggi logici
I messaggi devono essere conservati in ordine logico in un gruppo e alcuni o tutti potrebbero essere così grandi da richiedere la segmentazione dell'applicazione.

In questo esempio, è necessario inserire un gruppo di quattro messaggi logici. Tutti i messaggi tranne il terzo sono grandi e richiedono la segmentazione, che viene eseguita dall'applicazione di inserimento:

```
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
```

Nell'applicazione di richiamo, MQGMO_ALL_MSGS_AVAILABLE è specificata sul primo MQGET. Ciò significa che nessun messaggio o segmento di un gruppo viene richiamato fino a quando l'intero gruppo non è disponibile. Una volta richiamato il primo messaggio fisico di un gruppo, MQGMO_LOGICAL_ORDER viene utilizzato per garantire che i segmenti e i messaggi del gruppo vengano richiamati in ordine:

```
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
```

Nota: Se si specifica MQGMO_LOGICAL_ORDER e c'è un gruppo corrente, MQGMO_ALL_MSGS_AVAILABLE viene ignorato.

Messaggi di riferimento

Utilizzare queste informazioni per ulteriori informazioni sui messaggi di riferimento.

Nota: Non supportato in WebSphere MQ per z/OS.

Questo metodo consente ad un oggetto di grandi dimensioni di essere trasferito da un nodo ad un altro senza memorizzare l'oggetto nelle code WebSphere MQ sui nodi di origine o di destinazione. Ciò è particolarmente utile quando i dati esistono in un altro formato, ad esempio per le applicazioni di posta.

Per fare ciò, specificare un'exit dei messaggi ad entrambe le estremità di un canale. Per informazioni su come svolgere questa procedura, consultare “Programmi di uscita messaggi canale” a pagina 413.

WebSphere MQ definisce il formato di un'intestazione del messaggio di riferimento (MQRMH). Fare riferimento a MQRMH per una descrizione. Questo viene riconosciuto con un nome formato definito e potrebbe essere seguito da dati effettivi.

Per avviare un trasferimento di un LOB (large object), un'applicazione può inserire un messaggio costituito da un'intestazione del messaggio di riferimento senza dati che lo seguano. Quando questo messaggio lascia il nodo, l'exit dei messaggi richiama l'oggetto in modo appropriato e lo accoda al messaggio di riferimento. Restituisce quindi il messaggio (ora più grande di prima) all'MCA (Message Channel Agent) di invio per la trasmissione all'MCA di ricezione.

Un'altra uscita del messaggio è configurata sull'MCA di ricezione. Quando questa uscita messaggio riceve uno di questi messaggi, crea l'oggetto utilizzando i dati oggetto accodati e trasmette il messaggio di riferimento *senza* . Il messaggio di riferimento può ora essere ricevuto da un'applicazione e questa applicazione sa che l'oggetto (o almeno la parte di esso rappresentata da questo messaggio di riferimento) è stato creato in questo nodo.

La quantità massima di dati oggetto che un'uscita del messaggio di invio può accodare al messaggio di riferimento è limitata dalla lunghezza massima negoziata del messaggio per il canale. L'uscita può restituire solo un singolo messaggio all'MCA per ogni messaggio che viene passato, in modo che l'applicazione di inserimento possa inserire diversi messaggi per causare il trasferimento di un oggetto. Ogni messaggio deve identificare la lunghezza *logica* e lo scostamento dell'oggetto che deve essere accodato ad esso. Tuttavia, nei casi in cui non è possibile conoscere la dimensione totale dell'oggetto o la dimensione massima consentita dal canale, progettare l'uscita del messaggio di invio in modo che l'applicazione di inserimento immetta un singolo messaggio e l'uscita stessa immetta il messaggio successivo nella coda di trasmissione quando ha accodato più dati possibile al messaggio che è stato passato.

Prima di utilizzare questo metodo per gestire messaggi di grandi dimensioni, considerare i seguenti punti:

- L'MCA e l'uscita messaggio vengono eseguiti con un ID utente WebSphere MQ . L'uscita del messaggio (e quindi l'ID utente) deve accedere all'oggetto per richiamarlo all'estremità di invio o crearlo all'estremità di ricezione; ciò potrebbe essere possibile solo nei casi in cui l'oggetto è universalmente accessibile. Ciò pone un problema di sicurezza.
- Se il messaggio di riferimento con i dati di massa accodati deve viaggiare attraverso diversi gestori code prima di raggiungere la destinazione, i dati di massa *sono* presenti nelle code WebSphere MQ sui nodi intermedi. Tuttavia, in questi casi non è necessario fornire alcun supporto o uscite speciali.
- La progettazione dell'uscita del messaggio è resa difficile se è consentito il reinstradamento o l'accodamento di messaggi non recapitabili. In questi casi, le porzioni dell'oggetto potrebbero non essere in ordine.
- Quando un messaggio di riferimento arriva a destinazione, l'uscita del messaggio di ricezione crea un oggetto. Tuttavia, questo non è sincronizzato con l'unità di lavoro dell'MCA, quindi se viene eseguito il backout del batch, un altro messaggio di riferimento contenente questa stessa parte dell'oggetto arriverà in un batch successivo e l'uscita del messaggio potrebbe tentare di ricreare la stessa parte dell'oggetto. Se l'oggetto è, ad esempio, una serie di aggiornamenti del database, ciò potrebbe essere

inaccettabile. In tal caso, l'exit dei messaggi deve conservare un log di cui sono stati applicati gli aggiornamenti; ciò potrebbe richiedere l'utilizzo di una coda WebSphere MQ .

- A seconda delle caratteristiche del tipo di oggetto, le uscite del messaggio e le applicazioni potrebbero dover cooperare per mantenere i conteggi di utilizzo, in modo che l'oggetto possa essere eliminato quando non è più necessario. Potrebbe essere richiesto anche un identificativo di istanza; viene fornito un campo per questo nell'intestazione del messaggio di riferimento (consultare MQRMH).
- Se un messaggio di riferimento viene inserito come elenco di distribuzione, l'oggetto deve essere richiamabile per ogni elenco di distribuzione risultante o per ogni singola destinazione in quel nodo. Potrebbe essere necessario mantenere i conteggi di utilizzo. Considerare anche la possibilità che un nodo possa essere il nodo finale per alcune delle destinazioni nell'elenco, ma un nodo intermedio per altre.
- I dati di massa non vengono generalmente convertiti. Ciò è dovuto al fatto che la conversione avviene *prima* che venga richiamata l'uscita messaggio. Per questo motivo, la conversione non deve essere richiesta sul canale mittente di origine. Se il messaggio di riferimento passa attraverso un nodo intermedio, i dati di massa vengono convertiti quando vengono inviati dal nodo intermedio, se richiesto.
- I messaggi di riferimento non possono essere segmentati.

Utilizzo delle strutture di MQRMH e MQMD

Consultare MQRMH e MQMD per una descrizione dei campi nell'intestazione del messaggio di riferimento e nel descrizione del messaggio.

Nella struttura MQMD, impostare il campo *Format* su MQFMT_REF_MSG_HEADER. Il formato MQHREF, quando richiesto in MQGET, viene convertito automaticamente da WebSphere MQ insieme ai seguenti dati di massa.

Di seguito è riportato un esempio dell'utilizzo dei campi di *DataLogicalOffset* e *DataLogicalLength* di MQRMH:

Un'applicazione di inserimento potrebbe inserire un messaggio di riferimento con:

- Nessun dato fisico
- *DataLogicalLength* = 0 (questo messaggio rappresenta l'intero oggetto)
- *DataLogicalOffset* = 0.

Supponendo che l'oggetto sia lungo 70 000 byte, l'uscita del messaggio di invio invia i primi 40 000 byte lungo il canale in un messaggio di riferimento contenente:

- 40 000 byte di dati fisici dopo MQRMH
- *DataLogicalLength* = 40000
- *DataLogicalOffset* = 0 (dall'inizio dell'oggetto).

Posiziona quindi un altro messaggio sulla coda di trasmissione contenente:

- Nessun dato fisico
- *DataLogicalLength* = 0 (alla fine dell'oggetto). È possibile specificare un valore di 30 000.
- *DataLogicalOffset* = 40000 (a partire da questo punto).

Quando questa uscita del messaggio viene visualizzata dall'uscita del messaggio di invio, i rimanenti 30.000 byte di dati vengono aggiunti e i campi vengono impostati su:

- 30.000 byte di dati fisici dopo MQRMH
- *DataLogicalLength* = 30000
- *DataLogicalOffset* = 40000 (a partire da questo punto).

Viene impostato anche l'indicatore MQRMHF_LAST.

Per una descrizione dei programmi di esempio forniti per l'utilizzo dei messaggi di riferimento, consultare “Programmi di esempio per piattaforme distribuite” a pagina 97.

In attesa di messaggi

Se si desidera che un programma attenda che un messaggio arrivi su una coda, specificare l'opzione MQGMO_WAIT nel campo *Options* della struttura MQGMO.

Utilizzare il campo *WaitInterval* della struttura MQGMO per specificareil tempo massimo (in millisecondi) durante il quale si desidera che una chiamata MQGET attenda l'arrivo di un messaggio su una coda.

Se il messaggio non arriva entro questo periodo di tempo, la chiamata MQGET viene completata con il codice motivo MQRC_NO_MSG_AVAILABLE.

È possibile specificare un intervallo di attesa illimitato utilizzando la costante MQWI_UNLIMITED nel campo *WaitInterval* . Tuttavia, gli eventi al di fuori del controllo potrebbero causare un'attesa prolungata del programma, quindi utilizzare questa costante con cautela. Le applicazioni IMS non devono specificare un intervallo di attesa illimitato poiché ciò impedirebbe la chiusura del sistema IMS . (Quando IMS termina, richiede la fine di tutte le regioni dipendenti.) Invece, le applicazioni IMS possono specificare un intervallo di attesa finito; quindi, se la chiamata viene completata senza richiamare un messaggio dopo tale intervallo, emettere un'altra chiamata MQGET con l'opzione di attesa.

Nota: Se più di un programma è in attesa sulla stessa coda condivisa di *rimuovere* un messaggio, solo un programma viene attivato da un messaggio in arrivo. Tuttavia, se più di un programma è in attesa di visualizzare un messaggio, tutti i programmi possono essere attivati. Per ulteriori informazioni, consultare la descrizione del campo *Options* della struttura MQGMO in MQGMO.

Se lo stato della coda o del gestore code cambia prima della scadenza dell'intervallo di attesa, si verificano le azioni riportate di seguito:

- Se il gestore code entra nello stato di inattività ed è stata utilizzata l'opzione MQGMO_FAIL_IF_QUIESCING, l'attesa viene annullata e la chiamata MQGET viene completata con il codice motivo MQRC_Q_MGR_QUIESCING. Senza questa opzione, la chiamata rimane in attesa.
- Se il gestore code viene arrestato o annullato, la chiamata MQGET viene completata con il codice motivo MQRC_Q_MGR_STOPPING o MQRC_CONNECTION_BROKEN.
- Se gli attributi della coda (o una coda in cui si risolve il nome della coda) vengono modificati in modo che le richieste get siano ora inibite, l'attesa viene annullata e la chiamata MQGET viene completata con il codice motivo MQRC_GET_INHIBITED.
- Se gli attributi della coda (o una coda in cui si risolve il nome della coda) vengono modificati in modo che l'opzione FORCE sia obbligatoria, l'attesa viene annullata e la chiamata MQGET viene completata con il codice motivo MQRC_OBJECT_CHANGED.

Per ulteriori informazioni sulle circostanze in cui si verificano queste azioni, consultare MQGMO.

Backout ignorato

È possibile impedire a un programma di applicazione di immettere un loop *MQGET - error - backout* specificando l'opzione **MQGMO_MARK_SKIP_BACKOUT** nella chiamata MQGET.

Nota: Supportato solo su WebSphere MQ per z/OS.

Come parte di un'unità di lavoro, un programma applicativo può emettere una o più chiamate MQGET per richiamare i messaggi da una coda. Se il programma applicativo rileva un errore, può eseguire il backout dell'unità di lavoro. Ciò ripristina tutte le risorse aggiornate durante tale unità di lavoro allo stato in cui si trovavano prima dell'avvio dell'unità di lavoro e reinstalla i messaggi richiamati dalle chiamate MQGET.

Una volta ripristinati, questi messaggi sono disponibili per le successive chiamate MQGET emesse dal programma applicativo. In molti casi, ciò non causa un problema per il programma applicativo. Tuttavia, nei casi in cui l'errore che porta al backout non può essere aggirato, la reintegrazione del messaggio nella coda può far sì che il programma di applicazione immetta un loop *MQGET - error - backout* .

Per evitare questo problema, specificare l'opzione MQGMO_MARK_SKIP_BACKOUT nella chiamata MQGET. Ciò contrassegna la richiesta MQGET come non implicata nel backout avviato dall'applicazione; vale a dire, non deve essere ripristinata. L'utilizzo di questa opzione indica che quando si verifica

un backout, gli aggiornamenti ad altre risorse vengono ripristinati come richiesto, ma il messaggio contrassegnato viene considerato come se fosse stato richiamato in una nuova unità di lavoro.

Il programma applicativo deve emettere una chiamata WebSphere MQ per eseguire il commit della nuova unità di lavoro o per eseguire il backout della nuova unità di lavoro. Ad esempio, il programma può eseguire la gestione delle eccezioni, ad esempio informare il creatore che il messaggio è stato eliminato ed eseguire il commit dell'unità di lavoro in modo da rimuovere il messaggio dalla coda. Se la nuova unità di lavoro viene ripristinata (per qualsiasi motivo), il messaggio viene ripristinato sulla coda.

All'interno di un'unità di lavoro, può esistere una sola richiesta MQGET contrassegnata come backout ignorato; tuttavia, possono essere presenti diversi altri messaggi che non sono contrassegnati come backout ignorato. Una volta che un messaggio è stato contrassegnato come backout ignorato, eventuali ulteriori chiamate MQGET all'interno dell'unità di lavoro che specificano MQGMO_MARK_SKIP_BACKOUT non riescono con codice motivo MQRC_SECOND_MARK_NOT_ALLOWED.

Nota:

1. Il messaggio contrassegnato ignora il backout solo se l'unità di lavoro che lo contiene è terminata da una richiesta dell'applicazione di backout. Se l'unità di lavoro viene ripristinata per qualsiasi altro motivo, il messaggio viene ripristinato sulla coda nello stesso modo in cui lo sarebbe se non fosse stato contrassegnato per ignorare il backout.
2. Ignorare il backout non è supportato all'interno delle procedure memorizzate DB2 che partecipano in unità di lavoro controllate da RRS. Ad esempio, una chiamata MQGET con l'opzione MQGMO_MARK_SKIP_BACKOUT avrà esito negativo con il codice motivo MQRC_OPTION_ENVIRONMENT_ERROR.

Figura 36 a pagina 271 illustra una sequenza tipica di passi che un programma applicativo potrebbe contenere quando è richiesta una richiesta MQGET per ignorare il backout.

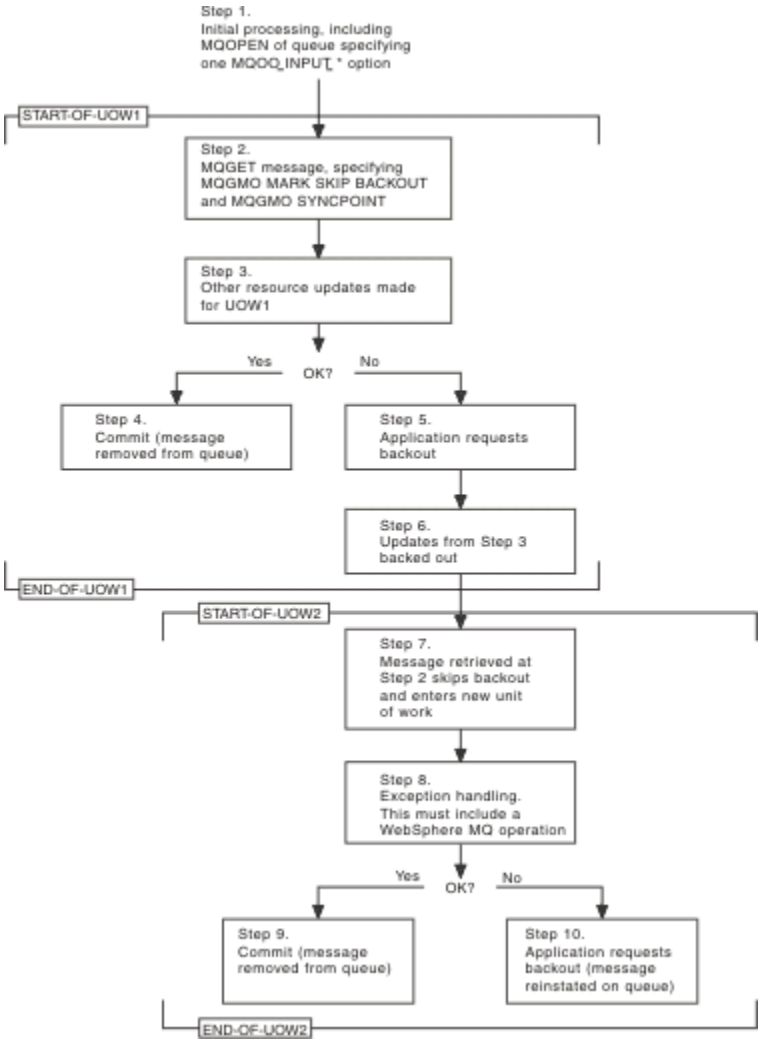


Figura 36. Il backout viene ignorato utilizzando MQGMO_MARK_SKIP_BACKOUT

I passaggi in Figura 36 a pagina 271 sono:

Passo 1
L'elaborazione iniziale si verifica all'interno della transazione, inclusa una chiamata MQOPEN per aprire una coda (specificando una delle opzioni MQOO_INPUT_ * per richiamare i messaggi dalla coda nel passo 2).

Passo 2
MQGET viene richiamato, con MQGMO_SYNCPOINT e MQGMO_MARK_SKIP_BACKOUT. MQGMO_SYNCPOINT è richiesto perché MQGET deve essere all'interno di un'unità di lavoro affinché MQGMO_MARK_SKIP_BACKOUT sia effettivo. In Figura 36 a pagina 271 questa unità di lavoro viene indicata come UOW1.

Passo 3
Altri aggiornamenti di risorse vengono effettuati come parte di UOW1. Queste possono includere ulteriori chiamate MQGET (emesse senza MQGMO_MARK_SKIP_BACKOUT).

Passo 4
Tutti gli aggiornamenti dei passi 2 e 3 vengono completati come richiesto. Il programma applicativo esegue il commit degli aggiornamenti e UOW1 termina. Il messaggio richiamato nel passo 2 viene rimosso dalla coda.

Passo 5
Alcuni degli aggiornamenti dei passi 2 e 3 non vengono completati come richiesto. Il programma applicativo richiede il backout degli aggiornamenti effettuati durante questi passi.

Passo 6
Gli aggiornamenti effettuati nel passo 3 vengono ripristinati.

Passo 7
La richiesta MQGET effettuata al passo 2 ignora il backout e diventa parte di una nuova unità di lavoro, UOW2.

Passo 8
UOW2 esegue la gestione delle eccezioni in risposta al backout di UOW1 . (Ad esempio, una chiamata MQPUT a un altra coda, che indica che si è verificato un problema che ha causato il backout di UOW1 .)

Passo 9
Il passo 8 viene completato come richiesto, il programma applicativo esegue il commit dell'attività e UOW2 termina. Poiché la richiesta MQGET fa parte di UOW2 (vedere il Passo 7), questo commit causa la rimozione del messaggio dalla coda.

Passo 10
Il passo 8 non viene completato come richiesto e il programma applicativo esegue il backout di UOW2. Poiché la richiesta di richiamo del messaggio fa parte di UOW2 (vedere il passo 7), viene eseguito il backout e reintegrato nella coda. È ora disponibile per ulteriori chiamate MQGET emesse da questo o da un altro programma applicativo (allo stesso modo di qualsiasi altro messaggio sulla coda).

Conversione dati applicazione

Quando è necessario, gli MCA convertono il descrittore del messaggio e i dati di intestazione nella serie di caratteri e nella codifica richiesti. L'estremità del collegamento (ovvero, l'MCA locale o l'MCA remoto) può eseguire la conversione.

Quando un'applicazione inserisce i messaggi in una coda, il gestore code locale aggiunge le informazioni di controllo ai descrittori di messaggi per semplificare il controllo dei messaggi quando vengono elaborati dai gestori code e dagli MCA. A seconda dell'ambiente, i campi dei dati di intestazione del messaggio vengono creati nella serie di caratteri e nella codifica del sistema locale.

Quando si spostano i messaggi tra i sistemi, a volte è necessario convertire i dati dell'applicazione nella serie di caratteri e nella codifica richieste dal sistema ricevente. Questa operazione può essere eseguita dall'interno dei programmi applicativi sul sistema di ricezione o dagli MCA sul sistema di invio. Se la conversione dei dati è supportata sul sistema ricevente, utilizzare i programmi applicativi per convertire i dati dell'applicazione, piuttosto che in base alla conversione che si è già verificata sul sistema di invio.

I dati dell'applicazione vengono convertiti all'interno di un programma applicativo quando si specifica l'opzione MQGMO_CONVERT nel campo *Options* della struttura MQGMO passata a una chiamata MQGET e *tutte* le seguenti condizioni sono vere:

- I campi *CodedCharSetId* o *Encoding* impostati nella struttura MQMD associata al messaggio sulla coda differiscono dai campi *CodedCharSetId* o *Encoding* impostati nella struttura MQMD specificata sulla chiamata MQGET.
- Il campo *Format* nella struttura MQMD associata al messaggio non è MQFMT_NONE.
- Il valore *BufferLength* specificato nella chiamata MQGET non è zero.
- La lunghezza dei dati del messaggio è diversa da zero.
- Il gestore code supporta la conversione tra i campi *CodedCharSetId* e *Encoding* specificati nelle strutture MQMD associate al messaggio e alla chiamata MQGET. Consultare CodedCharSetId e Codifica per dettagli sugli identificativi della serie di caratteri codificati e sulle codifiche di macchina supportate.
- Il gestore code supporta la conversione del formato del messaggio. Se il campo *Format* della struttura MQMD associata al messaggio è uno dei formati integrati, il gestore code può convertire il messaggio. Se *Format* non è uno dei formati integrati, è necessario scrivere un'uscita di conversione dati per convertire il messaggio.

Se l'MCA di invio deve convertire i dati, specificare la parola chiave CONVERT (YES) sulla definizione di ogni canale mittente o server per cui è necessaria la conversione. Se la conversione dei dati non riesce,

il messaggio viene inviato alla DLQ sul gestore code di invio e il campo *Feedback* della struttura MQDLH indica il motivo. Se il messaggio non può essere inserito nella DLQ, il canale si chiude e il messaggio non convertito rimane nella coda di trasmissione. La conversione dei dati all'interno delle applicazioni piuttosto che all'invio di MCA evita questa situazione.

Come regola, i dati nel messaggio descritti come dati *carattere* dal formato integrato o dall'uscita di conversione dati vengono convertiti dalla serie di caratteri codificati utilizzata dal messaggio a quella richiesta e i campi *numerici* vengono convertiti nella codifica richiesta.

Per ulteriori dettagli sulle convenzioni di elaborazione della conversione utilizzate durante la conversione dei formati integrati e per informazioni sulla scrittura delle proprie uscite di conversione dati, consultare “Scrittura delle uscite di conversione dati” a pagina 417. Consultare anche Lingue nazionali e Codici macchina per informazioni sulle tabelle di supporto lingua e sulle codifiche macchina supportate.

Conversione dei caratteri di nuova riga EBCDIC

Se è necessario assicurarsi che i dati inviati da una piattaforma EBCDIC a una ASCII siano identici ai dati che si ricevono di nuovo, è necessario controllare la conversione dei caratteri di nuova riga EBCDIC.

È possibile effettuare questa operazione utilizzando uno switch dipendente dalla piattaforma che forza WebSphere MQ ad utilizzare le tabelle di conversione non modificate, ma è necessario essere consapevoli del comportamento incongruente che potrebbe risultare.

Il problema si verifica perché il carattere di nuova riga EBCDIC non viene convertito in maniera congruente tra piattaforme o tabelle di conversione. Di conseguenza, se i dati vengono visualizzati su una piattaforma ASCII, la formattazione potrebbe non essere corretta. Ciò renderebbe difficile, ad esempio, gestire un sistema IBM i in remoto da una piattaforma ASCII utilizzando RUNMQSC.

Consultare Conversione dati per ulteriori informazioni sulla conversione dei dati in formato EBCDIC in formato ASCII.

Visualizzazione dei messaggi su una coda

Utilizzare queste informazioni per informazioni sull'esplorazione dei messaggi su una coda utilizzando la chiamata MQGET.

Per utilizzare la chiamata MQGET per sfogliare i messaggi su una coda:

1. Richiamare MQOPEN per aprire la coda per la ricerca, specificando l'opzione MQOO_BROWSE.
2. Per esaminare il primo messaggio sulla coda, richiamare MQGET con l'opzione MQGMO_BROWSE_FIRST. Per individuare il messaggio desiderato, richiamare ripetutamente MQGET con l'opzione MQGMO_BROWSE_NEXT per esaminare molti messaggi.

   È necessario impostare i campi *MsgId* e *CorrelId* della struttura MQMD su null dopo ogni chiamata MQGET per visualizzare tutti i messaggi.
3. Richiamare MQCLOSE per chiudere la coda.

Il cursore di ricerca

Quando si apre (MQOPEN) una coda per la ricerca, la chiamata stabilisce un cursore di ricerca da utilizzare con le chiamate MQGET che utilizzano una delle opzioni di ricerca. È possibile considerare il cursore di ricerca come un puntatore logico posizionato prima del primo messaggio sulla coda.

È possibile avere più di un cursore di ricerca attivo (da un singolo programma) emettendo diverse richieste MQOPEN per la stessa coda.

Quando si richiama MQGET per l'esplorazione, utilizzare una delle seguenti opzioni nella struttura MQGMO:

MQGMO_BROWSE_FIRST
Richiama una copia del primo messaggio che soddisfa le condizioni specificate nella struttura MQMD.

MQGMO_BROWSE_SUCCESSIVO
Richiama una copia del successivo messaggio che soddisfa le condizioni specificate nella propria struttura MQMD.

MQGMO_BROWSE_MSG_UNDER_CURSOR
Richiama una copia del messaggio attualmente puntato dal cursore, ovvero quello che è stato richiamato l'ultima volta utilizzando l'opzione MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT.

In tutti i casi, il messaggio rimane nella coda.

Quando si apre una coda, il cursore di ricerca viene posizionato logicamente appena prima del primo messaggio sulla coda. Ciò significa che se si effettua la propria chiamata MQGET immediatamente dopo la chiamata MQOPEN, è possibile utilizzare l'opzione MQGMO_BROWSE_NEXT per sfogliare il primo messaggio; non è necessario utilizzare l'opzione MQGMO_BROWSE_FIRST.

L'ordine di copia dei messaggi dalla coda è determinato dall'attributo *MsgDeliverySequence* della coda. (Per ulteriori informazioni, consultare "L'ordine in cui i messaggi vengono richiamati da una coda" a pagina 246.)

- "Code nella sequenza FIFO (first in, first out)" a pagina 274
- "Code in sequenza di priorità" a pagina 274
- "Messaggi senza commit" a pagina 274
- "Modifica in sequenza coda" a pagina 275
- "Utilizzo dell'indice della coda" a pagina 275

Code nella sequenza FIFO (first in, first out)

Il primo messaggio in una coda in questa sequenza è il messaggio che è stato sulla coda più a lungo.

Utilizzare MQGMO_BROWSE_NEXT per leggere i messaggi in modo sequenziale nella coda. Si vedranno tutti i messaggi inseriti nella coda durante la navigazione, poiché una coda in questa sequenza ha i messaggi posizionati alla fine. Quando il cursore riconosce di aver raggiunto la fine della coda, il cursore di ricerca rimane dove si trova e restituisce con MQRC_NO_MSG_AVAILABLE. È quindi possibile lasciarlo in attesa di ulteriori messaggi o ripristinarlo all'inizio della coda con una chiamata MQGMO_BROWSE_FIRST.

Code in sequenza di priorità

Il primo messaggio in una coda in questa sequenza è il messaggio che è stato sulla coda più a lungo e che ha la priorità più alta al tempo in cui viene emessa la chiamata MQOPEN.

Utilizzare MQGMO_BROWSE_NEXT per leggere i messaggi nella coda.

Il cursore di ricerca punta al messaggio successivo, operando dalla priorità del primo messaggio per terminare con il messaggio con la priorità più bassa. Esamina tutti i messaggi inseriti nella coda durante questo periodo di tempo, purché abbiano una priorità uguale o inferiore al messaggio identificato dal cursore di ricerca corrente.

Qualsiasi messaggio inserito nella coda con priorità più alta può essere sfogliato solo da:

- Apertura della coda per la ricerca di nuovo, a quel punto viene stabilito un nuovo cursore di ricerca
- Utilizzo dell'opzione MQGMO_BROWSE_FIRST

Messaggi senza commit

Un messaggio di cui non è stato eseguito il commit non è mai visibile a una ricerca; il cursore di ricerca lo supera.

I messaggi all'interno di un'unità di lavoro non possono essere esaminati fino a quando non viene eseguito il commit dell'unità di lavoro. I messaggi non cambiano la loro posizione sulla coda quando viene eseguito il commit, quindi i messaggi ignorati e non sottoposti a commit non verranno visualizzati, anche quando *sono* sottoposti a commit, a meno che non si utilizzi l'opzione MQGMO_BROWSE_FIRST e non si lavori di nuovo nella coda.

Modifica in sequenza coda

Se la sequenza di consegna del messaggio viene modificata da priorità a FIFO mentre ci sono messaggi sulla coda, l'ordine dei messaggi che sono già accodati non viene modificato. I messaggi aggiunti alla coda successivamente, assumono la priorità predefinita della coda.

Utilizzo dell'indice della coda

Quando si sfoglia una coda indicizzata che contiene solo messaggi con una singola priorità (persistente o non persistente o entrambi), il gestore code utilizza l'indice per sfogliare quando vengono utilizzate determinate forme di ricerca.

Nota: Supportato solo su WebSphere MQ per z/OS.

Una delle seguenti forme di ricerca viene utilizzata quando una coda indicizzata contiene solo messaggi con priorità singola:

1. Se la coda è indicizzata da MSGID, le richieste di ricerca che passano un MSGID nella struttura MQMD vengono elaborate utilizzando l'indice per trovare il messaggio di destinazione.
2. Se la coda è indicizzata da CORRELID, le richieste di ricerca che passano un CORRELID nella struttura MQMD vengono elaborate utilizzando l'indice per trovare il messaggio di destinazione.
3. Se la coda è indicizzata da GROUPID, le richieste di esplorazione che passano un GROUPID nella struttura MQMD vengono elaborate utilizzando l'indice per trovare il messaggio di destinazione.

Se la richiesta di ricerca non passa un MSGID, CORRELID o GROUPID nella struttura MQMD, la coda viene indicizzata e viene restituito un messaggio, è necessario trovare la voce di indice per il messaggio e le informazioni al suo interno utilizzate per aggiornare il cursore di ricerca. Se si utilizza una vasta selezione di valori di indice, ciò non aggiunge un'ulteriore elaborazione significativa alla richiesta di ricerca.

Visualizzazione dei messaggi quando la lunghezza del messaggio è sconosciuta

Per sfogliare un messaggio quando non si conosce la dimensione del messaggio e non si desidera utilizzare i campi *MsgId*, *CorrelId*o *GroupId* per individuare il messaggio, è possibile utilizzare l'opzione MQGMO_BROWSE_MSG_UNDER_CURSOR:

1. Emettere un MQGET con:
   - L'opzione MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT
   - L'opzione MQGMO_ACCEPT_TRUNCATED_MSG
   - Lunghezza buffer zero

   Nota: Se è probabile che un altro programma ricevi lo stesso messaggio, utilizzare anche l'opzione MQGMO_LOCK. MQRC_TRUNCATED_MSG_ACCEPTED deve essere restituito.
2. Utilizzare il *DataLength* restituito per allocare la memoria necessaria.
3. Emettere MQGET con MQGMO_BROWSE_MSG_UNDER_CURSOR.

Il messaggio puntato è l'ultimo che è stato richiamato; il cursore di ricerca non sarà stato spostato. È possibile scegliere di bloccare il messaggio utilizzando l'opzione MQGMO_LOCK o di sbloccare un messaggio bloccato utilizzando l'opzione MQGMO_UNLOCK.

La chiamata non riesce se nessuna opzione MQGET con le opzioni MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT è stata emessa correttamente da quando è stata aperta la coda.

Rimozione di un messaggio visualizzato

È possibile rimuovere dalla coda un messaggio che è già stato sfogliato, purché sia stata aperta la coda per la rimozione dei messaggi e per la navigazione. (È necessario specificare una delle opzioni MQOO_INPUT_ *, così come l'opzione MQOO_BROWSE, nella chiamata MQOPEN.)

Per rimuovere il messaggio, richiamare di nuovo MQGET, ma nel campo *Options* della struttura MQGMO, specificare MQGMO_MSG_UNDER_CURSOR.In questo caso, la chiamata MQGET ignora i campi *MsgId*, *CorrelId*e *GroupId* della struttura MQMD.

Nel periodo tra la navigazione e la rimozione, un altro programma potrebbe aver rimosso i messaggi dalla coda, incluso il messaggio sotto il cursore di esplorazione. In questo caso, la chiamata MQGET restituisce un codice motivo per indicare che il messaggio non è disponibile.

Visualizzazione dei messaggi in ordine logico

"Ordinamento logico e fisico" a pagina 246 spiega la differenza tra l'ordine logico e fisico dei messaggi su una coda. Questa distinzione è particolarmente importante quando si sfoglia una coda, poiché, in generale, i messaggi non vengono eliminati e le operazioni di ricerca non iniziano necessariamente all'inizio della coda.

Se un'applicazione sfoglia i vari messaggi di un gruppo (utilizzando l'ordine logico), è importante che l'ordine logico venga seguito per raggiungere l'inizio del gruppo successivo, poiché l'ultimo messaggio di un gruppo potrebbe verificarsi fisicamente *dopo* il primo messaggio del gruppo successivo. L'opzione MQGMO_LOGICAL_ORDER garantisce che l'ordine logico venga seguito durante la scansione di una coda.

Utilizzare MQGMO_ALL_MSGS_AVAILABLE (o MQGMO_ALL_SEGMENTS_AVAILABLE) con attenzione per le operazioni di ricerca. Considerare il caso dei messaggi logici con MQGMO_ALL_MSGS_AVAILABLE. L'effetto è che un messaggio logico è disponibile solo se sono presenti anche tutti i restanti messaggi nel gruppo. In caso contrario, il messaggio viene trasmesso. Ciò può significare che quando i messaggi mancanti arrivano in seguito, non vengono notate da un'operazione di ricerca successiva.

Ad esempio, se sono presenti i seguenti messaggi logici,

```
Logical message 1 (not last) of group 123
Logical message 1 (not last) of group 456
Logical message 2 (last)     of group 456
```

e viene emessa una funzione di ricerca con MQGMO_ALL_MSGS_AVAILABLE, viene restituito il primo messaggio logico del gruppo 456, lasciando il cursore di ricerca su questo messaggio logico. Se arriva il secondo (ultimo) messaggio del gruppo 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
```

e viene emessa la stessa funzione browse - next, non si nota che il gruppo 123 è ora completo, perché il primo messaggio di questo gruppo è *prima* del cursore browse.

In alcuni casi (ad esempio, se i messaggi vengono richiamati in modo distruttivo quando il gruppo è presente nella sua interezza), è possibile utilizzare MQGMO_ALL_MSGS_AVAILABLE insieme a MQGMO_BROWSE_FIRST. In caso contrario, è necessario ripetere la scansione di esplorazione per prendere nota dei messaggi appena arrivati che sono stati persi; l'emissione di MQGMO_WAIT insieme a MQGMO_BROWSE_NEXT e MQGMO_ALL_MSGS_AVAILABLE non ne tiene conto. Ciò si verifica anche per i messaggi con priorità più elevata che potrebbero arrivare dopo che la scansione dei messaggi è stata completata.

Le sezioni successive esaminano esempi di esplorazione che trattano messaggi non segmentati; i messaggi segmentati seguono principi simili.

Visualizzazione dei messaggi in gruppi
In questo esempio, l'applicazione esamina ogni messaggio sulla coda, in ordine logico.

I messaggi sulla coda potrebbero essere raggruppati. Per i messaggi raggruppati, l'applicazione non desidera avviare l'elaborazione di alcun gruppo fino a quando non sono arrivati tutti i messaggi al suo interno. MQGMO_ALL_MSGS_AVAILABLE è quindi specificato per il primo messaggio nel gruppo; per messaggi successivi nel gruppo, questa opzione non è necessaria.

MQGMO_WAIT viene utilizzato in questo esempio. Tuttavia, anche se l'attesa può essere soddisfatta se arriva un nuovo gruppo, per i motivi riportati in "Visualizzazione dei messaggi in ordine logico" a pagina 276, non viene soddisfatta se il cursore di esplorazione ha già inoltrato il primo messaggio logico in

un gruppo e ora arrivano i restanti messaggi. Tuttavia, l'attesa di un intervallo adeguato assicura che l'applicazione non sia costantemente in loop durante l'attesa di nuovi messaggi o segmenti.

MQGMO_LOGICAL_ORDER viene utilizzato per garantire che la scansione sia in ordine logico.
Ciò è in contrasto con l'esempio distruttivo MQGET, dove poiché ogni gruppo viene rimosso, MQGMO_LOGICAL_ORDER non viene utilizzato quando si cerca il primo (o solo) messaggio in un gruppo.

Si presume che il buffer dell'applicazione sia sempre abbastanza grande da contenere l'intero messaggio, indipendentemente dal fatto che il messaggio sia stato segmentato o meno. MQGMO_COMPLETE_MSG viene quindi specificato su ogni MQGET.

Di seguito viene riportato un esempio di esplorazione dei messaggi logici in un gruppo:

```
/* 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 */
   ...
```

Il gruppo viene ripetuto fino a quando non viene restituito MQRC_NO_MSG_AVAILABLE.

Esplorazione e recupero in modo distruttivo
In questo esempio, l'applicazione sfoglia ciascuno dei messaggi logici all'interno di un gruppo, prima di decidere se richiamare tale gruppo in modo distruttivo.

La prima parte di questo esempio è simile alla precedente. Tuttavia, in questo caso, dopo aver sfogliato un intero gruppo, decidiamo di tornare indietro e recuperarlo in modo distruttivo.

Poiché ogni gruppo viene rimosso in questo esempio, MQGMO_LOGICAL_ORDER non viene utilizzato durante la ricerca del primo o unico messaggio in un gruppo.

Di seguito viene riportato un esempio di esplorazione e recupero distruttivo:

```
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 */
         ...
```

Evitare la consegna ripetuta dei messaggi visualizzati

Utilizzando determinate opzioni di apertura e di ricezione messaggi, è possibile contrassegnare i messaggi come esplorati in modo che non vengano richiamati di nuovo dalle applicazioni correnti o da altre applicazioni correlate. I messaggi possono essere decontrassegnati esplicitamente o automaticamente per renderli nuovamente disponibili per la navigazione.

Se si sfogliano i messaggi su una coda, è possibile richiamarli in un ordine diverso rispetto a quello in cui li si richiamerebbe se li si ottenesse in modo distruttivo. In particolare, è possibile sfogliare lo stesso messaggio più volte, il che non è possibile se viene rimosso dalla coda. Per evitare ciò, è possibile *contrassegnare* i messaggi come visualizzati ed evitare di richiamare i messaggi contrassegnati. A volte viene indicato come *sfoglia con contrassegno*. Per contrassegnare i messaggi sfogliati, utilizzare l'opzione di richiamo del messaggio MQGMO_MARK_BROWSE_HANDLE e per richiamare solo i messaggi non contrassegnati, utilizzare MQGMO_UNMARKED_BROWSE_MSG. Se si utilizza la combinazione delle opzioni MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG e MQGMO_MARK_BROWSE_HANDLE e si emettono MQGET ripetute, si richiamerà ogni messaggio sulla coda a turno. Ciò impedisce la consegna ripetuta dei messaggi anche se MQGMO_BROWSE_FIRST viene utilizzato per assicurare che i messaggi non vengano ignorati. Questa combinazione di opzioni può essere rappresentata dalla singola costante MQGMO_BROWSE_HANDLE. Quando non ci sono messaggi sulla coda che non sono stati esaminati, viene restituito MQRC_NO_MSG_AVAILABLE.

Se più applicazioni stanno sfogliando la stessa coda, possono aprire la coda con le opzioni MQOO_CO_OP e MQOO_BROWSE. L'handle dell'oggetto restituito da ogni MQOPEN è considerato parte di un gruppo cooperante. Qualsiasi messaggio restituito da una chiamata MQGET che specifichi l'opzione MQGMO_MARK_BROWSE_CO_OP viene considerato come contrassegnato per questa serie di handle cooperante.

Se un messaggio è stato contrassegnato per un certo periodo di tempo, può essere automaticamente annullato dal gestore code e reso disponibile per una nuova esplorazione. L'attributo del gestore code MsgMarkBrowseInterval fornisce il tempo, in millisecondi, per cui un messaggio deve rimanere contrassegnato per la serie di handle correlati. Un MsgMarkBrowseInterval di -1 significa che i messaggi non vengono mai contrassegnati automaticamente.

Quando il singolo processo o la serie di processi cooperativi contrassegnano i messaggi vengono arrestati, tutti i messaggi contrassegnati non vengono contrassegnati.

Esempi di navigazione cooperativa

È possibile eseguire più copie di un'applicazione dispatcher per sfogliare i messaggi su una coda e avviare un consumer in base al contenuto di ciascun messaggio. In ogni dispatcher, aprire la coda con MQOO_CO_OP. Ciò indica che i dispatcher cooperano e saranno a conoscenza dei rispettivi messaggi contrassegnati. Ogni dispatcher effettua quindi chiamate MQGET ripetute, specificando le opzioni MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG e MQGMO_MARK_BROWSE_CO_OP (è possibile utilizzare la singola costante MQGMO_BROWSE_CO_OP per rappresentare questa combinazione di opzioni). Ogni applicazione dispatcher richiama quindi solo i messaggi che non sono stati già contrassegnati da altri dispatcher cooperanti. Il dispatcher inizializza un consumer e trasmette il MsgToken restituito da MQGET al consumer, che riceve in maniera distruttiva il messaggio dalla coda. Se il consumer esegue il backout di MQGET del messaggio, il messaggio è disponibile per la ridistribuzione da parte di uno dei browser, poiché non è più contrassegnato. Se il consumer non esegue un MQGET sul messaggio, dopo il passaggio di MsgMarkBrowseInterval , il gestore code annulla il contrassegno del messaggio per la serie di handle correlati e può essere ridistribuito.

Piuttosto che più copie della stessa applicazione dispatcher, è possibile disporre di un numero di applicazioni dispatcher differenti che sfogliano la coda, ciascuna adatta per l'elaborazione di un sottoinsieme di messaggi sulla coda. In ogni dispatcher, aprire la coda con MQOO_CO_OP. Ciò indica che i dispatcher cooperano e saranno a conoscenza dei rispettivi messaggi contrassegnati.

- Se l'ordine di elaborazione dei messaggi per un singolo dispatcher è importante, ogni dispatcher effettua chiamate MQGET ripetute, specificando le opzioni MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG e MQGMO_MARK_BROWSE_HANDLE (o MQGMO_BROWSE_HANDLE). Se il messaggio sfogliato è adatto per l'elaborazione da parte di questo dispatcher, effettua una chiamata MQGET specificando MQMO_MATCH_MSG_TOKEN, MQGMO_MARK_BROWSE_CO_OP e il MsgToken restituito dalla precedente chiamata MQGET. Se la chiamata ha esito positivo, il dispatcher inizializza il consumer, passando il MsgToken .
- Se l'ordine di elaborazione dei messaggi non è importante e si prevede che il dispatcher elabori la maggior parte dei messaggi rilevati, utilizzare le opzioni MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG e MQGMO_MARK_BROWSE_CO_OP (o MQGMO_BROWSE_CO_OP). Se il dispatcher sfoglia un messaggio che non può elaborare, annulla il contrassegno del messaggio richiamando MQGET con l'opzione MQMO_MATCH_MSG_TOKEN, MQGMO_UNMARK_BROWSE_CO_OP e il MsgToken precedentemente restituito.

Alcuni casi in cui la chiamata MQGET non riesce

Se alcuni attributi di una coda vengono modificati utilizzando l'opzione FORCE su un comando tra l'emissione di una chiamata MQOPEN e MQGET, la chiamata MQGET ha esito negativo e restituisce il codice motivo MQRC_OBJECT_CHANGED.

Il gestore code contrassegna l'handle dell'oggetto come non più valido. Ciò si verifica anche se le modifiche si applicano a qualsiasi coda in cui il nome della coda si risolve. Gli attributi che influenzano l'handle in questo modo sono riportati nella descrizione della chiamata MQOPEN in MQOPEN. Se la chiamata restituisce il codice motivo MQRC_OBJECT_CHANGED, chiudere la coda, riaprirla, quindi provare a richiamare nuovamente un messaggio.

Se le operazioni di richiamo sono inibite per una coda da cui si sta tentando di richiamare i messaggi (o qualsiasi coda in cui il nome della coda si risolve), la chiamata MQGET ha esito negativo e restituisce il codice motivo MQRC_GET_INHIBITED. Ciò si verifica anche se si sta utilizzando la chiamata MQGET per la navigazione. Potrebbe essere possibile ricevere un messaggio correttamente se si tenta la chiamata MQGET in un secondo momento, se la progettazione dell'applicazione è tale che altri programmi modificano regolarmente gli attributi delle code.

Se una coda dinamica (temporanea o permanente) è stata eliminata, le chiamate MQGET che utilizzano un handle di oggetto precedentemente acquisito hanno esito negativo e restituiscono il codice motivo MQRC_Q_DELETED.

Scrittura di applicazioni di pubblicazione / sottoscrizione

Iniziare a scrivere applicazioni WebSphere MQ di pubblicazione / sottoscrizione.

Per una panoramica dei concetti di pubblicazione / sottoscrizione, consultare Introduzione alla messaggistica di pubblicazione / sottoscrizione di WebSphere MQ.

Consultare i seguenti argomenti per informazioni sulla scrittura di diversi tipi di applicazioni di pubblicazione / sottoscrizione:

- “Scrittura delle applicazioni del publisher” a pagina 280
- “Scrittura delle applicazioni del sottoscrittore” a pagina 287
- “Cicli di vita di pubblicazione / sottoscrizione” a pagina 305
- “Proprietà dei messaggi di pubblicazione / sottoscrizione” a pagina 310
- “Ordinamento dei messaggi” a pagina 312
- “Intercettazione delle pubblicazioni” a pagina 312
- “Opzioni di pubblicazione” a pagina 320
- “Opzioni di sottoscrizione” a pagina 320

Concetti correlati

“Concetti dello sviluppo di applicazioni” a pagina 8

È possibile utilizzare una scelta di linguaggi procedurali o orientati agli oggetti per scrivere applicazioni IBM WebSphere MQ . Utilizzare i link in questo argomento per informazioni sui concetti di IBM WebSphere MQ che sono utili per sviluppatori di applicazioni.

“Scelta del linguaggio di programmazione da utilizzare” a pagina 79
Utilizza queste informazioni per scoprire i linguaggi di programmazione e i framework supportati da IBM WebSphere MQe alcune considerazioni per utilizzarli.

“Progettazione di applicazioni IBM WebSphere MQ” a pagina 90
Una volta stabilito il modo in cui le applicazioni possono sfruttare le piattaforme e gli ambienti disponibili, è necessario decidere come utilizzare le funzioni offerte da WebSphere MQ.

“Programmi di WebSphere MQ di esempio” a pagina 97
Utilizzare questa raccolta di argomenti per informazioni sui programmi WebSphere MQ di esempio su piattaforme differenti.

“Scrittura di un'applicazione di accodamento” a pagina 195
Utilizzare queste informazioni per informazioni sulla scrittura delle applicazioni di accodamento, la connessione e la disconnessione da un gestore code, la pubblicazione / sottoscrizione e l'apertura e la chiusura di oggetti.

“Scrittura delle applicazioni client” a pagina 354
Informazioni necessarie per scrivere le applicazioni client su WebSphere MQ.

“Utilizzo dei servizi Web in WebSphere MQ” a pagina 948
È possibile sviluppare applicazioni IBM WebSphere MQ per servizi Web utilizzando il trasporto IBM WebSphere MQ per SOAP o il bridge IBM WebSphere MQ per HTTP.

“Creazione di un'applicazione IBM WebSphere MQ” a pagina 431
Utilizzare queste informazioni per informazioni sulla generazione di un'applicazione IBM WebSphere MQ su piattaforme differenti.

“Gestione degli errori del programma” a pagina 550
Queste informazioni spiegano gli errori associati alle chiamate MQI delle applicazioni quando effettua una chiamata o quando il suo messaggio viene consegnato alla destinazione finale.

Scrittura delle applicazioni del publisher

Inizia a scrivere applicazioni di pubblicazione studiando due esempi. Il primo è modellato il più strettamente possibile su un'applicazione point to point inserendo i messaggi in una coda e il secondo dimostra la creazione dinamica degli argomenti - un modello più comune per le applicazioni publisher.

Scrivere una semplice applicazione publisher WebSphere MQ è come scrivere un'applicazione point to point WebSphere MQ che inserisce i messaggi in una coda (Tabella 41 a pagina 280). La differenza è che i messaggi MQPUT vengono inviati a un argomento, non a una coda.

Tabella 41. Modello di programma di pubblicazione / sottoscrizione di WebSphere MQ .

Fase	**Chiamata MQ punto a punto**	**Pubblica chiamata MQ**
Connetterti a un gestore code	MQCONN	MQCONN
Apri coda	MQOPEN	
Apri argomento		MQOPEN
Inserisci messaggi	MQPUT	MQPUT
Chiudi argomento		MQCLOSE
Chiudi coda	MQCLOSE	
Disconnetti dal gestore code	MQDISC	MQDISC

Per renderlo concreto, ci sono due esempi di applicazioni per pubblicare i prezzi delle azioni. Nel primo esempio (“Esempio 1: pubblicazione di un argomento fisso” a pagina 281), modellato strettamente sull'inserimento di messaggi in una coda, l'amministratore crea una definizione di argomento in modo simile alla creazione di una coda. Il programmatore codifica MQPUT per scrivere messaggi nell'argomento invece di scriverli in coda. Nel secondo esempio (“Esempio 2: pubblicazione di un argomento variabile” a pagina 284), il modello di interazione del programma con WebSphere MQ è simile. La differenza è che il programmatore fornisce l'argomento in cui viene scritto il messaggio, piuttosto che l'amministratore. In pratica, ciò di solito significa che la stringa di argomenti è un contenuto definito o fornito da un'altra origine, come ad esempio l'input umano tramite un browser.

Concetti correlati

“Scrittura delle applicazioni del sottoscrittore” a pagina 287
Iniziare con la scrittura delle applicazioni del sottoscrittore studiando tre esempi: un'applicazione WebSphere MQ che utilizza i messaggi da una coda, un'applicazione che crea una sottoscrizione e non richiede alcuna conoscenza dell'accodamento e, infine, un esempio che utilizza sia l'accodamento che le sottoscrizioni.

Riferimenti correlati

DEFINISCI ARGOMENTO
VISUALIZZA ARGOMENTO
VISUALIZZA TPSTATUS

Esempio 1: pubblicazione di un argomento fisso

Un programma WebSphere MQ per illustrare la pubblicazione in un argomento definito amministrativamente.

Nota: Lo stile di codifica compatto è progettato per la leggibilità e non per l'utilizzo in produzione.

Vedere l'output in Figura 38 a pagina 282

```
#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);
}
```

Figura 37. Publisher WebSphere MQ semplice per un argomento fisso.

```
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
```

Figura 38. Esempio di output del primo programma di pubblicazione

Le seguenti righe di codice selezionate illustrano gli aspetti della scrittura di un'applicazione publisher per WebSphere MQ.

char topicNameDefault[] = "IBMSTOCKPRICE";
Nel programma viene definito un nome argomento predefinito. È possibile sovrascriverlo fornendo il nome di un oggetto argomento differente come primo argomento del programma.

MQCHAR resTopicStr[151];
`resTopicStr` è puntato da `td.ResObjectString.VSPtr` e viene utilizzato da `MQOPEN` per restituire la stringa di argomenti risolta. Aumentare la lunghezza di `resTopicStr` di uno rispetto alla lunghezza passata in `td.ResObjectString.VSBufSize` per dare spazio per la terminazione null.

memset (resTopicStr, 0, sizeof(resTopicStr));
Inizializza `resTopicStr` su valori null per garantire che la stringa di argomenti risolta restituita in `MQCHARV` sia terminata con valore null.

td.ObjectType = MQOT_TOPIC
Esiste un nuovo tipo di oggetto per la pubblicazione / sottoscrizione: l' *oggetto argomento*.

td.Version = MQOD_VERSION_4;
Per utilizzare il nuovo tipo di oggetto, è necessario utilizzare almeno la *versione 4* del descrittore oggetto.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
`topicName` è il nome di un oggetto argomento, a volte denominato oggetto argomento di gestione. Nell'esempio l'oggetto argomento deve essere creato in anticipo, utilizzando WebSphere MQ Explorer o questo comando MQSC,

```
DEFINE TOPIC(IBMSTOCKPRICE) TOPICSTR(NYSE/IBM/PRICE) REPLACE;
```

td.ResObjectString.VSPtr = resTopicStr;
La stringa di argomenti risolta viene rimandata nel `printf` finale nel programma. Impostare la struttura `MQCHARV ResObjectString` per WebSphere MQ per restituire la stringa risolta al programma.

MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
Aprire l'argomento per l'emissione, proprio come aprire una coda per l'emissione.

pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
Si desidera che i nuovi sottoscrittori siano in grado di ricevere la pubblicazione e specificando `MQPMO_RETAIN` nel publisher, quando si avvia un sottoscrittore riceve l'ultima pubblicazione, pubblicata prima dell'avvio del sottoscrittore, come prima pubblicazione corrispondente. L'alternativa consiste nel fornire ai sottoscrittori le pubblicazioni pubblicate solo dopo l'avvio del sottoscrittore. Inoltre un sottoscrittore ha la possibilità di rifiutare di ricevere una pubblicazione conservata specificando `MQSO_NEW_PUBLICATIONS_ONLY` nella propria sottoscrizione.

MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
Aggiungere 1 alla lunghezza della stringa inoltrata a `MQPUT` per passare il carattere di terminazione null a WebSphere MQ come parte del buffer di messaggi.

Che cosa dimostra il primo esempio? L'esempio imita il più possibile il modello tradizionale provato e testato per la scrittura punto a punto dei programmi WebSphere MQ . Una funzione importante del modello di programmazione di WebSphere MQ è che il programmatore non è interessato a dove vengono inviati i messaggi. L'attività del programmatore consiste nel connettersi a un gestore code e trasmettergli i messaggi che devono essere distribuiti ai destinatari. Nel paradigma point-to-point, il programmatore apre una coda (probabilmente una coda alias) configurata dall'amministratore. La coda alias instrada i messaggi a una coda di destinazione, sul gestore code locale o su un gestore code remoto. Mentre i messaggi sono in attesa di essere consegnati, vengono memorizzati nelle code tra l'origine e la destinazione.

Nel modello di pubblicazione / sottoscrizione, invece di aprire una coda, il programmatore apre un argomento. In questo esempio, l'argomento è associato a una stringa di argomenti da un amministratore. Il gestore code inoltra la pubblicazione, utilizzando le code, ai sottoscrittori locali o remoti che hanno sottoscrizioni che corrispondono alla stringa argomento della pubblicazione. Se le pubblicazioni sono

conservate, il gestore code conserva l'ultima copia della pubblicazione, anche se al momento non dispone di sottoscrittori. La pubblicazione conservata è disponibile per l'inoltro ai sottoscrittori futuri. L'applicazione publisher non ha alcun ruolo nella selezione o nell'instradamento della pubblicazione verso una destinazione; il suo compito è creare e inserire le pubblicazioni negli argomenti definiti dall'amministratore.

Questo esempio di argomento fisso è atipico di molte applicazioni di pubblicazione / sottoscrizione: è statico. Richiede un amministratore per definire le stringhe degli argomenti e modificare gli argomenti pubblicati. Generalmente, le applicazioni di pubblicazione / sottoscrizione devono conoscere alcune o tutte le strutture ad albero degli argomenti. Forse gli argomenti cambiano frequentemente, o forse anche se gli argomenti non cambiano molto, il numero di combinazioni di argomenti è elevato ed è troppo oneroso per un amministratore definire un nodo di argomenti per ogni stringa di argomenti su cui potrebbe essere necessario pubblicare. Forse le stringhe argomento non sono note prima della pubblicazione; un'applicazione publisher potrebbe utilizzare le informazioni del contenuto della pubblicazione per specificare una stringa argomento oppure potrebbe avere informazioni sulle stringhe argomento da pubblicare da un'altra origine, come ad esempio l'input umano da un browser. Per fornire stili di pubblicazione più dinamici, l'esempio successivo mostra come creare gli argomenti in modo dinamico, come parte dell'applicazione publisher.

Gli argomenti uniscono editori e sottoscrittori. La progettazione delle regole o dell'architettura per la denominazione degli argomenti e la loro organizzazione in strutture ad albero degli argomenti è un passo importante nello sviluppo di una soluzione di pubblicazione / sottoscrizione. Esaminare attentamente la misura in cui l'organizzazione della struttura ad albero degli argomenti si collega ai programmi di pubblicazione e sottoscrizione e li collega al contenuto della struttura ad albero degli argomenti. Porsi la domanda se le modifiche nella struttura ad albero degli argomenti influiscono sulle applicazioni del publisher e del sottoscrittore e su come è possibile ridurre al minimo l'effetto. Integrato nell'architettura del modello di pubblicazione / sottoscrizione di WebSphere MQ è la nozione di un oggetto argomento di gestione che fornisce la parte root, o la struttura secondaria root, di un argomento. L'oggetto dell'argomento fornisce l'opzione di definire la parte root della struttura ad albero dell'argomento in modo amministrativo che semplifica la programmazione e le operazioni dell'applicazione e, di conseguenza, migliora la manutenibilità. Ad esempio, se si distribuiscono più applicazioni di pubblicazione / sottoscrizione che hanno strutture ad albero degli argomenti isolate, definendo amministrativamente la parte root della struttura ad albero degli argomenti, è possibile garantire l'isolamento delle strutture ad albero degli argomenti, anche se non vi è coerenza nelle convenzioni di denominazione degli argomenti adottate dalle diverse applicazioni.

In pratica, le applicazioni degli editori coprono uno spettro che va dall'utilizzo esclusivo di argomenti fissi, come in questo esempio, e argomenti variabili, come nel prossimo. “Esempio 2: pubblicazione di un argomento variabile” a pagina 284 dimostra anche la combinazione dell'utilizzo di argomenti e stringhe di argomento.

Concetti correlati

“Esempio 2: pubblicazione di un argomento variabile” a pagina 284
Un programma Websphere MQ per illustrare la pubblicazione in un argomento definito in modo programmatico.

“Scrittura delle applicazioni del sottoscrittore” a pagina 287
Iniziare con la scrittura delle applicazioni del sottoscrittore studiando tre esempi: un'applicazione WebSphere MQ che utilizza i messaggi da una coda, un'applicazione che crea una sottoscrizione e non richiede alcuna conoscenza dell'accodamento e, infine, un esempio che utilizza sia l'accodamento che le sottoscrizioni.

Esempio 2: pubblicazione di un argomento variabile

Un programma Websphere MQ per illustrare la pubblicazione in un argomento definito in modo programmatico.

Nota: Lo stile di codifica compatto è progettato per la leggibilità e non per l'utilizzo in produzione.

Consultare l'output in Figura 40 a pagina 286.

```
#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);
}
```

Figura 39. Publisher WebSphere MQ semplice per un argomento variabile.

```
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
```

Figura 40. Esempio di output del secondo publisher

Ci sono alcuni punti da notare su questo esempio.

char topicNameDefault[] = "STOCKS";
Il nome argomento predefinito STOCKS definisce parte della stringa di argomenti. È possibile sovrascrivere questo nome argomento fornendolo come primo argomento al programma oppure eliminare l'utilizzo del nome argomento fornendo / come primo parametro.

char topicString[101] = "IBM/PRICE";
IBM/PRICE è la stringa argomento predefinita. È possibile sovrascrivere questa stringa di argomenti fornendolo come secondo argomento al programma.

Il gestore code combina la stringa di argomento fornita dall'oggetto argomento STOCKS , "NYSE", con la stringa di argomento fornita dal programma "IBM/PRICE" e inserisce un "/" tra le due stringhe di argomento. Il risultato è la stringa argomento risolta "NYSE/IBM/PRICE". La stringa di argomenti risultante è uguale a quella definita nell'oggetto argomento IBMSTOCKPRICE e ha esattamente lo stesso effetto.

L'oggetto argomento di gestione associato alla stringa argomento risolta non è necessariamente lo stesso oggetto argomento passato a MQOPEN dal publisher. WebSphere MQ utilizza la struttura ad albero implicita nella stringa argomento risolta per determinare quale oggetto argomento di gestione definisce gli attributi associati alla pubblicazione.

Si supponga che vi siano due oggetti argomento A e B, e A definisce l'argomento "a"e B definisce l'argomento "a/b" (Figura 41 a pagina 287). Se il programma di pubblicazione fa riferimento all'oggetto argomento A e fornisce la stringa argomento "b", risolvendo l'argomento nella stringa argomento "a/b", la pubblicazione eredita le proprietà dall'oggetto argomento B poiché l'argomento corrisponde alla stringa argomento "a/b" definita per B.

if (strcmp(argv[1],"/"))
argv[1] è il topicNamefornito facoltativamente. "/" non è valido come nome argomento; in questo caso indica che non esiste alcun nome argomento e che la stringa argomento viene fornita interamente dal programma. L'output in Figura 40 a pagina 286 mostra l'intera stringa argomento fornita dinamicamente dal programma.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
Per il caso predefinito, il topicName facoltativo deve essere creato in anticipo, utilizzando WebSphere MQ Explorer o questo comando MQSC:

```
DEFINE TOPIC(STOCKS) TOPICSTR(NYSE) REPLACE;
```

td.ObjectString.VSPtr = topicString;
La stringa di argomenti è un campo MQCHARV nel descrittore di argomento

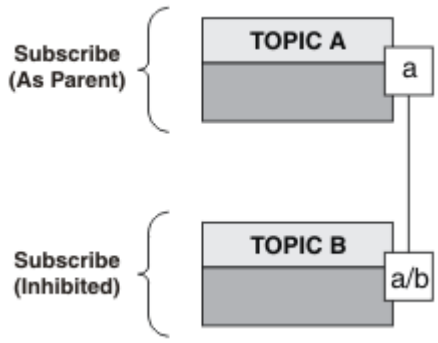


Figura 41. Associazioni oggetto argomento

Che cosa dimostra il secondo esempio? Anche se il codice è molto simile al primo esempio - effettivamente ci sono solo due righe di differenza - il risultato è un programma significativamente diverso dal primo. Il programmatore controlla le destinazioni a cui vengono inviate le pubblicazioni. Insieme all'input dell'amministratore minimo utilizzato per progettare le applicazioni del sottoscrittore, non è necessario che siano predefiniti argomenti o code per instradare le pubblicazioni dai publisher ai sottoscrittori.

Nel paradigma della messaggistica point-to-point, le code devono essere definite prima che i messaggi siano in grado di fluire. Per la pubblicazione / sottoscrizione, non lo fanno, anche se WebSphere MQ implementa la pubblicazione / sottoscrizione utilizzando il sistema di accodamento sottostante; i vantaggi di una consegna garantita, della transazionalità e di un accoppiamento debole associati alla messaggistica e all'accodamento vengono ereditati dalle applicazioni di pubblicazione / sottoscrizione.

Un progettista deve decidere se i programmi di pubblicazione e di sottoscrizione devono essere a conoscenza della struttura ad albero degli argomenti sottostante o meno e se i programmi di sottoscrizione sono a conoscenza dell'accodamento o meno. Esaminare le applicazioni di esempio del sottoscrittore. Sono progettati per essere utilizzati con gli esempi del publisher, generalmente la pubblicazione e la sottoscrizione a `NYSE/IBM/PRICE`.

Concetti correlati

“Esempio 1: pubblicazione di un argomento fisso” a pagina 281
Un programma WebSphere MQ per illustrare la pubblicazione in un argomento definito amministrativamente.

“Scrittura delle applicazioni del sottoscrittore” a pagina 287
Iniziare con la scrittura delle applicazioni del sottoscrittore studiando tre esempi: un'applicazione WebSphere MQ che utilizza i messaggi da una coda, un'applicazione che crea una sottoscrizione e non richiede alcuna conoscenza dell'accodamento e, infine, un esempio che utilizza sia l'accodamento che le sottoscrizioni.

Scrittura delle applicazioni del sottoscrittore

Iniziare con la scrittura delle applicazioni del sottoscrittore studiando tre esempi: un'applicazione WebSphere MQ che utilizza i messaggi da una coda, un'applicazione che crea una sottoscrizione e non richiede alcuna conoscenza dell'accodamento e, infine, un esempio che utilizza sia l'accodamento che le sottoscrizioni.

In Tabella 42 a pagina 288 vengono elencati i tre stili di consumer o sottoscrittore, insieme alle sequenze di chiamate della funzione WebSphere MQ che li caratterizzano.

1. Il primo stile, MQ Publication Consumer, è identico a un programma MQ punto a punto che esegue solo MQGET. L'applicazione non è a conoscenza del fatto che sta consumando pubblicazioni - si tratta semplicemente di leggere i messaggi da una coda. La sottoscrizione che fa in modo che le pubblicazioni vengano instradate alla coda viene creata amministrativamente utilizzando WebSphere MQ Explorer o un comando.

2. Il secondo stile è il modello preferito per la maggior parte delle applicazioni del sottoscrittore. L'applicazione sottoscrittore crea la sottoscrizione e ottiene le pubblicazioni. La gestione della coda viene eseguita dal gestore code.
3. Nel terzo stile, l'applicazione del sottoscrittore sceglie di aprire e chiudere la coda sottostante utilizzata per le pubblicazioni e di emettere sottoscrizioni per riempire la coda con le pubblicazioni.

Un modo per comprendere questi stili è studiare i programmi C di esempio elencati in Tabella 42 a pagina 288 per ognuno degli stili. Gli esempi sono progettati per essere eseguiti insieme all'esempio del publisher trovato in “Scrittura delle applicazioni del publisher” a pagina 280 .

Tabella 42. Confronto tra modelli di programma WebSphere MQ punto a punto e sottoscrizione.

Fase	Utente del messaggio MQ	**“Esempio 1: consumer di pubblicazione MQ” a pagina 288**	**“Esempio 2: sottoscrittore MQ gestito” a pagina 291**	**“Esempio 3: sottoscrittore MQ non gestito” a pagina 296**
Connetterti a un gestore code	MQCONN	MQCONN	MQCONN	MQCONN
Apri coda	MQOPEN	MQOPEN		MQOPEN
Sottoscrivi			MQSUB	MQSUB
Ottieni messaggi	MQGET	MQGET	MQGET	MQGET
Chiudi coda	MQCLOSE	MQCLOSE	(MQCLOSE)	MQCLOSE
Chiudi sottoscrizione			MQCLOSE	MQCLOSE
Disconnetti dal gestore code	MQDISC	MQDISC	MQDISC	MQDISC

L'utilizzo di MQCLOSE è sempre facoltativo, per rilasciare le risorse, passare le opzioni MQCLOSE o solo per la simmetria con MQOPEN. Poiché è improbabile che sia necessario specificare le opzioni MQCLOSE quando la coda di sottoscrizione viene chiusa nel caso del sottoscrittore (subscriber) MQ gestito e l'argomento della simmetria non è rilevante, la coda di sottoscrizione non viene chiusa esplicitamente in Esempio 2: Sottoscrittore (subscriber) MQ gestito .

Un altro modo per comprendere i modelli di applicazione di pubblicazione / sottoscrizione è esaminare troppo le interazioni tra le diverse entità coinvolte. I diagrammi di sequenza Lifeline o UML sono un buon modo per studiare le interazioni. Tre esempi di lifeline sono descritti in “Cicli di vita di pubblicazione / sottoscrizione” a pagina 305.

Esempio 1: consumer di pubblicazione MQ

Il consumer di pubblicazione di MQ è un consumer di messaggi IBM WebSphere MQ che non sottoscrive gli argomenti.

Per creare la sottoscrizione e la coda di pubblicazione per questo esempio, eseguire i comandi riportati di seguito oppure definire gli oggetti utilizzando WebSphere MQ Explorer.

```
DEFINE QLOCAL(STOCKTICKER) REPLACE;
DEFINE SUB(IBMSTOCKPRICESUB) DEST(STOCKTICKER) TOPICOBJ(IBMSTOCKPRICE) REPLACE;
```

La sottoscrizione IBMSTOCKPRICESUB fa riferimento all'oggetto argomento IBMSTOCK creato per l'esempio del publisher e la coda locale STOCKTICKER. L'oggetto argomento IBMSTOCK definisce la stringa di argomenti utilizzata nella sottoscrizione, NYSE/IBM/PRICE. Si noti che l'oggetto argomento e la coda utilizzata per ricevere le pubblicazioni devono essere definiti prima della creazione della sottoscrizione.

Esistono diversi facet utili per il pattern del consumer della pubblicazione MQ :

1. Multiprocessing: condivisione del lavoro di lettura delle pubblicazioni. Le pubblicazioni vanno tutte nella singola coda associata all'argomento della sottoscrizione. Più utenti possono aprire la coda utilizzando `MQOO_INPUT_SHARED`.
2. Sottoscrizioni gestite centralmente. Le applicazioni non costruiscono i propri argomenti di sottoscrizione o sottoscrizioni; l'amministratore è responsabile di dove vengono inviate le pubblicazioni.
3. Concentrazione sottoscrizione: più sottoscrizioni differenti possono essere inviate a una singola coda.
4. Durata della sottoscrizione: la coda riceve tutte le pubblicazioni indipendentemente dal fatto che i consumer siano attivi o meno.
5. Migrazione e coesistenza: il codice consumer funziona allo stesso modo per uno scenario point-to-point e di pubblicazione / sottoscrizione.

La sottoscrizione crea una relazione tra la stringa argomento `NYSE/IBM/PRICE` e la coda `STOCKTICKER`. Le pubblicazioni, inclusa qualsiasi pubblicazione attualmente conservata, vengono inoltrate a `STOCKTICKER` dal momento in cui viene creata la sottoscrizione.

Una sottoscrizione creata in modo amministrativo può essere gestita o non gestita. Una sottoscrizione gestita diventa effettiva non appena è stata creata, proprio come una sottoscrizione non gestita. Non tutti i facet pattern sono disponibili per una sottoscrizione gestita. Vedi “Esempio 3: sottoscrittore MQ non gestito” a pagina 296

Nota: Lo stile di codifica compatto è progettato per la leggibilità e non per l'utilizzo in produzione.

I risultati sono riportati in Figura 43 a pagina 290.

```
#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);
}
```

Figura 42. Consumer della pubblicazione 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
```

Figura 43. Output dal consumer della pubblicazione MQ

Ci sono un paio di suggerimenti di programmazione di linguaggio *C* standard di WebSphere MQ :

memset(publication, 0, sizeof(publicationBuffer));
Assicurarsi che il messaggio abbia un valore null finale per una formattazione semplice utilizzando `printf`. L'esempio publisher include il valore null finale nel buffer di messaggi passato a `MQPUT` aggiungendo `1` a `strlen(publication)`. L'impostazione dei buffer `MQCHAR` su null è un buon stile di programmazione per i programmi IBM WebSphere MQ C che utilizzano i buffer per memorizzare le stringhe, garantendo che un valore null segua un array di caratteri che non riempia completamente il buffer.

MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen, &CompCode, &Reason);
Riservare un valore null alla fine del buffer di messaggi per assicurarsi che il messaggio restituito abbia un valore null finale nel caso in cui `"if (messlen == strlen(publication));"` sia true. Questo suggerimento integra quello precedente e garantisce che sia presente almeno un valore null in `publicationBuffer` non sovrascritto dal contenuto di `publication`.

Concetti correlati

“Esempio 2: sottoscrittore MQ gestito” a pagina 291
Il sottoscrittore MQ gestito è il modello preferito per la maggior parte delle applicazioni del sottoscrittore. L'esempio richiede *nessuna* definizione di gestione di code, argomenti o sottoscrizioni.

“Esempio 3: sottoscrittore MQ non gestito” a pagina 296
Il sottoscrittore non gestito è una classe importante dell'applicazione del sottoscrittore. Con esso, si combinano i vantaggi della pubblicazione / sottoscrizione con il *controllo* dell'accodamento e dell'utilizzo delle pubblicazioni. L'esempio illustra diversi modi di combinazione di sottoscrizioni e code.

“Scrittura delle applicazioni del publisher” a pagina 280
Inizia a scrivere applicazioni di pubblicazione studiando due esempi. Il primo è modellato il più strettamente possibile su un'applicazione point to point inserendo i messaggi in una coda e il secondo dimostra la creazione dinamica degli argomenti - un modello più comune per le applicazioni publisher.

Esempio 2: sottoscrittore MQ gestito

Il sottoscrittore MQ gestito è il modello preferito per la maggior parte delle applicazioni del sottoscrittore. L'esempio richiede *nessuna* definizione di gestione di code, argomenti o sottoscrizioni.

Questo tipo più semplice di sottoscrittore gestito generalmente utilizza una sottoscrizione *non durevole* . L'esempio si concentra su una sottoscrizione non durevole. La sottoscrizione dura solo per tutta la durata dell'handle di sottoscrizione da `MQSUB`. Qualsiasi pubblicazione che corrisponde alla stringa di argomenti durante la durata della sottoscrizione viene inviata alla coda di sottoscrizione (e probabilmente una pubblicazione conservata se l'indicatore `MQSO_NEW_PUBLICATIONS_ONLY` non è impostato o predefinito, una pubblicazione precedente corrispondente alla stringa di argomenti è stata conservata e la pubblicazione era persistente o il gestore code non è stato terminato, poiché la pubblicazione è stata creata).

Con questo modello è anche possibile utilizzare una sottoscrizione *durevole* . Generalmente se viene utilizzata una sottoscrizione durevole gestita, viene eseguita per motivi di affidabilità, piuttosto che per stabilire una sottoscrizione che, senza che si verifichino errori, sopravviva al sottoscrittore. Per ulteriori informazioni sui diversi cicli di vita associati alle sottoscrizioni gestite, non gestite, durevoli e non durevoli, consultare la sezione degli argomenti correlati.

Le sottoscrizioni durevoli sono spesso associate a pubblicazioni persistenti e a sottoscrizioni non durevoli con pubblicazioni non persistenti, ma non esiste una relazione necessaria tra la durata della sottoscrizione e la persistenza della pubblicazione. Tutte e quattro le combinazioni di persistenza e durata sono possibili.

Per il caso non durevole gestito, il gestore code crea una coda di sottoscrizione che viene eliminata quando la coda viene chiusa. Le pubblicazioni vengono rimosse dalla coda quando la sottoscrizione non durevole viene chiusa.

Di seguito sono riportati gli aspetti importanti del modello non durevole gestito esemplificato da questo codice:

1. Sottoscrizione su richiesta : la stringa dell'argomento sottoscrizione è dinamica. Viene fornito dall'applicazione quando viene eseguito.

2. Coda di gestione automatica: la coda di sottoscrizione è di definizione e gestione automatica.
3. Ciclo di vita della sottoscrizione a gestione automatica: *non-Le sottoscrizionidurevoli* esistono solo per la durata dell'applicazione del sottoscrittore.
   - Se si definisce una sottoscrizione gestita *durevole* , ne risulta una coda di sottoscrizione permanente e le pubblicazioni continuano ad essere memorizzate su di essa senza alcun programma sottoscrittore attivo. Il gestore code elimina la coda (e cancella tutte le pubblicazioni non richiamate da essa) solo dopo che l'applicazione o l'amministratore ha scelto di eliminare la sottoscrizione. La sottoscrizione può essere eliminata utilizzando un comando di gestione o chiudendo la sottoscrizione con l'opzione `MQCO_REMOVE_SUB` .
   - Considerare l' impostazione `SubExpiry` per le sottoscrizioni durevoli in modo che le pubblicazioni cessino di essere inviate alla coda e il sottoscrittore possa utilizzare tutte le pubblicazioni rimanenti prima di rimuovere la sottoscrizione e fare in modo che il gestore code elimini la coda e le pubblicazioni rimanenti su di essa.
4. Distribuzione flessibile della stringa di argomenti: la gestione dell'argomento di sottoscrizione viene semplificato definendo la parte root della sottoscrizione utilizzando un argomento definito amministrativamente. La parte root della struttura ad albero degli argomenti viene quindi nascosta dall'applicazione. Per nascondere la parte root, un'applicazione può essere distribuita senza che l'applicazione crei inavvertitamente una struttura ad albero degli argomenti che si sovrappone ad un'altra struttura ad albero degli argomenti creata da un'altra istanza o da un'altra applicazione.
5. Argomenti gestiti: mediante utilizzando una stringa di argomenti in cui la prima parte corrisponde a un oggetto argomento definito dall'amministratore, le pubblicazioni sono gestite in base agli attributi dell'oggetto .
   - Ad esempio, , se la prima parte della stringa argomento corrisponde alla stringa argomento associata a un oggetto argomento in cluster, la sottoscrizione può ricevere pubblicazioni da altri membri del cluster
   - La corrispondenza selettiva di oggetti argomento definiti in modo amministrativo e sottoscrizioni definite in modo programmatico consente di combinare i vantaggi di entrambi. L'amministratore fornisce attributi per gli argomenti e il programmatore definisce dinamicamente "sub-topics" senza preoccuparsi della gestione degli argomenti.
   - è la stringa dell'argomento risultante che viene utilizzata per mettere in corrispondenza l'oggetto argomento che fornisce gli attributi associati all'argomento e non necessariamente l'oggetto argomento denominato in `sd.Objectname`, anche se in genere risultano essere uno e lo stesso. Consultare “Esempio 2: pubblicazione di un argomento variabile” a pagina 284.

rendendo la sottoscrizione durevole nell'esempio, le pubblicazioni continuano ad essere inviate alla coda di sottoscrizione dopo che il sottoscrittore ha chiuso la sottoscrizione con `MQCO_KEEP_SUB` opzione. La coda continua a ricevere pubblicazioni quando il sottoscrittore non è attivo. È possibile sovrascrivere questo comportamento creando la sottoscrizione con l'opzione `MQSO_PUBLICATIONS_ON_REQUEST` e utilizzando `MQSUBRQ` per richiedere la pubblicazione conservata.

La sottoscrizione può essere ripresa in un secondo momento aprendo la sottoscrizione con l'opzione `MQCO_RESUME` .

È possibile utilizzare l'handle della coda, `Hobj`, restituito da `MQSUB` in vari modi. L'handle di coda viene utilizzato nell'esempio per analizzare il nome della coda di sottoscrizione. Le code gestite vengono aperte utilizzando le code modello predefinite `SYSTEM.NDURABLE.MODEL.QUEUE` o `SYSTEM.DURABLE.MODEL.QUEUE`. È possibile sovrascrivere i valori predefiniti fornendo le proprie code modello durevoli e non durevoli su un argomento per argomento come proprietà dell'oggetto argomento associato alla sottoscrizione.

Indipendentemente dagli attributi ereditati dalle code modello, non è possibile riutilizzare un handle della coda gestita per creare una sottoscrizione aggiuntiva. Inoltre, non è possibile ottenere un altro handle per la coda gestita aprendo la coda gestita una seconda volta utilizzando il nome della coda restituito. La coda si comporta come se fosse stata aperta per l' inputesclusivo.

Le code non gestite sono più flessibili delle code gestite. È possibile, ad esempio, condividere code non gestite o definire più sottoscrizioni su una coda. L'esempio successivo, “Esempio 3: sottoscrittore MQ non gestito” a pagina 296, illustra come combinare le sottoscrizioni con una coda di sottoscrizione non gestita.

Nota: Lo stile di codifica compatto è progettato per la leggibilità e non per l'utilizzo in produzione.

I risultati sono riportati in Figura 46 a pagina 294.

```
#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);
    }
```

Figura 44. Utente MQ gestito - parte 1: dichiarazioni e gestione parametri.

Ci sono alcuni commenti aggiuntivi da fare sulle dichiarazioni in questo esempio.

MQHOBJ Hobj = MQHO_NONE;
Non è possibile aprire esplicitamente una coda di sottoscrizioni gestite non durevoli per ricevere le pubblicazioni, ma è necessario assegnare la memoria per l'oggetto che il gestore code restituisce quando apre la coda per conto dell'utente. È importante inizializzare l'handle in MQHO_OBJECT. Questo indica al gestore code che deve restituire un handle della coda alla coda di sottoscrizione.

MQSD sd = {MQSD_DEFAULT};
Il nuovo descrittore di sottoscrizione, utilizzato in MQSUB.

MQCHAR48 qName;
Anche se l'esempio n' t richiede la conoscenza della coda di sottoscrizione, l'esempio richiede il nome della coda di sottoscrizione - il bind MQINQ è un po 'imbarazzante nel linguaggio C, quindi potresti trovare questa parte dell'esempio utile per studiare.

```
    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;
}
```

Figura 45. Utente MQ gestito - parte 2: corpo del codice.

```
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
```

Figura 46. Output dal sottoscrittore MQ gestito

Ci sono alcuni commenti aggiuntivi da fare sul codice in questo esempio.

strncpy(sd.ObjectName, topicName, MQ_Q_NAME_LENGTH);
Se topicName è null o vuoto *(valore predefinito)*, il nome dell'argomento non viene utilizzato per calcolare la stringa di argomenti risolta.

sd.ObjectString.VSPtr = topicString;
Invece di utilizzare solo un oggetto argomento predefinito, in questo esempio il programma fornisce un oggetto argomento e una stringa argomento, combinati da MQSUB. Si noti che la stringa di argomenti è una struttura MQCHARV .

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Un'alternativa all'impostazione della lunghezza di un campo MQCHARV .

sd.Options = MQSO_CREATE | MQSO_MANAGED | MQSO_NON_DURABLE | MQSO_FAIL_IF_QUIESCING;
Dopo aver definito la stringa di argomenti, gli indicatori sd.Options richiedono l'attenzione più attenta. Ci sono molte opzioni, l'esempio specifica solo quelle più comunemente utilizzate; le altre sono lasciate a default.

1. Poiché la sottoscrizione è *non durevole*, ossia, ha una durata della sottoscrizione aperta nell'applicazione, impostare l' MQSO_CREATE indicatore. È anche possibile impostare l'indicatore *(predefinito)* MQSO_NON_DURABLE per la leggibilità.
2. Il completamento di MQSO_CREATE è MQSO_RESUME. Entrambi gli indicatori possono essere impostati insieme; il gestore code crea una nuova sottoscrizione o riprende una sottoscrizione esistente, a seconda dei casi. Tuttavia, se si specifica MQSO_RESUME , è necessario inizializzare anche la struttura MQCHARV per sd.SubName, anche se non è presente alcuna sottoscrizione da riprendere. L'errore di inizializzazione di SubName determina un codice di ritorno di 2440: MQRC_SUB_NAME_ERROR da MQSUB.

   Nota: MQSO_RESUME viene sempre ignorato per una sottoscrizione gestita non durevole: ma specificarlo senza inizializzare la struttura MQCHARV per sd.SubName causa l'errore.
3. Inoltre, è presente un terzo indicatore che influenza la modalità di apertura della sottoscrizione, MQSO_ALTER. Date le autorizzazioni corrette, le proprietà di una sottoscrizione ripresa vengono modificate per corrispondere ad altri attributi specificati in MQSUB.

   Nota: È necessario specificare almeno uno degli indicatori MQSO_CREATE, MQSO_RESUME e MQSO_ALTER . Vedere Opzioni (MQLONG). Esistono esempi di utilizzo di tutti e tre gli indicatori in “Esempio 3: sottoscrittore MQ non gestito” a pagina 296.
4. Impostare MQSO_MANAGED per il gestore code per gestire automaticamente la sottoscrizione.

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Facoltativamente, omettere l'impostazione della lunghezza di MQCHARV per le stringhe con terminazione null e utilizzare invece l'indicatore di terminazione null.

sd.ResObjectString.VSPtr = resTopicStr;
La stringa di argomenti risultante viene riportata nel primo printf del programma. Impostare MQCHARV ResObjectString per WebSphere MQ per restituire la stringa risolta al programma.

Nota: resTopicStringBuffer viene inizializzato con valori null in memset(resTopicStr, 0, sizeof(resTopicStrBuffer)). Le stringhe argomento restituite non terminano con un valore null finale.

sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
Impostare la dimensione del buffer di sd.ResObjectString su un valore inferiore alla dimensione effettiva. Questo impedisce la sovrascrittura del terminatore null fornito, nel caso in cui la stringa di argomenti risolta riempia l'intero buffer.

Nota: Non viene restituito alcun errore se la stringa dell'argomento è più lunga di sizeof(resTopicStrBuffer)-1. Anche se VSLength > VSBufSiz la lunghezza restituita in sd.ResObjectString.VSLength è la lunghezza della stringa completa e non necessariamente la lunghezza della stringa restituita. Verificare sd.ResObjectString.VSLength < sd.ResObjectString.VSBufSiz per confermare che la stringa dell'argomento sia completa.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
La funzione MQSUB crea una sottoscrizione. Se non è durevole, probabilmente non si è interessati al suo nome, anche se è possibile esaminarne lo stato in WebSphere MQ Explorer. È possibile fornire il parametro sd.SubName come input, in modo da sapere quale nome cercare; è ovviamente necessario evitare conflitti di nomi con altre sottoscrizioni.

MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
La chiusura della sottoscrizione e della coda di sottoscrizione è facoltativa. Nell'esempio la sottoscrizione è chiusa, ma non la coda. L'opzione MQCLOSE MQCO_REMOVE_SUB è comunque l'opzione predefinita in questo caso poiché la sottoscrizione non è durevole. L'utilizzo di MQCO_KEEP_SUB è un errore.

Nota: la sottoscrizione *queue* non viene chiusa da MQSUBe il relativo handle, Hobj, rimane valido fino a quando la coda non viene chiusa da MQCLOSE o MQDISC. Se l'applicazione termina prematuramente, la coda e la sottoscrizione vengono ripulite dal gestore code qualche tempo dopo la chiusura dell'applicazione.

Concetti correlati

"Esempio 1: consumer di pubblicazione MQ" a pagina 288
Il consumer di pubblicazione di MQ è un consumer di messaggi IBM WebSphere MQ che non sottoscrive gli argomenti.

"Esempio 3: sottoscrittore MQ non gestito" a pagina 296
Il sottoscrittore non gestito è una classe importante dell'applicazione del sottoscrittore. Con esso, si combinano i vantaggi della pubblicazione / sottoscrizione con il *controllo* dell'accodamento e dell'utilizzo delle pubblicazioni. L'esempio illustra diversi modi di combinazione di sottoscrizioni e code.

"Scrittura delle applicazioni del publisher" a pagina 280
Inizia a scrivere applicazioni di pubblicazione studiando due esempi. Il primo è modellato il più strettamente possibile su un'applicazione point to point inserendo i messaggi in una coda e il secondo dimostra la creazione dinamica degli argomenti - un modello più comune per le applicazioni publisher.

Esempio 3: sottoscrittore MQ non gestito

Il sottoscrittore non gestito è una classe importante dell'applicazione del sottoscrittore. Con esso, si combinano i vantaggi della pubblicazione / sottoscrizione con il *controllo* dell'accodamento e dell'utilizzo delle pubblicazioni. L'esempio illustra diversi modi di combinazione di sottoscrizioni e code.

Il pattern non gestito è più comunemente associato a sottoscrizioni *durevoli* rispetto a *non durevoli*. In genere, il ciclo di vita di una sottoscrizione creata da un sottoscrittore non gestito è indipendente dal ciclo di vita dell'applicazione di sottoscrizione stessa. Rendendo la sottoscrizione durevole la sottoscrizione riceve le pubblicazioni anche quando non è attiva alcuna applicazione di sottoscrizione.

È possibile creare sottoscrizioni *gestite* durevoli per ottenere lo stesso risultato, ma alcune applicazioni richiedono maggiore flessibilità e controllo su code e messaggi rispetto a quanto è possibile con una sottoscrizione gestita. Per una sottoscrizione gestita durevole, il gestore code crea una coda permanente per le pubblicazioni che corrispondono all'argomento della sottoscrizione. Elimina la coda e le pubblicazioni associate quando la sottoscrizione viene eliminata.

Generalmente, le sottoscrizioni *gestite* durevoli vengono utilizzate se il ciclo di vita dell'applicazione e la sottoscrizione sono essenzialmente uguali, ma difficili da garantire. Rendendo la sottoscrizione durevole e facendo in modo che il publisher crei pubblicazioni persistenti, non vi sono messaggi persi nel caso in cui il gestore code o il sottoscrittore terminino prematuramente e debbano essere ripristinati.

Il gestore code apre implicitamente la coda di sottoscrizione gestita durevole per un sottoscrittore in modo che l'elaborazione condivisa della coda non sia possibile. Inoltre, non è possibile creare più di una sottoscrizione per ogni coda gestita ed è possibile che le code siano più difficili da gestire perché si ha meno controllo sui nomi delle code. Per questi motivi, considerare se il sottoscrittore *non gestito* MQ è più adatto per le applicazioni che richiedono sottoscrizioni durevoli rispetto al sottoscrittore *gestito* MQ .

Il codice in Figura 49 a pagina 302 dimostra un pattern di sottoscrizione durevole non gestito. Per l'illustrazione il codice crea anche sottoscrizioni non gestite e non durevoli. Questo esempio illustra i seguenti facet di pattern:

- Sottoscrizioni on demand: le stringhe dell'argomento di sottoscrizione sono dinamiche. Vengono forniti dall'applicazione quando viene eseguita.
- Gestione degli argomenti di sottoscrizione semplificata: la gestione degli argomenti di sottoscrizione viene semplificata definendo la parte root della stringa degli argomenti di sottoscrizione utilizzando un argomento definito amministrativamente. Questa operazione nasconde la parte root della struttura ad albero dell'argomento dall'applicazione. Nascondendo la parte root, un sottoscrittore può essere distribuito a diverse strutture ad albero degli argomenti.
- Gestione flessibile della sottoscrizione: è possibile definire una sottoscrizione in modo amministrativo o crearla su richiesta in un programma sottoscrittore. Non esiste alcuna differenza tra le sottoscrizioni create in modo amministrativo e in modo programmatico, tranne un attributo che mostra come è stata creata la sottoscrizione. Esiste un terzo tipo di sottoscrizione che viene creato automaticamente dal gestore code per la distribuzione delle sottoscrizioni. Tutte le sottoscrizioni vengono visualizzate in WebSphere MQ Explorer.
- Associazione flessibile di sottoscrizioni con code: una coda locale predefinita è associata a una sottoscrizione dalla funzione MQSUB . Esistono diversi modi per utilizzare MQSUB per associare le sottoscrizioni alle code:
  - Associare una sottoscrizione a una coda con *nessuna* sottoscrizione esistente, `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Associare una *nuova* sottoscrizione a una coda con sottoscrizioni esistenti, `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Spostare una sottoscrizione esistente in una diversa coda, `MQSO_ALTER + (Hobj from MQOPEN)`.
  - Riprendere una sottoscrizione esistente associata a una coda esistente, `MQSO_RESUME + (Hobj = MQHO_NONE)` o `MQSO_RESUME + (Hobj = from MQOPEN of queue with existing subscription)` .
  - Combinando `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` in diverse combinazioni, è possibile soddisfare i diversi stati di input della sottoscrizione e della coda senza dover codificare più versioni di MQSUB con valori differenti `sd.Options` .
  - In alternativa, codificando una scelta specifica di `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` , il gestore code restituisce un errore (Tabella 43 a pagina 299) se gli stati della sottoscrizione e della coda forniti come input in MQSUB non sono congruenti con il valore di `sd.Options`. Figura 55 a pagina 305 mostra i risultati dell'emissione di MQSUB per la sottoscrizione X con diverse impostazioni individuali dell'indicatore `sd.Options` e la trasmissione di tre diversi handle di oggetto.

  Esplorare diversi input al programma di esempio in Figura 48 a pagina 301 per acquisire familiarità con questi diversi tipi di errore. Un errore comune, `RC = 2440`, che non è incluso nei casi elencati nella tabella, è un errore del nome della sottoscrizione. è generalmente causato dal passaggio di un nome sottoscrizione null o non valido con `MQSO_RESUME` o `MQSO_ALTER` .
- Multiprocessing: è possibile condividere tra molti consumatori il lavoro di lettura delle pubblicazioni. Le pubblicazioni vanno tutte nella singola coda associata all'argomento della sottoscrizione. I consumatori possono scegliere di aprire la coda direttamente utilizzando MQOPEN o di riprendere la sottoscrizione utilizzando MQSUB.
- Concentrazione sottoscrizione: è possibile creare più sottoscrizioni sulla stessa coda. Prestare attenzione con questa funzionalità in quanto può portare a sottoscrizioni "sovrapposte" e ricevere la stessa pubblicazione più volte. L'opzione `MQSO_GROUP_SUB` elimina le pubblicazioni duplicate causate dalla sovrapposizione delle sottoscrizioni.
- Separazione tra sottoscrittore e consumatore: oltre ai tre modelli illustrati negli esempi, un altro modello consiste nel separare il consumatore dal sottoscrittore. Si tratta di una variazione del sottoscrittore di MQ non gestito, ma piuttosto che emettere MQOPEN e MQSUB nello stesso programma, un programma sottoscrive le pubblicazioni e un altro programma le utilizza. Ad esempio, il sottoscrittore potrebbe far parte di un cluster di pubblicazione / sottoscrizione e l'utente collegato a un gestore code esterno al cluster del gestore code. L'utente riceve le pubblicazioni tramite l'accodamento distribuito standard definendo la coda di sottoscrizione come una definizione di coda remota.

Comprendere il comportamento di MQSO_CREATE | MQSO_RESUME | MQSO_ALTER è importante, soprattutto se si intende semplificare il codice utilizzando combinazioni di queste opzioni. Esaminare la tabella Tabella 43 a pagina 299 che mostra i risultati della trasmissione di diversi handle di coda a MQSUB e i risultati dell'esecuzione del programma di esempio mostrato in da a Figura 55 a pagina 305.

Lo scenario utilizzato per costruire la tabella ha una sottoscrizione X e due code, A e B . Il parametro del nome sottoscrizione sd.SubName è impostato su X, il nome di una sottoscrizione collegata alla coda A. Alla coda B non è allegata alcuna sottoscrizione.

In Tabella 43 a pagina 299, MQSUB viene passata la sottoscrizione X e l'handle di coda alla coda A. I risultati delle opzioni di sottoscrizione sono i seguenti:

- MQSO_CREATE non riesce perché l'handle di coda corrisponde alla coda A che ha già una sottoscrizione a X. Confronta questo comportamento con la chiamata riuscita. Tale chiamata ha esito positivo perché la coda B non dispone di una sottoscrizione a X ad essa collegata.
- MQSO_RESUME riesce perché l'handle della coda corrisponde alla coda A che ha già una sottoscrizione a X. Al contrario, la chiamata non riesce quando la sottoscrizione X non esiste sulla coda A.
- MQSO_ALTER si comporta in modo simile a MQSO_RESUME rispetto all'apertura della sottoscrizione e della coda. Tuttavia, se gli attributi contenuti nel descrittore della sottoscrizione inoltrato a MQSUB differiscono dagli attributi della sottoscrizione, MQSO_RESUME ha esito negativo, mentre MQSO_ALTER ha esito positivo finché l'istanza di programma dispone dell'autorizzazione per modificare gli attributi. Si noti che non è mai possibile modificare la stringa di argomenti in una sottoscrizione; ma, invece di restituire un errore, MQSUB ignora i valori del nome argomento e della stringa argomento nel descrittore della sottoscrizione e utilizza i valori nella sottoscrizione esistente.

Successivamente, esaminare Tabella 43 a pagina 299 dove MQSUB viene passato alla sottoscrizione X e l'handle di coda alla coda B. I risultati delle opzioni di sottoscrizione sono i seguenti:

- MQSO_CREATE riesce e crea la sottoscrizione X sulla coda B perché si tratta di una nuova sottoscrizione sulla coda B.
- MQSO_RESUME non viene eseguito. MQSUB cerca la sottoscrizione X sulla coda B e non la trova, ma invece di restituire *RC = 2428 - la sottoscrizione X non esiste* , restituisce *RC = 2019 - La coda di sottoscrizione non corrisponde all'handle dell'oggetto coda*. Il comportamento della terza opzione MQSO_ALTER suggerisce il motivo di questo errore imprevisto. MQSUB prevede che l'handle della coda punti a una coda con una sottoscrizione. Lo verifica prima di verificare se la sottoscrizione denominata in sd.SubName esiste.
- MQSO_ALTER riesce e sposta la sottoscrizione dalla coda A alla coda B.

Un caso non visualizzato nella tabella è se il nome della sottoscrizione sulla coda A non corrisponde al nome della sottoscrizione in sd.SubName. La chiamata ha esito negativo con un *RC = 2428 - la sottoscrizione X non esiste nella coda A* .

Tabella 43. Errori da MQSUB con diversi handle di coda e combinazioni di sottoscrizioni

	Coda A **Sottoscrizione X** **Coda B** **Nessuna sottoscrizione**	**Coda A** **Nessuna sottoscrizione** **Coda B** **Nessuna sottoscrizione**
Hobj per *Coda A* passato a MQSUB	**MQSO_CREATE** RC = 2432 - La sottoscrizione X esiste già nella coda A **MQSO_RESUME** Riprende la sottoscrizione X sulla coda A **MQSO_ALTER** Riprende la sottoscrizione X sulla Coda A e apporta le modifiche consentite	**MQSO_CREATE** Crea la sottoscrizione X sulla coda A **MQSO_RESUME** RC = 2428 - La sottoscrizione X non esiste nella coda A **MQSO_ALTER** RC = 2428 - La sottoscrizione X non esiste nella coda A
Hobj per *Coda B* passato a MQSUB	**MQSO_CREATE** Crea una nuova sottoscrizione X sulla coda B **MQSO_RESUME** RC = 2019 - La coda di sottoscrizione non corrisponde al gestore oggetti della coda **MQSO_ALTER** Spostare la sottoscrizione X dalla coda A alla coda B	**MQSO_CREATE** Crea una nuova sottoscrizione X sulla coda B **MQSO_RESUME** RC = 2428 - la sottoscrizione X non esiste sulla coda B **MQSO_ALTER** RC = 2428 - la sottoscrizione X non esiste sulla coda B
MQHO_NONE passato a MQSUB	**MQSO_CREATE** RC = 2019 - Handle oggetto non valido: impostare l'indicatore MQSO_MANAGED per creare una sottoscrizione gestita e creare una coda gestita **MQSO_RESUME** Riprende la sottoscrizione X sulla coda A e restituisce Hobj alla coda A **MQSO_ALTER** Riprende la sottoscrizione X sulla coda A, restituisce Hobj alla coda A e apporta le modifiche consentite	**MQSO_CREATE** RC = 2019 - Handle oggetto non valido: impostare l'indicatore MQSO_MANAGED per creare una sottoscrizione gestita e creare una coda gestita **MQSO_RESUME** RC = 2428 - Nessuna sottoscrizione X **MQSO_ALTER** RC = 2019 - Handle oggetto non valido: nessuna coda A o B

Nota: Lo stile di codifica compatto è progettato per la leggibilità e non per l'utilizzo in produzione.

```
#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));
```

Figura 47. Sottoscrittore MQ non gestito - parte 1: dichiarazioni.

```
    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);
    }
```

Figura 48. Utente MQ non gestito - parte 2: gestione parametri.

Ulteriori commenti sulla gestione dei parametri in questo esempio sono i seguenti:

switch((argv[5][0]))
È possibile immettere Alter | C reate | Resume nel parametro 5 per verificare l'effetto della sovrascrittura di parte dell'impostazione dell'opzione MQSUB utilizzata per impostazione predefinita nell'esempio. L'impostazione predefinita utilizzata dall'esempio è MQSO_CREATE | MQSO_RESUME | MQSO_DURABLE.

Nota: L'impostazione MQSO_ALTER o MQSO_RESUME senza l'impostazione MQSO_DURABLE è un errore e sd.SubName deve essere impostato e fare riferimento a una sottoscrizione che può essere ripresa o modificata.

***subscriptionQueue = '\0';**
sdOptions = sdOptions + MQSO_MANAGED;
Se la coda di sottoscrizione predefinita, STOCKTICKER viene sostituita da una stringa nulla, se MQSO_CREATE è impostato, l'esempio imposta l'indicatore MQSO_MANAGED e crea una coda di sottoscrizione dinamica. Se Alter or Resume sono impostati nel quinto parametro, il comportamento dell'esempio dipenderà dal valore di subscriptionName.

***subscriptionName = '\0';**
sdOptions = sdOptions - MQSO_DURABLE;

Se la sottoscrizione predefinita, IBMSTOCKPRICESUB , viene sostituita da una stringa nulla, l'esempio rimuove l'indicatore MQSO_DURABLE . Se si esegue l'esempio fornendo i valori predefiniti per gli altri parametri, viene creata una sottoscrizione temporanea aggiuntiva destinata a STOCKTICKER e riceve pubblicazioni duplicate. La prossima volta che si esegue l'esempio, senza alcun parametro, si riceve di nuovo una sola pubblicazione.

```
    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;
}
```

Figura 49. Sottoscrittore MQ non gestito - parte 3: corpo del codice.

Ulteriori commenti sul codice in questo esempio sono i seguenti:

if (strlen(subscriptionQueue))
Se non è presente alcun nome coda di sottoscrizione, l'esempio utilizza MQHO_NONE come valore di Hobj.

MQOPEN(...);
La coda di sottoscrizione viene aperta e l'handle di coda salvato in Hobj.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
La sottoscrizione viene aperta utilizzando il Hobj passato da MQOPEN (o MQHO_NONE se non è presente alcun nome coda di sottoscrizione). Una coda non gestita può essere ripresa senza aprirla esplicitamente con un MQOPEN.

MQCLOSE(Hconn, &Hsub, MQCO_NONE, &CompCode, &Reason);
La sottoscrizione viene chiusa utilizzando l'handle di sottoscrizione. A seconda che la sottoscrizione sia durevole o meno, la sottoscrizione viene chiusa con un MQCO_KEEP_SUB o MQCO_REMOVE_SUBimplicito. Puoi chiudere una sottoscrizione durevole con MQCO_REMOVE_SUB, ma *non puoi* chiudere una sottoscrizione non durevole con MQCO_KEEP_SUB. L'azione di MQCO_REMOVE_SUB consiste nel rimuovere la sottoscrizione che arresta l'invio di ulteriori pubblicazioni alla coda di sottoscrizione.

MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
Non viene eseguita alcuna azione speciale se la sottoscrizione non è gestita. Se la coda è gestita e la sottoscrizione è chiusa con un MQCO_REMOVE_SUBesplicito o implicito, tutte le pubblicazioni vengono eliminate dalla coda e la coda viene eliminata a questo punto.

gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
Assicurarsi che i messaggi ricevuti siano quelli per la nostra sottoscrizione.

I risultati dell'esempio illustrano alcuni aspetti della pubblicazione / sottoscrizione:

In Figura 50 a pagina 303 l'esempio inizia pubblicando 130 nell'argomento 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
```

Figura 50. Pubblica 130 su NYSE/IBM/PRICE

In Figura 51 a pagina 303 l'esecuzione dell'esempio utilizzando i parametri predefiniti riceve la pubblicazione conservata 130. L'oggetto argomento fornito e la stringa argomento vengono ignorati, come mostrato in Figura 55 a pagina 305. L'oggetto argomento e la stringa argomento vengono sempre presi dall'oggetto sottoscrizione, quando ne viene fornito uno, e la stringa argomento è immutabile. Il funzionamento effettivo dell'esempio dipende dalla scelta o dalla combinazione di MQSO_CREATE, MQSO_RESUMEe MQSO_ALTER . In questo esempio MQSO_RESUME è l'opzione selezionata.

```
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
```

Figura 51. Ricevere la pubblicazione conservata

In (Figura 52 a pagina 304) non vengono ricevute pubblicazioni, poiché la sottoscrizione durevole ha già ricevuto la pubblicazione conservata. In questo esempio, la sottoscrizione viene ripresa fornendo solo il nome della sottoscrizione senza il nome della coda. Se il nome della coda è stato fornito, la coda viene aperta per prima e l'handle viene passato a MQSUB.

Nota: L'errore 2038 proveniente da MQINQ è dovuto al MQOPEN implicito di STOCKTICKER da MQSUB che non include l'opzione MQOO_INQUIRE . Evitare il codice di ritorno 2038 da MQINQ aprendo esplicitamente la coda.

```
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
```

Figura 52. Riprendi sottoscrizione

In Figura 53 a pagina 304 , l'esempio crea una sottoscrizione non gestita durevole utilizzando STOCKTICKER come destinazione. Poiché si tratta di una nuova sottoscrizione, riceve la pubblicazione conservata.

```
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
```

Figura 53. Ricevi pubblicazione conservata con nuova sottoscrizione non durevole non gestita

In Figura 54 a pagina 304 , per dimostrare le sottoscrizioni sovrapposte, viene inviata un'altra pubblicazione, modificando la pubblicazione conservata. Successivamente, viene creata una nuova sottoscrizione non durevole e non gestita non fornendo un nome sottoscrizione. La pubblicazione conservata viene ricevuta due volte, una per la nuova sottoscrizione e una per la sottoscrizione IBMSTOCKPRICESUB durevole ancora attiva sulla coda STOCKTICKER . L'esempio è un'illustrazione che indica che la coda ha sottoscrizioni e non l'applicazione. Nonostante non faccia riferimento alla sottoscrizione IBMSTOCKPRICESUB in questa chiamata dell'applicazione, l'applicazione riceve la pubblicazione due volte: una dalla sottoscrizione durevole creata amministrativamente e una dalla sottoscrizione non durevole creata dall'applicazione stessa.

```
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
```

Figura 54. Sovrapposizione delle sottoscrizioni

In Figura 55 a pagina 305 l'esempio dimostra che la fornitura di una nuova stringa di argomenti e di una sottoscrizione esistente non comporta una sottoscrizione modificata.

1. Nel primo caso, `Resume` riprende la sottoscrizione esistente, come potrebbe essere previsto, e ignora la stringa di argomenti modificata.
2. Nel secondo caso, `Alter` causa un errore, `RC = 2510, Topic not alterable`.
3. Nel terzo esempio, `Create` causa un errore `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
```

Figura 55. Impossibile modificare gli argomenti della sottoscrizione

Concetti correlati

“Esempio 1: consumer di pubblicazione MQ” a pagina 288
Il consumer di pubblicazione di MQ è un consumer di messaggi IBM WebSphere MQ che non sottoscrive gli argomenti.

“Esempio 2: sottoscrittore MQ gestito” a pagina 291
Il sottoscrittore MQ gestito è il modello preferito per la maggior parte delle applicazioni del sottoscrittore. L'esempio richiede *nessuna* definizione di gestione di code, argomenti o sottoscrizioni.

“Scrittura delle applicazioni del publisher” a pagina 280
Inizia a scrivere applicazioni di pubblicazione studiando due esempi. Il primo è modellato il più strettamente possibile su un'applicazione point to point inserendo i messaggi in una coda e il secondo dimostra la creazione dinamica degli argomenti - un modello più comune per le applicazioni publisher.

Cicli di vita di pubblicazione / sottoscrizione

Considerare i cicli di vita di argomenti, sottoscrizioni, sottoscrittori, pubblicazioni, publisher e code nella progettazione di applicazioni di pubblicazione / sottoscrizione.

Il ciclo di vita di un oggetto, ad esempio una sottoscrizione, inizia con la sua creazione e termina con la sua eliminazione. Può anche includere altri stati e modifiche che attraversa, come la sospensione temporanea, la presenza di argomenti principali e secondari, la scadenza e l'eliminazione.

Tradizionalmente, gli oggetti WebSphere MQ come le code vengono creati amministrativamente o dai programmi di gestione utilizzando PCF (Programmable Command Format). La pubblicazione / sottoscrizione è diversa nel fornire i verbi API `MQSUB` e `MQCLOSE` per creare ed eliminare le sottoscrizioni, avendo il concetto di sottoscrizioni gestite che non solo creano ed eliminano le code, ma anche ripuliscono i messaggi non utilizzati e hanno associazioni tra gli oggetti argomento creati amministrativamente e le stringhe argomento create in modo programmatico o amministrativo.

Questa ricchezza funzionale soddisfa un'ampia gamma di requisiti di pubblicazione / sottoscrizione e semplifica la progettazione di alcuni modelli comuni di applicazione di pubblicazione / sottoscrizione. Le sottoscrizioni gestite, ad esempio, semplificano sia la programmazione che la gestione di una sottoscrizione destinata a durare solo fino a quando il programma che l'ha creata. Le sottoscrizioni non gestite semplificano la programmazione quando vi è una connessione più flessibile tra la sottoscrizione e l'utilizzo delle pubblicazioni. Le sottoscrizioni create centralmente sono utili quando il modello è quello di instradare il traffico di pubblicazione verso i consumatori in base a un modello centralizzato di controllo, ad esempio l'invio di informazioni di volo a gate automatizzati, mentre le sottoscrizioni create programmaticamente possono essere utilizzate se il personale del gate è responsabile della sottoscrizione ai record dei passeggeri per quel volo, inserendo un numero di volo a un gate.

In questo ultimo esempio, una sottoscrizione durevole gestita potrebbe essere appropriata: gestita, perché le sottoscrizioni vengono create molto spesso e hanno un endpoint chiaro quando il gate si chiude e la sottoscrizione può essere rimossa in modo programmatico; durevole, per evitare di perdere un record di passeggeri a causa del programma del gate sottoscrittore che si disattiva per un motivo o per l'altro[2]. Per avviare la pubblicazione dei registri dei passeggeri al gate, una possibile progettazione potrebbe essere che l'applicazione del gate sottoscriva i registri dei passeggeri utilizzando il numero del

[2] L'editore deve inviare le registrazioni dei passeggeri come messaggi persistenti per evitare altri possibili guasti, naturalmente.

gate e pubblichi l'evento di apertura del gate utilizzando il numero del gate. L'editore risponde all'evento di apertura del gate pubblicando le registrazioni dei passeggeri - che potrebbero poi andare anche ad altre parti interessate, come la fatturazione, per registrare il volo in corso, e ai servizi clienti, per inviare notifiche di testo ai telefoni cellulari dei passeggeri del numero del gate.

La sottoscrizione gestita centralmente potrebbe utilizzare un modello non gestito durevole, instradando gli elenchi di passeggeri al gate utilizzando una coda predefinita per ogni gate.

I seguenti tre esempi di cicli di vita di pubblicazione / sottoscrizione illustrano il modo in cui i sottoscrittori non durevoli, non durevoli e non durevoli gestiti interagiscono con le sottoscrizioni, gli argomenti, le code, i publisher e il gestore code e il modo in cui le responsabilità possono essere suddivise tra i programmi di gestione e sottoscrittore.

Sottoscrittore non durevole gestito

Figura 56 a pagina 307 mostra un'applicazione che crea una sottoscrizione non durevole gestita, che riceve due messaggi pubblicati nell'argomento identificato nella sottoscrizione e che termina. Le interazioni etichettate in un carattere grigio corsivo con frecce tratteggiate sono implicite.

Ci sono alcuni punti da notare.

1. L'applicazione crea una sottoscrizione su un argomento che è già stato pubblicato due volte. Quando il sottoscrittore riceve la prima pubblicazione, riceve la *seconda* pubblicazione che è la pubblicazione attualmente conservata.
2. Il gestore code crea una coda di sottoscrizione temporanea e una sottoscrizione per l'argomento.
3. La sottoscrizione ha una scadenza. Quando la sottoscrizione scade, non vengono inviate ulteriori pubblicazioni sull'argomento a questa sottoscrizione, ma il sottoscrittore continua a ricevere i messaggi pubblicati prima della scadenza della sottoscrizione. La scadenza della pubblicazione non è influenzata dalla scadenza della sottoscrizione.
4. La quarta pubblicazione non viene collocata nella coda di sottoscrizione e di conseguenza l'ultima `MQGET` non restituisce una pubblicazione.
5. Sebbene il sottoscrittore chiuda la sottoscrizione, non chiude la connessione alla coda o al gestore code.
6. Il gestore code si ripulisce poco dopo la chiusura dell'applicazione. Poiché la sottoscrizione è gestita e non durevole, la coda di sottoscrizione viene eliminata.

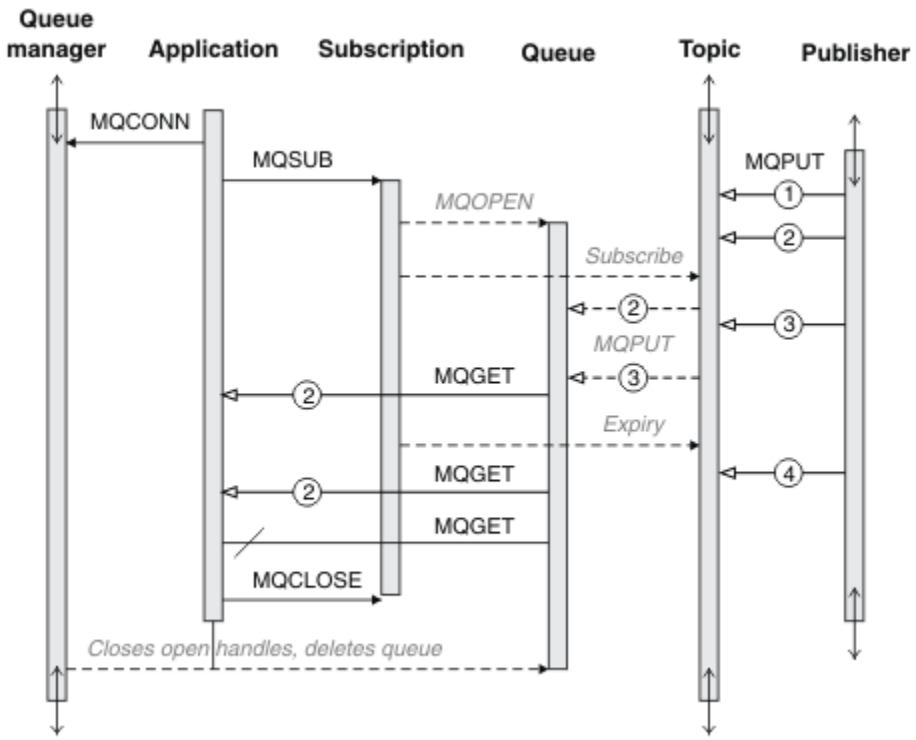


Figura 56. Cicli di vita del sottoscrittore non durevoli gestiti

Sottoscrittore durevole gestito

Il sottoscrittore (subscriber) duraturo gestito fa un passo avanti nell'esempio precedente e mostra una sottoscrizione gestita che sopravvive alla terminazione e al riavvio dell'applicazione di sottoscrizione.

Ci sono alcuni nuovi punti da notare.

1. In questo esempio, a differenza dell'ultimo, l'argomento della pubblicazione non esisteva prima che fosse definito nella sottoscrizione.
2. La prima volta che il sottoscrittore termina, chiude la sottoscrizione con l'opzione `MQCO_KEEP_SUB`. Questo è il comportamento predefinito per chiudere implicitamente una sottoscrizione durevole gestita.
3. Quando il sottoscrittore riprende la sottoscrizione, la coda di sottoscrizione viene riaperta.
4. La nuova pubblicazione 2, inserita nella coda prima della riapertura, è disponibile per `MQGET`, anche dopo la rimozione della sottoscrizione.

   Anche se la sottoscrizione è durevole, il sottoscrittore riceve in modo affidabile tutti i messaggi inviati dal publisher solo se *sia* la sottoscrizione è durevole e i messaggi persistenti. La persistenza del messaggio dipende dall'impostazione del campo `Persistent` nel `MQMD` del messaggio inviato dal publisher. Un sottoscrittore non ha alcun controllo su questo.
5. La chiusura della sottoscrizione con l'indicatore `MQCO_REMOVE_SUB` rimuove la sottoscrizione, arrestando eventuali ulteriori pubblicazioni inserite nella coda di sottoscrizione. Quando la coda di sottoscrizione viene chiusa, il gestore code rimuove la pubblicazione non letta 3ed elimina la coda. L'azione equivale all'eliminazione amministrativa della sottoscrizione.

   Nota: Non eliminare la coda manualmente o immettere `MQCLOSE` con l'opzione `MQCO_DELETE`o `MQCO_PURGE_DELETE`. I dettagli di implementazione visibili di una sottoscrizione gestita non fanno parte dell'interfaccia WebSphere MQ supportata. Il gestore code non è in grado di gestire una sottoscrizione in modo affidabile a meno che non disponga di un controllo completo.

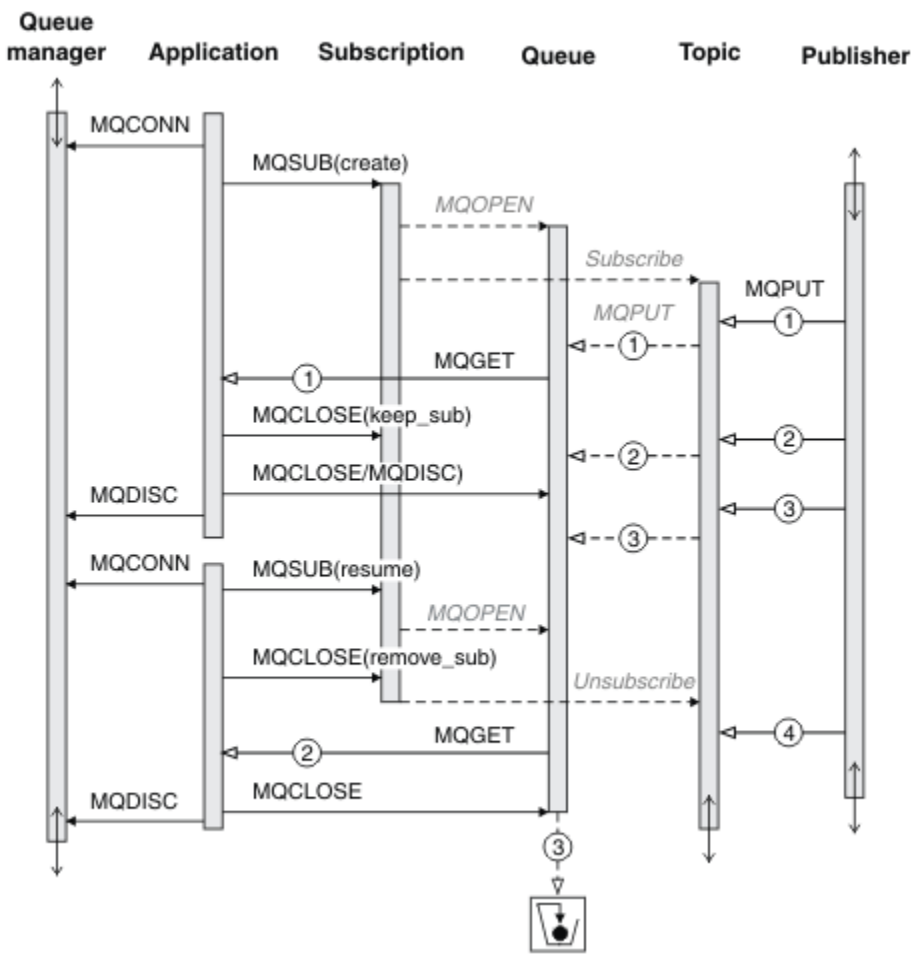


Figura 57. Cicli di vita del sottoscrittore durevoli gestiti

Sottoscrittore durevole non gestito

Un amministratore viene aggiunto nel terzo esempio: il sottoscrittore durevole non gestito. Questo è un buon esempio per mostrare come l'amministratore potrebbe interagire con un'applicazione di pubblicazione / sottoscrizione.

I punti da notare sono elencati.

1. Il publisher inserisce un messaggio, 1, in un argomento che in seguito viene associato all'oggetto argomento utilizzato per la sottoscrizione. L'oggetto argomento definisce una stringa di argomento che corrisponde all'argomento pubblicato utilizzando i caratteri jolly.
2. L'argomento ha una pubblicazione conservata.
3. L'amministratore crea un argomento, una coda e una sottoscrizione. L'oggetto argomento e la coda devono essere definiti prima della sottoscrizione.
4. L'applicazione apre la coda associata alla sottoscrizione e passa `MQSUB` l'handle della coda. In alternativa, potrebbe semplicemente aprire la sottoscrizione, passando l'handle della coda `MQHO_NONE`. L'inverso non è true, non può riprendere una sottoscrizione passando solo l'handle della coda senza un nome sottoscrizione - una coda potrebbe avere più sottoscrizioni.
5. L'applicazione apre la sottoscrizione utilizzando l'opzione `MQSO_RESUME` anche se è la prima volta che apre la sottoscrizione. Sta riprendendo una sottoscrizione creata in modo amministrativo.

6. Il sottoscrittore riceve la pubblicazione conservata, 1. La pubblicazione 2, sebbene sia stata pubblicata prima della ricezione di qualsiasi pubblicazione da parte del sottoscrittore, è stata pubblicata dopo l'inizio della sottoscrizione ed è la seconda pubblicazione sulla coda di sottoscrizione.

   Nota: Se la pubblicazione conservata non viene pubblicata come messaggio persistente, viene persa dopo il riavvio del gestore code.
7. In questo esempio la sottoscrizione è durevole. È possibile per un programma creare una sottoscrizione non durevole non gestita; dovrebbe essere ovvio che questo non è qualcosa che un amministratore può fare.
8. L'opzione `MQCO_REMOVE_SUB` alla chiusura della sottoscrizione ha l'effetto di rimuovere la sottoscrizione come se fosse stata eliminata dall'amministratore. Questa operazione arresta le ulteriori pubblicazioni inviate alla coda, ma non influisce sulle pubblicazioni già presenti nella coda, anche quando la coda è chiusa, a differenza di una sottoscrizione durevole *gestita* .
9. L'amministratore in seguito elimina il messaggio rimanente, 3, ed elimina la coda.

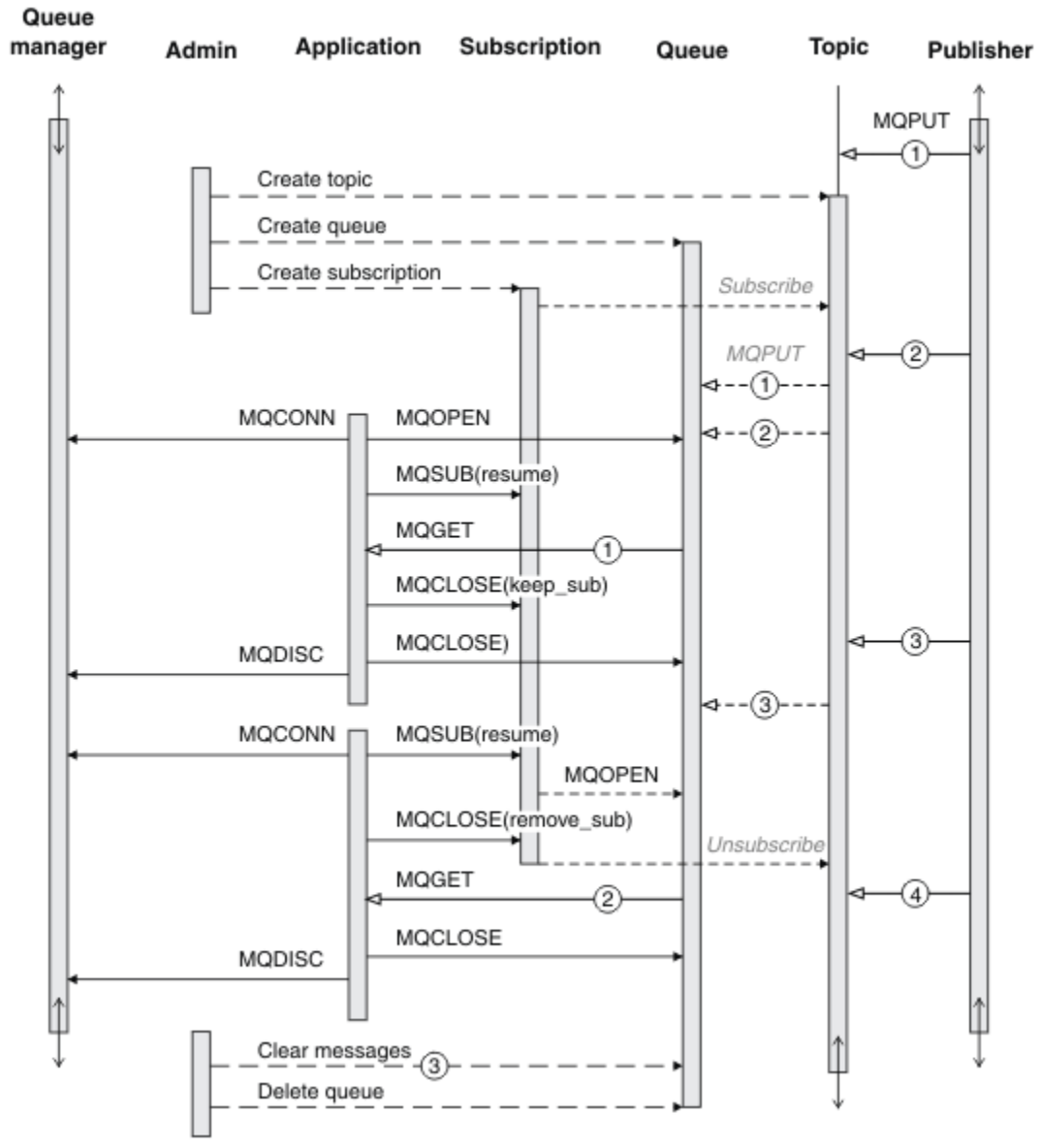


Figura 58. Cicli di vita del sottoscrittore durevoli non gestiti

Un modello normale per una sottoscrizione non gestita è per la gestione della coda e della sottoscrizione che deve essere eseguita dall'amministratore. In genere non si tenta di emulare il funzionamento di un sottoscrittore gestito e di riordinare le code e le sottoscrizioni in modo programmatico nel codice dell'applicazione. Se si ha bisogno di scrivere la logica di gestione, è necessario chiedersi se è possibile ottenere gli stessi risultati utilizzando un modello gestito. Non è facile scrivere codice di gestione strettamente sincronizzato e completamente affidabile. È più facile riordinare in un secondo momento, manualmente o utilizzando un programma di gestione automatizzato, quando è possibile essere sicuri che i messaggi, le sottoscrizioni e le code possano essere semplicemente eliminati, indipendentemente dal loro stato.

Proprietà dei messaggi di pubblicazione / sottoscrizione

Diverse proprietà dei messaggi sono correlate alla messaggistica di pubblicazione / sottoscrizione di WebSphere MQ .

Token PubAccounting

Questo è il valore che sarà nel campo AccountingToken di MQMD (Message Descriptor) di tutti i messaggi di pubblicazione corrispondenti a questa sottoscrizione. AccountingToken fa parte del contesto di identità del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare “Contesto messaggio” a pagina 39. Per ulteriori informazioni sul campo AccountingToken in MQMD, consultare AccountingToken.

PubApplIdentityData

Questo è il valore che si trova nel campo ApplIdentityData di MQMD (Message Descriptor) di tutti i messaggi di pubblicazione corrispondenti a questa sottoscrizione. ApplIdentityI dati fanno parte del contesto di identità del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare “Contesto messaggio” a pagina 39. Per ulteriori informazioni sul campo ApplIdentityData in MQMD, consultare ApplIdentityData.

Se l'opzione MQSO_SET_IDENTITY_CONTEXT non è specificata, i dati ApplIdentityche verranno impostati in ogni messaggio pubblicato per questa sottoscrizione sono vuoti, come informazioni di contesto predefinite.

Se viene specificata l'opzione MQSO_SET_IDENTITY_CONTEXT, l'IdentityData PubApplviene generato dall'utente e questo campo è un campo di input che contiene i dati ApplIdentityda impostare in ogni pubblicazione per questa sottoscrizione.

PubPriority

Questo è il valore che sarà nel campo Priorità di MQMD (Message Descriptor) di tutti i messaggi di pubblicazione che corrispondono a questa sottoscrizione. Per ulteriori informazioni sul campo Priorità in MQMD, consultare Priorità.

Il valore deve essere maggiore o uguale a zero; zero è la priorità più bassa. È inoltre possibile utilizzare i seguenti valori speciali:

- MQPRI_PRIORITY_AS_Q_DEF - Quando una coda di sottoscrizione viene fornita nel campo Hobj nella chiamata MQSUB e non è un handle gestito, la priorità per il messaggio viene presa dall'attributo DefProirity di questa coda. Se la coda così identificata è una coda cluster o esiste più di una definizione nel percorso di risoluzione del nome della coda, la priorità viene determinata quando il messaggio di pubblicazione viene inserito nella coda come descritto per Priorità in MQMD. Se la chiamata MQSUB utilizza un handle gestito, la priorità per il messaggio viene presa dall'attributo DefPriority della coda modello associata all'argomento sottoscritto.
- MQPRI_PRIORITY_AS_PUBLISHED - La priorità del messaggio è la priorità della pubblicazione originale. Questo è il valore iniziale di questo campo.

SubCorrelId

Attenzione: un identificatore di correlazione può essere passato solo tra i gestori code in un cluster di pubblicazione / sottoscrizione, non una gerarchia.

Tutte le pubblicazioni inviate per corrispondere a questa sottoscrizione conterranno questo identificativo di correlazione nel descrittore del messaggio. Se più sottoscrizioni utilizzano la stessa coda da cui ottenere le relative pubblicazioni, l'uso di MQGET per ID correlazione consente di ottenere solo le pubblicazioni per una sottoscrizione specifica. Questo identificativo di correlazione può essere generato dal gestore code o dall'utente.

Se l'opzione MQSO_SET_CORREL_ID non viene specificata, l'identificatore di correlazione viene generato dal gestore code e questo campo è un campo di output che contiene l'identificatore di correlazione che verrà impostato in ogni messaggio pubblicato per questa sottoscrizione.

Se viene specificata l'opzione MQSO_SET_CORREL_ID, l'identificativo di correlazione viene generato dall'utente e questo campo è un campo di immissione che contiene l'identificativo di correlazione da impostare in ogni pubblicazione per questa sottoscrizione. In questo caso, se il campo contiene MQCI_NONE, l'identificativo di correlazione che verrà impostato in ogni messaggio pubblicato per questa sottoscrizione sarà l'identificativo di correlazione creato dall'inserimento originale del messaggio.

Se viene specificata l'opzione MQSO_GROUP_SUB e l'identificatore di correlazione specificato è lo stesso di una sottoscrizione raggruppata esistente che utilizza la stessa coda e una stringa di argomento sovrapposta, solo la sottoscrizione più significativa nel gruppo viene fornita con una copia della pubblicazione.

SubUserData

Questi sono i dati utente della sottoscrizione. I dati forniti nella sottoscrizione in questo campo verranno inclusi come proprietà del messaggio di dati MQSubUserdi ogni pubblicazione inviata a questa sottoscrizione.

Proprietà della pubblicazione

Tabella 44 a pagina 311 elenca le proprietà di pubblicazione fornite con un messaggio di pubblicazione.

È possibile accedere a queste proprietà direttamente dalla cartella **MQRFH2** o richiamarle utilizzando MQINQMP. MQINQMP accetta il nome della proprietà o il nome **MQRFH2** come nome della proprietà da analizzare.

Tabella 44. Proprietà della pubblicazione

Nome della proprietà	Nome MQRFH2	Tipo	Descrizione
MQTopicString	mqps.Top	MQTYPE_STRING	Stringa argomento
MQSubUserData	mqps.Sud	MQTYPE_STRING	Dati utente sottoscrittore
MQIsRetained	mqps.Ret	MQTYPE_BOOLEAN	Pubblicazione conservata
MQPubOptions	mqps.Pub	MQTYPE_INT32	Opzioni di pubblicazione
MQPubLevel	mqps.Pbl	MQTYPE_INT32	Livello di pubblicazione
MQPubTime	mqpse.Pts	MQTYPE_STRING	Ora di pubblicazione
MQPubSeqNum	mqpse.Seq	MQTYPE_INT32	Numero di sequenza pubblicazione
MQPubStrIntData	mqpse.Sid	MQTYPE_STRING	Dati stringa / intero aggiunti dal publisher

Tabella 44. Proprietà della pubblicazione (Continua)			
Nome della proprietà	**Nome MQRFH2**	**Tipo**	**Descrizione**
MQPubFormat	mqpse.Pfmt	MQTYPE_INT32	Formato messaggio: MQRFH1 MQRFH2 PCF

Ordinamento dei messaggi

Per un particolare argomento, i messaggi vengono pubblicati dal gestore code nello stesso ordine in cui vengono ricevuti dalle applicazioni di pubblicazione (soggetto a riordinamento in base alla priorità del messaggio).

L'ordine dei messaggi normalmente indica che ogni sottoscrittore riceve messaggi da un determinato gestore code, su un determinato argomento, da uno specifico publisher nell'ordine in cui vengono pubblicati da tale publisher.

Tuttavia, come per tutti i messaggi WebSphere MQ , è possibile che i messaggi, occasionalmente, vengano consegnati in modo non ordinato. Ciò può verificarsi nelle situazioni seguenti:

- Se un collegamento nella rete si disattiva e i messaggi successivi vengono reinstradati lungo un altro collegamento
- Se una coda diventa temporaneamente piena o inibita, in modo che un messaggio venga inserito in una coda di messaggi non recapitabili e quindi ritardato, mentre i messaggi successivi passano direttamente attraverso di essa.
- Se l'amministratore elimina un gestore code quando i publisher e i sottoscrittori sono ancora operativi, i messaggi accodati vengono inseriti nella coda di messaggi non recapitabili e le sottoscrizioni vengono interrotte.

Se queste circostanze non possono verificarsi, le pubblicazioni vengono sempre consegnate in ordine.

Nota: Non è possibile utilizzare messaggi raggruppati o segmentati con Pubblicazione / Sottoscrizione.

Intercettazione delle pubblicazioni

È possibile intercettare una pubblicazione, modificarla e ripubblicarla prima che raggiunga qualsiasi altro sottoscrittore.

È possibile intercettare una pubblicazione prima che raggiunga un sottoscrittore per effettuare una delle seguenti operazioni:

- Allegare ulteriori informazioni al messaggio
- Blocca il messaggio
- Trasforma il messaggio

È possibile eseguire la stessa operazione su ciascun messaggio o modificare l'operazione in base alla sottoscrizione, al messaggio o all'intestazione del messaggio.

Riferimenti correlati

MQ_PUBLISH_EXIT - Uscita pubblicazione

Livelli di sottoscrizione

Impostare il livello di sottoscrizione di una sottoscrizione per intercettare una pubblicazione prima che raggiunga i relativi sottoscrittori finali. Un sottoscrittore intercettore effettua la sottoscrizione ad un livello di sottoscrizione superiore e ripubblica ad un livello di pubblicazione inferiore. Creare una catena di sottoscrittori di intercettazione per eseguire l'elaborazione dei messaggi su una pubblicazione prima che venga consegnata ai sottoscrittori finali.

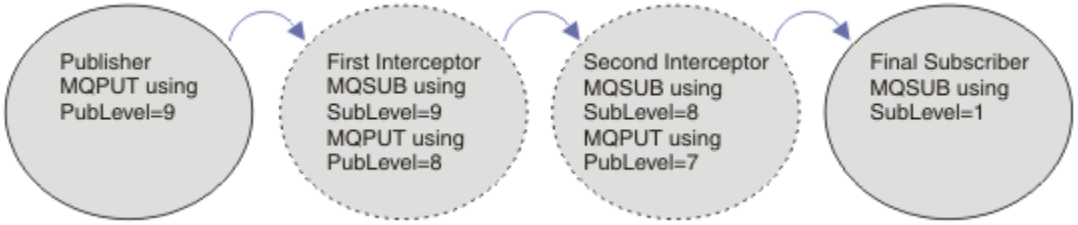


Figura 59. Sequenza di intercettazione dei sottoscrittori

Per intercettare una pubblicazione, utilizzare l'attributo **MQSD** `SubLevel` . Dopo che un messaggio è stato intercettato, è possibile trasformarlo e ripubblicarlo ad un livello di pubblicazione inferiore modificando l'attributo **MQPMO** `PubLevel` . Il messaggio viene quindi inviato ai sottoscrittori finali oppure viene intercettato nuovamente da un sottoscrittore intermedio a un livello di sottoscrizione inferiore.

Il sottoscrittore intercettore generalmente trasforma un messaggio prima di ripubblicarlo. Una sequenza di sottoscrittori di intercettazione forma un flusso di messaggi. In alternativa, è possibile non ripubblicare la pubblicazione intercettata: i sottoscrittori a livelli di sottoscrizione inferiori non riceveranno il messaggio.

Assicurarsi che l'intercettatore riceva le pubblicazioni prima di qualsiasi altro sottoscrittore. Impostare il livello di sottoscrizione dell'intercettatore su un valore superiore rispetto agli altri sottoscrittori. Per impostazione predefinita, i sottoscrittori hanno un `SubLevel` di 1. Il valore più alto è 9. Una pubblicazione deve cominciare con un `PubLevel` alto almeno quanto il `SubLevel`più alto. Pubblicare inizialmente con il `PubLevel` predefinito di 9.

- Se si dispone di un sottoscrittore intercettatore su un argomento, impostare `SubLevel` su 9.
- Per più applicazioni di intercettazione su un topic, impostare un `SubLevel` inferiore per ogni successivo sottoscrittore di intercettazione.
- È possibile implementare un massimo di 8 applicazioni di intercettazione, con livelli di sottoscrizione da 9 a 2 inclusi. Il destinatario finale del messaggio ha un `SubLevel` di 1.

L'intercettatore con il livello di sottoscrizione più alto uguale o inferiore al `PubLevel` della pubblicazione riceve prima la pubblicazione. Configurare solo un sottoscrittore intercettore per un argomento a un determinato livello di sottoscrizione. Se si dispone di più sottoscrittori a un determinato livello di sottoscrizione, vengono inviate più copie della pubblicazione alla serie finale di applicazioni di sottoscrizione.

Un sottoscrittore con un `SubLevel` `0` viene utilizzato come catchall. Riceve la pubblicazione se nessun sottoscrittore finale riceve il messaggio. Un sottoscrittore con `SubLevel` di `0` potrebbe essere utilizzato per monitorare le pubblicazioni che nessun altro sottoscrittore ha ricevuto.

Programmazione di un sottoscrittore intercettore

Utilizzare le opzioni di sottoscrizione descritte in Tabella 45 a pagina 313.

Tabella 45. Opzioni di sottoscrizione per intercettare i sottoscrittori

Opzione di sottoscrizione	**Note**
`MQSO_SET_CORREL_ID` e `SubCorrelId` impostato su `MQCI_NONE`	Mantenere il `CorrelId` della pubblicazione intercettata uguale alla pubblicazione originale. **Nota:** Non è possibile passare l'identificatore di correlazione di una pubblicazione in una gerarchia. Il campo viene utilizzato dal gestore code.
`PubPriority` impostato su `MQPRI_PRIORITY_AS_PUBLISHED`	Mantenere la priorità della pubblicazione intercettata uguale alla pubblicazione originale.

Le opzioni in Tabella 45 a pagina 313 devono essere utilizzate da tutti i sottoscrittori intercettatori. Il risultato è che l'identificatore di correlazione e la priorità del messaggio non vengono modificati dall'impostazione del publisher originale.

Quando il sottoscrittore intercettore ha elaborato la pubblicazione, ripubblica il messaggio nello stesso argomento in un `PubLevel` inferiore al `SubLevel` della propria sottoscrizione. Se il sottoscrittore di intercettazione imposta un `SubLevel` di 9, ripubblica il messaggio con un `PubLevel` di 8.

Per ripubblicare correttamente il messaggio, sono richieste diverse informazioni dalla pubblicazione originale. Riutilizzare lo stesso **MQMD** del messaggio originale e impostare `MQPMO_PASS_ALL_CONTEXT` per garantire che tutte le informazioni in **MQMD** vengano trasmesse al sottoscrittore successivo. Copiare i valori dalle proprietà del messaggio mostrate in Tabella 46 a pagina 314 nei corrispondenti campi del messaggio ripubblicato. Il sottoscrittore intercettore può modificare questi valori. Utilizzare l'operatore OR per aggiungere ulteriori valori al campo **MQPMO**.`Opzioni` , per combinare le opzioni di inserimento del messaggio.

È necessario aprire esplicitamente la coda di pubblicazione piuttosto che utilizzare una coda di pubblicazione gestita. Non è possibile impostare `MQSO_SET_CORREL_ID` per una coda gestita. Non è inoltre possibile impostare `MQOO_SAVE_ALL_CONTEXT` su una coda gestita. Consultare i frammenti di codice elencati in “Esempi” a pagina 314.

Tabella 46. Valori MQPUT per i messaggi ripubblicati

Ripubblica messaggio utilizzando MQPUT	**Informazioni nel messaggio di pubblicazione**
MQOD.`ObjectString`	proprietà dei messaggi `MQTopicString`
MQPMO.`Options`	proprietà dei messaggi `MQPubOptions`

Il sottoscrittore finale ha la possibilità di impostare le proprie opzioni di sottoscrizione in modo diverso. Ad esempio, potrebbe impostare esplicitamente la priorità di pubblicazione piuttosto che `MQPRI_PRIORITY_AS_PUBLISHED`. Le impostazioni di un sottoscrittore finale influiscono solo sulla pubblicazione del sottoscrittore intercettatore finale nella catena.

Pubblicazioni conservate

Una pubblicazione conservata deve essere conservata dopo essere stata intercettata, copiando le opzioni put - message originali nel messaggio ripubblicato.

L'opzione `MQPMO_RETAIN` è impostata dall'autore (publisher). Ogni sottoscrittore intercettatore deve trasferire il `MQPubOptions` alle opzioni put - message del messaggio ripubblicato come mostrato in Tabella 46 a pagina 314. La copia delle opzioni put - message conserva le opzioni impostate dal publisher originale, incluso se conservare la pubblicazione.

Quando una pubblicazione termina il suo passaggio nella catena di intercettazione dei sottoscrittori e viene consegnata ai sottoscrittori finali, viene infine conservata. I nuovi sottoscrittori, al livello secondario `SubLevel` 1, che richiedono la pubblicazione conservata, la ricevono senza ulteriori intercettazioni. I sottoscrittori a un `SubLevel` maggiore di 1 non ricevono la pubblicazione conservata. Di conseguenza, la pubblicazione conservata non viene modificata dalla catena di intercettazione dei sottoscrittori una seconda volta.

Esempi

Gli esempi sono frammenti di codice che possono essere combinati per creare un sottoscrittore intercettore. Il codice è scritto per essere breve, piuttosto che di qualità di produzione.

Le direttive del preprocessore in Figura 60 a pagina 315 definiscono le due proprietà da estrarre dai messaggi di pubblicazione richiesti dalla chiamata MQI `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
```

Figura 60. Direttive del preprocessore

Figura 61 a pagina 315 elenca le dichiarazioni utilizzate nei frammenti di codice. Ad eccezione dei termini evidenziati, le dichiarazioni sono standard per un'applicazione WebSphere MQ .

Le opzioni Put e Get evidenziate vengono inizializzate per passare tutto il contesto. MQTOPICSTRING e MQPUBOPTIONS evidenziati sono inizializzatori MQCHARV per i nomi delle proprietà definiti nelle direttive del preprocessore. I nomi vengono passati a 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;
```

Figura 61. Dichiarazioni

Le inizializzazioni che non vengono eseguite facilmente nelle dichiarazioni vengono mostrate in Figura 62 a pagina 316. I valori evidenziati richiedono una spiegazione.

SYSTEM.NDURABLE.MODEL.QUEUE
: In questo esempio, invece di utilizzare MQSUB per aprire una sottoscrizione non duratura gestita, la coda modello, SYSTEM.NDURABLE.MODEL.QUEUE, è utilizzata per creare una coda dinamica temporanea. Il relativo handle viene passato a MQSUB. Aprendo direttamente la coda è possibile salvare tutto il contesto del messaggio e impostare l'opzione di sottoscrizione, MQSO_SET_CORREL_ID.

MQGMO_CURRENT_VERSION
È importante utilizzare la versione corrente della maggior parte delle strutture WebSphere MQ . I campi come `gmo.MsgHandle` sono disponibili solo nella versione più recente delle strutture di controllo.

MQGMO_PROPERTIES_IN_HANDLE
La stringa di argomenti e le opzioni di inserimento dei messaggi impostati nella pubblicazione originale devono essere richiamati dal sottoscrittore intercettatore utilizzando le proprietà del messaggio. Un'alternativa potrebbe essere quella di leggere direttamente la struttura **MQRFH2** nel messaggio.

MQSO_SET_CORREL_ID
Utilizzare `MQSO_SET_CORREL_ID` in combinazione con,

```
memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
```

L'effetto di queste opzioni è quello di trasmettere l'identificativo di correlazione. L'identificativo di correlazione impostato dal publisher originale viene inserito nel campo dell'identificativo di correlazione della pubblicazione ricevuto dal sottoscrittore intercettatore. Ogni sottoscrittore intercettore trasmette lo stesso identificativo di correlazione. Il sottoscrittore finale ha quindi la possibilità di ricevere lo stesso identificativo di correlazione.

Nota: Se la pubblicazione viene passata attraverso una gerarchia di pubblicazione / sottoscrizione, l'identificativo di correlazione non viene mai conservato.

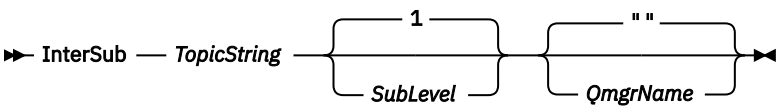
MQPRI_PRIORITY_AS_PUBLISHED
La pubblicazione viene inserita nella coda di pubblicazione con la stessa priorità del messaggio con cui è stata pubblicata.

```
  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;
```

Figura 62. Inizializzazioni

Figura 63 a pagina 317 mostra il frammento di codice per leggere i parametri della riga comandi, completare l'inizializzazione e creare la sottoscrizione intercettante.

Eseguire il programma con il comando,



Per rendere la gestione degli errori il più discreta possibile, il codice motivo di ogni chiamata MQI viene memorizzato in un elemento array differente. Dopo ogni chiamata, il codice di completamento viene verificato e, se il valore è MQCC_FAIL, il controllo esce dal blocco di codice do `{ } while(0)`.

Le due linee di codice degne di nota sono:

pmo.PubLevel = sd.SubLevel - 1;
Imposta il livello di pubblicazione per il messaggio ripubblicato su un valore inferiore al livello di sottoscrizione del sottoscrittore intercettore.

gmo.MsgHandle = Hmsg;
Fornisce un handle del messaggio per MQGET per restituire le proprietà del messaggio.

```
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;
```

Figura 63. Preparazione all'intercettazione delle pubblicazioni

Il frammento di codice principale, Figura 64 a pagina 318, richiama i messaggi dalla coda di pubblicazione. Esegue una query delle proprietà del messaggio e ripubblica i messaggi utilizzando la stringa di argomenti e le proprietà **MQPMO**.option originali della pubblicazione.

In questo esempio, non viene eseguita alcuna trasformazione sulla pubblicazione. La stringa di argomenti della pubblicazione ripubblicata corrisponde sempre alla stringa di argomenti sottoscritta dal sottoscrittore intercettatore. Se il sottoscrittore intercettatore è responsabile dell'intercettazione di più sottoscrizioni inviate alla stessa coda di pubblicazione, potrebbe essere necessario interrogare la stringa di argomenti per distinguere le pubblicazioni che corrispondono a sottoscrizioni differenti.

Vengono evidenziate le chiamate a MQINQMP . Le proprietà delle opzioni del messaggio di inserimento della pubblicazione e della stringa dell'argomento vengono scritte direttamente nelle strutture di controllo di output. L'unico motivo per modificare il campo di lunghezza MQCHARV di putOD.ObjectString da una lunghezza esplicita a una stringa con terminazione null è utilizzare printf per emettere la stringa.

```
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;
}
```

Figura 64. Intercetta pubblicazione e ripubblica

Il frammento di codice finale viene mostrato in Figura 65 a pagina 318.

```
    } 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);
}
```

Figura 65. Completamento

Intercettazione di pubblicazioni e pubblicazione / sottoscrizione distribuita

Seguire un pattern semplice quando si distribuisce l'intercettazione di sottoscrittori o le uscite di pubblicazione in una topologia di pubblicazione / sottoscrizione distribuita. Distribuire i sottoscrittori di intercettazione sugli stessi gestori code dei publisher e le uscite di pubblicazione sugli stessi gestori code dei sottoscrittori finali.

La Figura 66 a pagina 319 mostra due gestori code connessi in un cluster di pubblicazione - sottoscrizione. Un publisher crea una pubblicazione per un argomento cluster a livello di pubblicazione 9. Le frecce numerate mostrano la sequenza di passi intrapresi dalla pubblicazione durante il flusso ai sottoscrittori dell'argomento cluster. La pubblicazione viene intercettata dal sottoscrittore con `Sublevel` 9 e ripubblicata con `Publevel` 8. Viene intercettata nuovamente da un sottoscrittore in `Livello secondario` 8. Il sottoscrittore ripubblica a `Publevel` 7. Il sottoscrittore proxy fornito dal gestore code inoltra la pubblicazione al gestore code B, dove è stata distribuita un'uscita di pubblicazione oltre a un sottoscrittore finale. La pubblicazione viene elaborata dall'uscita di pubblicazione prima che venga finalmente ricevuta dal sottoscrittore finale a `Sottolivello` 1. I sottoscrittori intercettatori e l'uscita di pubblicazione vengono visualizzati con contorni interrotti.

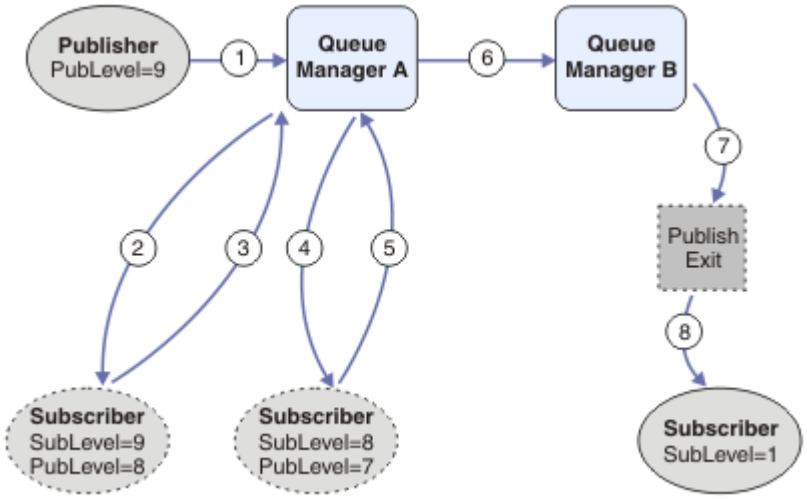


Figura 66. Uscita di pubblicazione e intercettazione in un cluster

L'obiettivo del modello semplice è che ogni sottoscrittore che riceve una pubblicazione riceva la pubblicazione identica. La pubblicazione passa attraverso la stessa sequenza di trasformazioni indipendentemente da dove è connesso il sottoscrittore. Probabilmente si desidera evitare che la sequenza di trasformazioni vari, a seconda di dove sono connessi i publisher o i sottoscrittori finali. Un'eccezione ragionevole sarebbe quella di adattare la pubblicazione finalmente consegnata a ogni singolo sottoscrittore. Utilizzare l'uscita di Pubblicazione per personalizzare la pubblicazione in base alla coda a cui la pubblicazione viene consegnata.

È necessario considerare attentamente dove distribuire le intercettazioni dei sottoscrittori e le uscite di pubblicazione in una topologia di pubblicazione / sottoscrizione distribuita. Il modello semplice distribuisce le intercettazioni dei sottoscrittori allo stesso gestore code dei publisher e le uscite di pubblicazione agli stessi gestori code dei sottoscrittori finali.

Anti - modello

Figura 67 a pagina 320 mostra come le cose possono andare storto, se non si segue un modello semplice. Per complicare la distribuzione, viene aggiunto un sottoscrittore finale al gestore code A e due ulteriori sottoscrittori di intercettazione vengono aggiunti al gestore code B.

La pubblicazione viene inoltrata al gestore code B in `PubLevel` 7, dove viene intercettata da un sottoscrittore in `SubLevel` 5 prima di essere utilizzata dal sottoscrittore finale in `SubLevel` 1. L'uscita di pubblicazione intercetta la pubblicazione prima che venga passata sia al consumer intercettore che a quello finale sul gestore code B. La pubblicazione raggiunge il sottoscrittore finale sul gestore code A senza essere elaborata dall'uscita di pubblicazione.

In una topologia di pubblicazione / sottoscrizione, i sottoscrittori proxy si sottoscrivono a `SubLevel` 1e passano il `PubLevel` impostato dall'ultimo sottoscrittore intercettore. In Figura 67 a pagina 320, il risultato è che la pubblicazione non viene intercettata dal sottoscrittore utilizzando `SubLevel` 9 sul gestore code B.

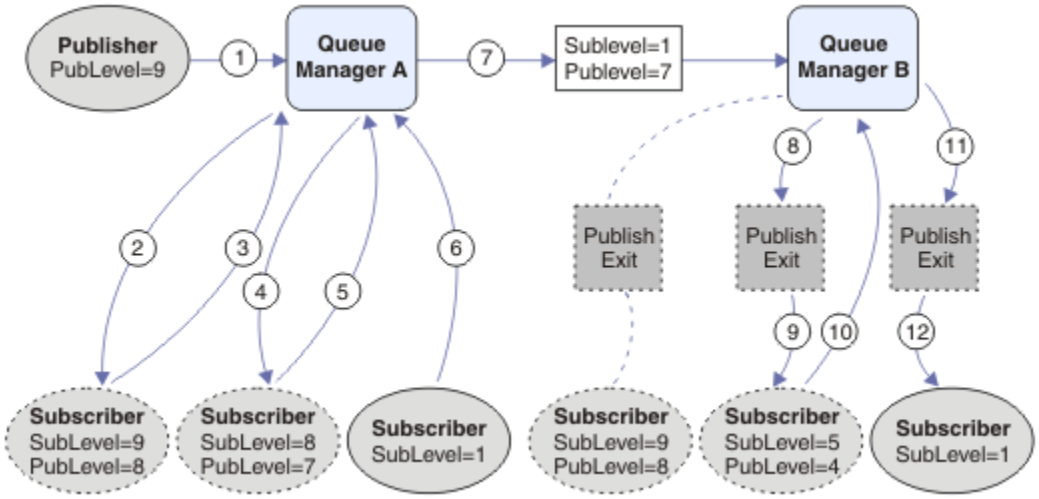


Figura 67. Distribuzione complessa dei sottoscrittori di intercettazioni

Opzioni di pubblicazione

Sono disponibili diverse opzioni che controllano la modalità di pubblicazione dei messaggi.

Rifiuto delle informazioni di risposta ai sottoscrittori

Se non si desidera che i sottoscrittori siano in grado di rispondere alle pubblicazioni che ricevono, è possibile nascondere le informazioni nei campi Qmgr ReplyToe ReplyTodi MQMD utilizzando l'opzione put - message MQPMO_SUPPRESS_REPLYTO. Se viene utilizzata questa opzione, il gestore code rimuove tali informazioni da MQMD quando riceve la pubblicazione prima di inoltrarla a qualsiasi sottoscrittore (subscriber).

Questa opzione non può essere utilizzata in combinazione con un'opzione di report che richiede una coda ReplyTo, se questa viene tentata con esito negativo con MQRC_MISSING_REPLY_TO_Q.

Livello di pubblicazione

L'utilizzo dei livelli di pubblicazione consente di controllare quali sottoscrittori ricevono la pubblicazione. Il livello di pubblicazione indica il livello di sottoscrizione di destinazione della pubblicazione. Solo le sottoscrizioni con il livello di sottoscrizione più alto inferiore o uguale al livello di pubblicazione della pubblicazione riceveranno la pubblicazione. Questo valore deve essere compreso tra zero e nove; zero è il livello di pubblicazione più basso. Il valore iniziale di questo campo è 9. Uno degli utilizzi dei livelli di pubblicazione e sottoscrizione è per intercettare le pubblicazioni.

Verifica se una pubblicazione non viene consegnata ad alcun sottoscrittore

Per controllare se una pubblicazione non è stata consegnata ad alcun sottoscrittore (subscriber), utilizzare l'opzione put - message MQPMO_WARN_IF_NO_SUBS_MATCHED con la chiamata MQPUT. Se l'operazione di inserimento restituisce un codice di completamento MQCC_WARNING e un codice motivo MQRC_NO_SUBS_MATCHED, la pubblicazione non è stata consegnata ad alcuna sottoscrizione. Se l'opzione MQPMO_RETAIN viene specificata sull'operazione di inserimento, il messaggio viene conservato e consegnato a qualsiasi sottoscrizione corrispondente definita successivamente. In un sistema di pubblicazione / sottoscrizione distribuito, il codice motivo MQRC_NO_SUBS_MATCHED viene restituito solo se non vi sono sottoscrizioni proxy registrate per l'argomento sul gestore code.

Opzioni di sottoscrizione

Sono disponibili diverse opzioni che controllano il modo in cui vengono gestite le sottoscrizioni dei messaggi.

Persistenza messaggio

I gestori code mantengono la persistenza delle pubblicazioni che inoltrano ai sottoscrittori come impostato dal publisher. Il publisher imposta la persistenza in modo che sia una delle seguenti opzioni:

0
Non permanente

1
permanente

2
Persistenza come definizione di coda / argomento

Per la pubblicazione / sottoscrizione, il publisher risolve l'oggetto argomento e **topicString** in un oggetto argomento risolto. Se il publisher specifica Persistenza come definizione coda / argomento, la persistenza predefinita dall'oggetto argomento risolto viene impostata per la pubblicazione.

Pubblicazioni conservate

Per controllare quando vengono ricevute le pubblicazioni conservate, i sottoscrittori possono utilizzare due opzioni di sottoscrizione:

Pubblica solo su richiesta, MQSO_PUBLICATIONS_ON_REQUEST

Se si desidera che un sottoscrittore abbia il controllo di quando riceve le pubblicazioni, è possibile utilizzare l'opzione di sottoscrizione MQSO_PUBLICATIONS_ON_REQUEST. Un sottoscrittore può quindi controllare quando riceve le pubblicazioni utilizzando la chiamata MQSUBRQ (specificando l'handle Hsub restituito dalla chiamata MQSUB originale) per richiedere che venga inviata la pubblicazione conservata di un argomento. I sottoscrittori che utilizzano l'opzione di sottoscrizione MQSO_PUBLICATIONS_ON_REQUEST non ricevono alcuna pubblicazione non conservata.

Se si specifica MQSO_PUBLICATIONS_ON_REQUEST è necessario utilizzare MQSUBRQ per richiamare qualsiasi pubblicazione. Se non si utilizza MQSO_PUBLICATIONS_ON_REQUEST, vengono visualizzati i messaggi man mano che vengono pubblicati.

Se un sottoscrittore (subscriber) utilizza la chiamata MQSUBRQ e utilizza i caratteri jolly nell'argomento della sottoscrizione, la sottoscrizione potrebbe corrispondere a più argomenti o nodi su una struttura ad albero degli argomenti, tutti i quali con messaggi conservati (se presenti) verranno inviati al sottoscrittore.

Questa opzione può essere particolarmente utile quando viene utilizzata con sottoscrizioni durevoli perché un gestore code continuerà a inviare pubblicazioni a un sottoscrittore se ha sottoscritto in modo duraturo anche se l'applicazione del sottoscrittore non è in esecuzione. Ciò potrebbe causare un accumulo di messaggi sulla coda del sottoscrittore. Questa creazione può essere evitata se il sottoscrittore (subscriber) si registra utilizzando l'opzione MQSO_PUBLICATIONS_ON_REQUEST. In alternativa, è possibile utilizzare sottoscrizioni non durevoli, se appropriato per la propria applicazione, per evitare un accumulo di messaggi indesiderati.

Se una sottoscrizione è durevole e un publisher utilizza pubblicazioni conservate, l'applicazione del sottoscrittore può utilizzare la chiamata MQSUBRQ per aggiornare le informazioni sullo stato dopo un riavvio. Il sottoscrittore deve quindi aggiornare periodicamente il proprio stato utilizzando la chiamata MQSUBRQ.

Nessuna pubblicazione verrà inviata come risultato della chiamata MQSUB che utilizza questa opzione. Una sottoscrizione durevole che è stata ripresa dopo la disconnessione utilizzerà l'opzione MQSO_PUBLICATIONS_ON_REQUEST se la sottoscrizione originale è stata configurata per utilizzare questa opzione.

Solo nuove pubblicazioni, MQSO_NEW_PUBLICATIONS_ONLY

Se una pubblicazione conservata esiste su un argomento, tutti i sottoscrittori che effettuano una sottoscrizione dopo che la pubblicazione è stata effettuata riceveranno una copia di tale pubblicazione. Se un sottoscrittore (subscriber) non desidera ricevere le

pubblicazioni effettuate prima della sottoscrizione, può utilizzare l'opzione di sottoscrizione MQSO_NEW_PUBLICATIONS_ONLY.

Raggruppamento di sottoscrizioni

Considerare il raggruppamento delle sottoscrizioni se è stata impostata una coda per ricevere le pubblicazioni e si dispone di un numero di sottoscrizioni sovrapposte che alimentano le pubblicazioni nella stessa coda. Questa situazione è simile all'esempio in Sottoscrizioni sovrapposte.

È possibile evitare di ricevere pubblicazioni duplicate impostando l'opzione MQSO_GROUP_SUB quando si sottoscrive un argomento. Il risultato è che quando più di una sottoscrizione nel gruppo corrisponde all'argomento di una pubblicazione, solo una sottoscrizione è responsabile dell'inserimento della pubblicazione nella coda. Le altre sottoscrizioni corrispondenti all'argomento della pubblicazione vengono ignorate.

La sottoscrizione responsabile dell'inserimento della pubblicazione nella coda viene scelta in base al fatto che ha la stringa di argomento corrispondente più lunga, prima di incontrare qualsiasi carattere jolly. Può essere pensato come l'abbonamento corrispondente più vicino. Le sue proprietà vengono propagate alla pubblicazione, incluso se dispone della proprietà MQSO_NOT_OWN_PUBS . In caso contrario, non viene consegnata alcuna pubblicazione alla coda, anche se altre sottoscrizioni corrispondenti potrebbero non avere la proprietà MQSO_NOT_OWN_PUBS .

Non è possibile inserire tutte le sottoscrizioni in un unico gruppo per eliminare le pubblicazioni duplicate. Le sottoscrizioni raggruppate devono soddisfare queste condizioni:

1. Nessuna delle sottoscrizioni è gestita.
2. Un gruppo di sottoscrizioni consegna pubblicazioni alla stessa coda.
3. Ogni sottoscrizione deve essere allo stesso livello di sottoscrizione.
4. Il messaggio di pubblicazione per ogni sottoscrizione nel gruppo ha lo stesso identificativo di correlazione.

   Per assicurarsi che ogni sottoscrizione risulti in un messaggio di pubblicazione con lo stesso identificativo di correlazione, impostare MQSO_SET_CORREL_ID per creare il proprio identificativo di correlazione nella pubblicazione e impostare lo stesso valore nel campo **SubCorrelId** in ciascuna sottoscrizione. Non impostare **SubCorrelId** sul valore MQCI_NONE.

Consultare ../com.ibm.mq.ref.dev.doc/q100080_.dita#q100080_/mqso_group_sub per ulteriori informazioni.

Richiesta di informazioni e impostazione degli attributi dell'oggetto

Gli attributi sono proprietà che definiscono le caratteristiche di un oggetto WebSphere MQ .

Influiscono sul modo in cui un gestore code elabora un oggetto. Gli attributi di ciascun tipo di oggetto WebSphere MQ sono descritti in dettaglio in Attributi di oggetti.

Alcuni attributi vengono impostati quando l'oggetto viene definito e possono essere modificati solo utilizzando i comandi WebSphere MQ ; un esempio di tale attributo è la priorità predefinita per i messaggi inseriti in una coda. Altri attributi sono influenzati dal funzionamento del gestore code e possono cambiare nel tempo; un esempio è la profondità corrente di una coda.

È possibile richiedere informazioni sui valori correnti della maggior parte degli attributi utilizzando la chiamata MQINQ. La MQI fornisce anche una chiamata MQSET con cui è possibile modificare alcuni attributi della coda. Non è possibile utilizzare le chiamate MQI per modificare gli attributi di qualsiasi altro tipo di oggetto; è invece necessario utilizzare:

Per WebSphere MQ per piattaforme Windows, UNIX e Linux
La funzione MQSC, descritta in Riferimento MQSC.

Nota: I nomi degli attributi degli oggetti vengono mostrati in questa documentazione nel formato in cui vengono utilizzati con le chiamate MQINQ e MQSET. Quando si utilizzano i comandi WebSphere MQ per

definire, modificare o visualizzare gli attributi, è necessario identificare gli attributi utilizzando le parole chiave riportate nelle descrizioni dei comandi nei link degli argomenti.

Sia le chiamate MQINQ che MQSET utilizzano array di selettori per identificaregli attributi che si desidera interrogare o impostare. Esiste un selettore per ogni attributo che è possibile utilizzare. Il nome del selettore ha un prefisso, determinato dalla natura dell'attributo:

MQCA_	Questi selettori fanno riferimento ad attributi che contengono dati di caratteri (ad esempio, il nome di una coda).
MQIA_	Questi selettori fanno riferimento ad attributi che contengono valori numerici (ad esempio, *CurrentQueueDepth*, il numero di messaggi su una coda) o un valore costante (ad esempio, *SyncPoint*, se il gestore code supporta i punti di sincronizzazione).

Prima di utilizzare le chiamate MQINQ o MQSET, l'applicazione deve essere connessa al gestore code ed è necessario utilizzare la chiamata MQOPEN per aprire l'oggetto per l'impostazione o la richiesta di attributi. Queste operazioni sono descritte in "Connessione e disconnessione da un gestore code" a pagina 207 e "Apertura e chiusura di oggetti" a pagina 215.

Utilizzare i seguenti collegamenti per ulteriori informazioni sull'acquisizione delle informazioni e sull'impostazione degli attributi degli oggetti:

- "Richiesta di informazioni sugli attributi di un oggetto" a pagina 323
- "Alcuni casi in cui la chiamata MQINQ ha esito negativo" a pagina 324
- "Impostazione degli attributi della coda" a pagina 325

Concetti correlati

"Panoramica su Message Queue Interface" a pagina 195
Informazioni sui componenti MQI (Message Queue Interface).

"Connessione e disconnessione da un gestore code" a pagina 207
Per utilizzare i servizi di programmazione WebSphere MQ , un programma deve disporre di una connessione a un gestore code. Utilizzare queste informazioni per informazioni su come connettersi e disconnettersi da un gestore code.

"Apertura e chiusura di oggetti" a pagina 215
Queste informazioni forniscono informazioni sull'apertura e la chiusura degli oggetti WebSphere MQ .

"Inserimento di messaggi in una coda" a pagina 225
Utilizzare queste informazioni per informazioni su come inserire i messaggi su una coda.

"Richiamo dei messaggi da una coda" a pagina 241
Utilizzare queste informazioni per ottenere messaggi da una coda.

"Commit e backout delle unità di lavoro" a pagina 325
Queste informazioni descrivono come eseguire il commit e il backout di tutte le operazioni get e put recuperabili che si sono verificate in un'unità di lavoro.

"Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger" a pagina 331
Informazioni sui trigger e su come avviare le applicazioni IBM WebSphere MQ utilizzando i trigger.

"Utilizzo di MQI e cluster" a pagina 349
Esistono opzioni speciali sulle chiamate e sui codici di ritorno relativi al cluster.

Richiesta di informazioni sugli attributi di un oggetto

Utilizzare la chiamata MQINQ per richiedere informazioni sugli attributi di qualsiasi tipo di IBM WebSphere MQ.

Come input per questa chiamata, è necessario fornire:

- Un handle di connessione.
- Un handle di oggetto.

- Il numero di selettori.
- Un array di selettori di attributi, ogni selettore con formato MQCA_ * o MQIA_ *. Ogni selettore rappresenta un attributo con un valore che si desidera analizzare e ciascun selettore deve essere valido per il tipo di oggetto rappresentato dall'handle dell'oggetto. È possibile specificare i selettori in qualsiasi ordine.
- Il numero di attributi interi che si stanno interrogando. Specificare zero se non si sta analizzando gli attributi integer.
- La lunghezza del buffer degli attributi carattere in *CharAttrLength*. Deve essere almeno la somma delle lunghezze richieste per contenere ciascuna stringa attributo carattere. Specificare zero se non si stanno analizzando gli attributi dei caratteri.

L'output di MQINQ è:

- Una serie di valori di attributo interi copiati nell'array. Il numero di valori è determinato da *IntAttrCount*. Se *IntAttrCount* o *SelectorCount* è zero, questo parametro non viene utilizzato.
- Il buffer in cui vengono restituiti gli attributi carattere. La lunghezza del buffer viene fornita dal parametro *CharAttrLength* . Se *CharAttrLength* o *SelectorCount* è zero, questo parametro non viene utilizzato.
- Un codice di completamento. Se il codice di completamento fornisce un'avvertenza, ciò significa che la chiamata è stata completata solo parzialmente. In questo caso, esaminare il codice di errore.
- Un codice di errore. Esistono tre situazioni di completamento parziale:
  - Il selettore non si applica al tipo di coda
  - Non è consentito spazio sufficiente per gli attributi integer
  - Non è consentito spazio sufficiente per gli attributi carattere

  Se si verifica più di una di queste situazioni, viene restituita la prima che si applica.

Se si apre una coda per l'emissione o l'interrogazione e questa si risolve in una coda cluster non locale, è possibile interrogare solo il nome della coda, il tipo di coda e gli attributi comuni. I valori degli attributi comuni sono quelli della coda scelta se è stato utilizzato MQOO_BIND_ON_OPEN. I valori sono quelli di una delle possibili code cluster arbitrarie se è stato utilizzato MQOO_BIND_NOT_FIXED o MQOO_BIND_ON_GROUP o MQOO_BIND_AS_Q_DEF e l'attributo della coda *DefBind* era MQBND_BIND_NOT_FIXED. Per ulteriori informazioni, consultare “MQOPEN e cluster” a pagina 350 e MQOPEN .

Nota: I valori restituiti dalla chiamata sono un'istantanea degli attributi selezionati. Gli attributi possono essere modificati prima che il programma agisca sui valori restituiti.

Esiste una descrizione della chiamata MQINQ in MQINQ.

Alcuni casi in cui la chiamata MQINQ ha esito negativo

Se si apre un alias per informarne gli attributi, vengono restituiti gli attributi della coda alias (l'oggetto WebSphere MQ utilizzato per accedere a un'altra coda), non quelli della coda di base.

Tuttavia, la definizione della coda di base in cui si risolve l'alias viene aperta anche dal gestore code e se un altro programma modifica l'utilizzo della coda di base nell'intervallo tra le chiamate MQOPEN e MQINQ, la chiamata MQINQ ha esito negativo e restituisce il codice motivo MQRC_OBJECT_CHANGED. La chiamata ha esito negativo anche se vengono modificati gli attributi dell'oggetto coda alias.

Allo stesso modo, quando si apre una coda remota per interrogarne gli attributi, vengono restituiti solo gli attributi della definizione locale della coda remota.

Se si specificano uno o più selettori non validi per il tipo di attributi della coda su cui si sta analizzando, la chiamata MQINQ viene completata con un'avvertenza e imposta l'output come segue:

- Per gli attributi integer, gli elementi corrispondenti di *IntAttrs* sono impostati su MQIAV_NOT_APPLICABLE.
- Per gli attributi carattere, le parti corrispondenti della stringa *CharAttrs* sono impostate su asterischi.

Se si specificano uno o più selettori che non sono validi per il tipo di attributi oggetto su cui si sta analizzando, la chiamata MQINQ non riesce e restituisce il codice motivo MQRC_SELECTOR_ERROR.

Non è possibile richiamare MQINQ per esaminare una coda modello; utilizzare la funzionalità MQSC o i comandi disponibili sulla propria piattaforma.

Impostazione degli attributi della coda

Utilizzare queste informazioni per informazioni su come impostare gli attributi della coda utilizzando la chiamata MQSET.

È possibile impostare solo i seguenti attributi della coda utilizzando la chiamata MQSET:

- *InhibitGet* (ma non per le code remote)
- *DistList* (non su z/OS)
- *InhibitPut*
- *TriggerControl*
- *TriggerType*
- *TriggerDepth*
- *TriggerMsgPriority*
- *TriggerData*

La chiamata MQSET ha gli stessi parametri della chiamata MQINQ. Tuttavia, per MQSET, tutti i parametri tranne il codice di completamento e il codice motivo sono parametri di immissione. Non esistono situazioni di completamento parziale.

Nota: Non è possibile utilizzare MQI per impostare gli attributi di oggetti WebSphere MQ diversi dalle code definite localmente.

Per ulteriori dettagli sulla chiamata MQSET, consultare MQSET.

Commit e backout delle unità di lavoro

Queste informazioni descrivono come eseguire il commit e il backout di tutte le operazioni get e put recuperabili che si sono verificate in un'unità di lavoro.

I seguenti termini vengono utilizzati in questo argomento:

- Sincronizza
- Backout
- Coordinamento del punto di sincronizzazione
- Punto di sincronizzazione
- Unità di lavoro
- commit a singola fase
- commit a due fasi

Se si ha familiarità con questi termini di elaborazione della transazione, è possibile passare a “Considerazioni sul punto di sincronizzazione nelle applicazioni IBM WebSphere MQ” a pagina 327.

Commit e backout

Quando un programma inserisce un messaggio in una coda all'interno di un'unità di lavoro, quel messaggio viene reso visibile ad altri programmi solo quando il programma esegue il commit dell'unità di lavoro. Per eseguire il commit di un'unità di lavoro, tutti gli aggiornamenti devono essere eseguiti correttamente per preservare l'integrità dei dati. Se il programma rileva un errore e decide che l'operazione di inserimento non è permanente, può eseguire il backout dell'unità di lavoro. Quando un programma esegue un backout, IBM WebSphere MQ ripristina la coda rimuovendo i messaggi inseriti nella coda da tale unità di lavoro. Il modo in cui il programma esegue le operazioni di commit e di backout dipende dall'ambiente in cui il programma è in esecuzione.

Allo stesso modo, quando un programma richiama un messaggio da una coda all'interno di un'unità di lavoro, quel messaggio rimane nella coda fino a quando il programma non esegue il commit dell'unità di lavoro, ma il messaggio non è disponibile per essere richiamato da altri programmi. Il messaggio viene cancellato definitivamente dalla coda quando il programma esegue il commit dell'unità di lavoro. Se il programma esegue il backout dell'unità di lavoro, IBM WebSphere MQ ripristina la coda rendendo i messaggi disponibili per essere richiamati da altri programmi.

Coordinamento del punto di sincronizzazione, punto di sincronizzazione, unità di lavoro
Il *coordinamento del punto di sincronizzazione* è il processo mediante il quale le unità di lavoro vengono sottoposte a commit o a backout con l'integrità dei dati.

La decisione di eseguire il commit o il backout delle modifiche viene presa, nel caso più semplice, al termine di una transazione. Tuttavia, può essere più utile per un'applicazione sincronizzare le modifiche dei dati in altri punti logici all'interno di una transazione. Questi punti logici sono denominati *punti di sincronizzazione* (o *punti di sincronizzazione*) e il periodo di elaborazione di una serie di aggiornamenti tra due punti di sincronizzazione è denominato *unità di lavoro*. Diverse chiamate MQGET e MQPUT possono essere parte di una singola unità di lavoro. Il numero massimo di messaggi all'interno di un'unità di lavoro può essere controllato dall'attributo MAXUMSGS del comando ALTER QMGR su altre piattaforme, ad eccezione di z/OS. Per dettagli su questi comandi, consultare il manuale Guida di riferimento a MQSC *WebSphere MQ Script (MQSC) Command Reference* .

commit a singola fase
Un processo *commit a fase singola* è un processo in cui un programma può eseguire il commit degli aggiornamenti su una coda senza coordinare le relative modifiche con altri gestori risorse.

commit a due fasi
Un processo di *commit a due fasi* è un processo in cui gli aggiornamenti effettuati da un programma alle code IBM WebSphere MQ possono essere coordinati con gli aggiornamenti ad altre risorse (ad esempio, database sotto il controllo di DB2). In tale processo, viene eseguito il commit o il backout degli aggiornamenti a tutte le risorse.

Per facilitare la gestione delle unità di lavoro, IBM WebSphere MQ fornisce l'attributo *BackoutCount* . Viene incrementato ogni volta che viene eseguito il backout di un messaggio all'interno di un'unità di lavoro. Se il messaggio causa ripetutamente la fine anomala dell'unità di lavoro, il valore di *BackoutCount* alla fine supera quello di *BackoutThreshold*. Questo valore viene impostato quando la coda è definita. In questa situazione, l'applicazione può rimuovere il messaggio dall'unità di lavoro e inserirlo in un'altra coda, come definito in *BackoutRequeueQName*. Quando il messaggio viene spostato, l'unità di lavoro può eseguire il commit.

Utilizzare i seguenti collegamenti per ulteriori informazioni sul commit e il backout delle unità di lavoro:

- “Considerazioni sul punto di sincronizzazione nelle applicazioni IBM WebSphere MQ” a pagina 327
- “Punti di sincronizzazione in IBM WebSphere MQ nei sistemi UNIX, Linux, and Windows” a pagina 328

Concetti correlati
“Panoramica su Message Queue Interface” a pagina 195
Informazioni sui componenti MQI (Message Queue Interface).

“Connessione e disconnessione da un gestore code” a pagina 207
Per utilizzare i servizi di programmazione WebSphere MQ , un programma deve disporre di una connessione a un gestore code. Utilizzare queste informazioni per informazioni su come connettersi e disconnettersi da un gestore code.

“Apertura e chiusura di oggetti” a pagina 215
Queste informazioni forniscono informazioni sull'apertura e la chiusura degli oggetti WebSphere MQ .

“Inserimento di messaggi in una coda” a pagina 225
Utilizzare queste informazioni per informazioni su come inserire i messaggi su una coda.

“Richiamo dei messaggi da una coda” a pagina 241
Utilizzare queste informazioni per ottenere messaggi da una coda.

“Richiesta di informazioni e impostazione degli attributi dell'oggetto” a pagina 322
Gli attributi sono proprietà che definiscono le caratteristiche di un oggetto WebSphere MQ .

“Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger” a pagina 331
Informazioni sui trigger e su come avviare le applicazioni IBM WebSphere MQ utilizzando i trigger.

“Utilizzo di MQI e cluster” a pagina 349
Esistono opzioni speciali sulle chiamate e sui codici di ritorno relativi al cluster.

Considerazioni sul punto di sincronizzazione nelle applicazioni IBM WebSphere MQ

Utilizzare queste informazioni per informazioni sull'utilizzo dei punti di sincronizzazione nelle applicazioni IBM WebSphere MQ .

Il commit a due fasi è supportato in:

- WebSphere MQ per AIX
- WebSphere MQ per HP-UX
- WebSphere MQ per Linux
- WebSphere MQ per Solaris
- WebSphere MQ per Windows
- CICS per MVS/ESA 4.1
- CICS Transaction Server per z/OS
- TXSeries
- IMS/ESA
- Altri coordinatori esterni che utilizzano l'interfaccia X/Open XA

Il commit a fase singola è supportato in:

- WebSphere MQ su sistemi UNIX
- WebSphere MQ per Windows

Nota: Per ulteriori dettagli sulle interfacce esterne, consultare “Interfacce per gestori syncpoint esterni” a pagina 330e la documentazione XA *CAE Specification Distributed Transaction Processing: The XA Specification*, pubblicata da The Open Group. I gestori transazioni (come CICS, IMS, Encinae Tuxedo) possono partecipare al commit a due fasi, coordinato con altre risorse recuperabili. Ciò significa che le funzioni di accodamento fornite da WebSphere MQ possono essere portate all'interno di un'unità di lavoro, gestita dal gestore transazioni.

Gli esempi forniti con WebSphere MQ mostrano WebSphere MQ che coordina i database conformi a XA. Per ulteriori informazioni su questi esempi, consultare “Programmi di esempio per piattaforme distribuite” a pagina 97.

Nell'applicazione WebSphere MQ , è possibile specificare su ogni chiamata put e get se si desidera che la chiamata sia sotto il controllo del punto di sincronizzazione. Per far funzionare un'operazione di inserimento sotto il controllo del punto di sincronizzazione, utilizzare il valore MQPMO_SYNCPOINT nel campo *Options* della struttura MQPMO quando si richiama MQPUT. Per un'operazione get, utilizzare il valore MQGMO_SYNCPOINT nel campo *Options* della struttura MQGMO. Se non si sceglie esplicitamente un'opzione, l'azione predefinita dipende dalla piattaforma. Il valore predefinito di controllo del punto di sincronizzazione è no.

Quando una chiamata MQPUT1 viene emessa con MQPMO_SYNCPOINT, il funzionamento predefinito cambia, in modo che l'operazione di inserimento venga completata in modo asincrono. Ciò potrebbe causare una modifica nel comportamento di alcune applicazioni che si basano su determinati campi nelle strutture MQOD e MQMD che vengono restituiti, ma che ora contengono valori non definiti. Un'applicazione può specificare MQPMO_SYNC_RESPONSE per garantire che l'operazione di inserimento venga eseguita in maniera sincrona e che tutti i valori dei campi appropriati siano completati.

Quando l'applicazione riceve un codice motivo MQRC_BACKED_OUT in risposta a un MQPUT o MQGET nel punto di sincronizzazione, l'applicazione normalmente deve eseguire il backout della transazione corrente utilizzando MQBACK e quindi, se appropriato, riprovare l'intera transazione. Se l'applicazione

riceve MQRC_BACKED_OUT in risposta a una chiamata MQCMIT o MQDISC, non è necessario richiamare MQBACK.

Ogni volta che viene eseguito il backout di una chiamata MQGET, il campo *BackoutCount* della struttura MQMD del messaggio interessato viene incrementato. Un valore elevato *BackoutCount* indica un messaggio di cui è stato ripetutamente eseguito il backout. Ciò potrebbe indicare un problema con questo messaggio, che è necessario esaminare. Consultare BackoutCount per i dettagli di *BackoutCount*.

Se un programma emette la chiamata MQDISC mentre sono presenti richieste non sottoposte a commit, si verifica un punto di sincronizzazione implicito. Se il programma termina in modo anomalo, si verifica un backout implicito.

Le modifiche agli attributi della coda (dalla chiamata MQSET o dai comandi) non sono influenzate dal commit o dal backout delle unità di lavoro.

Punti di sincronizzazione in IBM WebSphere MQ nei sistemi UNIX, Linux, and Windows

Il supporto del punto di sincronizzazione opera su due tipi di unità di lavoro: locale e globale.

Un'unità di lavoro *locale* è un'unità in cui le uniche risorse aggiornate sono quelle del gestore code WebSphere MQ . Qui il coordinamento del punto di sincronizzazione viene fornito dal gestore code stesso utilizzando una procedura di commit a fase singola.

Un'unità di lavoro *globale* è un'unità in cui vengono aggiornate anche le risorse appartenenti ad altri gestori risorse, come i database. WebSphere MQ può coordinare tali unità di lavoro. Possono anche essere coordinati da un controller di commit esterno, come un altro gestore transazioni o il controller di commit IBM i .

Per la piena integrità, utilizzare una procedura di commit a due fasi. Il commit a due fasi può essere fornito da gestori transazioni e database conformi a XA come IBM TXSeries e UDB. I prodotti WebSphere MQ (tranne WebSphere MQ per IBM i e WebSphere MQ per z/OS) possono coordinare unità di lavoro globali utilizzando un processo di commit a due fasi.

Unità di lavoro locali

Le unità di lavoro che coinvolgono solo il gestore code sono denominate unità di lavoro *locali* . Il coordinamento del punto di sincronizzazione viene fornito dal gestore code stesso (coordinamento interno) utilizzando un processo di commit a fase singola.

Per avviare un'unità di lavoro locale, l'applicazione emette richieste MQGET, MQPUT o MQPUT1 che specificano l'opzione del punto di sincronizzazione appropriata. L'unità di lavoro viene sottoposta a commit utilizzando MQCMIT o a rollback utilizzando MQBACK. Tuttavia, l'unità di lavoro termina anche quando la connessione tra l'applicazione e il gestore code viene interrotta, intenzionalmente o involontariamente.

Se un'applicazione si disconnette (MQDISC) da un gestore code mentre un'unità di lavoro globale coordinata da WebSphere MQ è ancora attiva, viene effettuato un tentativo di eseguire il commit dell'unità di lavoro. Se, tuttavia, l'applicazione termina senza disconnettersi, viene eseguito il rollback dell'unità di lavoro poiché si ritiene che l'applicazione sia terminata in modo anomalo.

Unità di lavoro globali

Utilizzare le unità di lavoro globali quando è necessario anche includere aggiornamenti alle risorse appartenenti ad altri gestori risorse.

Qui il coordinamento può essere interno o esterno al gestore code:

Coordinamento del punto di sincronizzazione interno

La coordinazione dei gestori code delle unità di lavoro globali non è supportata da WebSphere MQ per IBM i o WebSphere MQ per z/OS. Non è supportato in WebSphere MQ ambiente client MQI.

In questo caso, WebSphere MQ esegue il coordinamento. Per avviare un'unità di lavoro globale, l'applicazione emette la chiamata MQBEGIN.

Come input per la chiamata MQBEGIN, è necessario fornire l'handle di connessione (*Hconn*) restituito dalla chiamata MQCONN o MQCONNX. Questo handle rappresenta la connessione al gestore code WebSphere MQ .

L'applicazione emette richieste MQGET, MQPUT o MQPUT1 che specificano l'opzione del punto di sincronizzazione appropriata. Ciò significa che è possibile utilizzare MQBEGIN per avviare un'unità di lavoro globale che aggiorna le risorse locali, le risorse appartenenti ad altri gestori risorse o entrambi. Gli aggiornamenti effettuati alle risorse appartenenti ad altri gestori risorse vengono effettuati utilizzando l'API di tale gestore risorse. Tuttavia, non è possibile utilizzare la MQI per aggiornare le code che appartengono ad altri gestori code. Immettere MQCMIT o MQBACK prima di avviare ulteriori unità di lavoro (locali o globali).

L'unità di lavoro globale viene sottoposta a commit utilizzando MQCMIT; questo avvia un commit a due fasi di tutti i gestori risorse coinvolti nell'unità di lavoro. Viene utilizzato un processo di commit a due fasi in cui i gestori risorse (ad esempio, i gestori database conformi a XA come DB2, Oraclee Sybase) vengono prima richiesti per preparare il commit. Solo se tutti sono preparati viene chiesto loro di eseguire il commit. Se un gestore risorse segnala che non è possibile eseguire il commit, a ciascuno viene richiesto di eseguire il backout. In alternativa, è possibile utilizzare MQBACK per eseguire il rollback degli aggiornamenti di tutti i gestori risorse.

Se un'applicazione si disconnette (MQDISC) mentre un'unità di lavoro globale è ancora attiva, viene eseguito il commit dell'unità di lavoro. Se, tuttavia, l'applicazione termina senza disconnettersi, viene eseguito il rollback dell'unità di lavoro poiché si ritiene che l'applicazione sia terminata in modo anomalo.

L'output di MQBEGIN è un codice di completamento e un codice motivo.

Quando si utilizza MQBEGIN per avviare un'unità di lavoro globale, vengono inclusi tutti i gestori risorse esterni configurati con il gestore code. Tuttavia, la chiamata avvia un'unità di lavoro ma viene completata con un'avvertenza se:

- Non sono presenti gestori risorse partecipanti (ossia, non è stato configurato alcun gestore risorse con il gestore code)

oppure

- Uno o più gestori risorse non sono disponibili.

In questi casi, l'unità di lavoro deve includere aggiornamenti solo ai gestori risorse che erano disponibili quando è stata avviata l'unità di lavoro.

Se uno dei gestori risorse non è in grado di eseguire il commit dei relativi aggiornamenti, a tutti i gestori risorse viene richiesto di eseguire il rollback dei relativi aggiornamenti e MQCMIT viene completato con un'avvertenza. In circostanze inusuali (di solito, l'intervento dell'operatore), una chiamata MQCMIT potrebbe non riuscire se alcuni gestori risorse eseguono il commit dei propri aggiornamenti ma altri ne eseguono il rollback; si ritiene che il lavoro sia stato completato con un risultato *misto* . Tali ricorrenze vengono diagnosticate nel log degli errori del gestore code in modo che sia possibile intraprendere un'azione correttiva.

Un MQCMIT di un'unità di lavoro globale ha esito positivo se tutti i gestori risorse coinvolti eseguono il commit dei relativi aggiornamenti.

Per una descrizione della chiamata MQBEGIN, consultare MQBEGIN.

Coordinamento del punto di sincronizzazione esterno

Ciò si verifica quando un coordinatore del punto di sincronizzazione diverso da WebSphere MQ è stato selezionato; ad esempio, CICS, Encinao Tuxedo.

In questa situazione, WebSphere MQ su sistemi UNIX and Linux e WebSphere MQ per Windows registrano il loro interesse nel risultato dell'unità di lavoro con il coordinatore del punto di sincronizzazione in modo che possano eseguire il commit o il rollback delle operazioni get o put non sottoposte a commit come

richiesto. Il coordinatore del punto di sincronizzazione esterno determina se vengono forniti protocolli di commit a una o due fasi.

Quando si utilizza un coordinatore esterno, MQCMIT, MQBACK e MQBEGIN non possono essere emessi. Le chiamate a queste funzioni non riescono con il codice motivo MQRC_ENVIRONMENT_ERROR.

Il modo in cui viene avviata un'unità di lavoro coordinata esternamente dipende dall'interfaccia di programmazione fornita dal coordinatore del punto di sincronizzazione. Potrebbe essere richiesta una chiamata esplicita. Se è richiesta una chiamata esplicita e si emette una chiamata MQPUT specificando l'opzione MQPMO_SYNCPOINT quando un'unità di lavoro non è avviata, viene restituito il codice di completamento MQRC_SYNCPOINT_NOT_AVAILABLE.

L'ambito dell'unità di lavoro è determinato dal coordinatore del punto di sincronizzazione. Lo stato della connessione tra l'applicazione e il gestore code influisce sull'esito positivo o negativo delle chiamate MQI che un'applicazione emette e non sullo stato dell'unità di lavoro. Un'applicazione può, ad esempio, disconnettersi e riconnettersi a un gestore code durante un'unità di lavoro attiva ed eseguire ulteriori operazioni MQGET e MQPUT all'interno della stessa unità di lavoro. Ciò è noto come disconnessione in sospeso.

È possibile utilizzare le chiamate API WebSphere MQ nei programmi CICS , se si sceglie di utilizzare le capacità XA di CICS. Se non si utilizza XA, le operazioni di inserimento e ricezione dei messaggi da e verso le code non verranno gestite all'interno delle unità di lavoro atomiche CICS . Un motivo per scegliere questo metodo è che la coerenza generale dell'unità di lavoro non è importante per voi.

Se l'integrità delle unità di lavoro è importante per l'utente, è necessario utilizzare XA. Quando si utilizza XA CICS utilizza un protocollo di commit a due fasi per garantire che tutte le risorse all'interno dell'unità di lavoro vengano aggiornate insieme.

Per ulteriori informazioni sull'impostazione del supporto transazionale, consultare “Scenari di supporto transazionale” a pagina 42e anche la documentazione TXSeries CICS , ad esempio, *TXSeries for Multiplatforms CICS Administration Guide for Open Systems*.

Interfacce per gestori syncpoint esterni

WebSphere MQ su sistemi UNIX and Linux , e WebSphere MQ per Windows supportano il coordinamento delle transazioni da parte di gestori syncpoint esterni che utilizzano l'interfaccia XA X/Open.

Alcuni gestori transazioni XA (TXSeries) richiedono che ciascun gestore risorse XA fornisca il proprio nome. Questa è la stringa denominata name nella struttura dello switch XA. Il gestore risorse per WebSphere MQ su sistemi UNIX, Linux e Windows è denominato MQSeries_XA_RMI. Per ulteriori dettagli sulle interfacce XA, consultare la documentazione XA *CAE Specification Distributed Transaction Processing: The XA Specification*, pubblicata da The Open Group.

In una configurazione XA, WebSphere MQ su sistemi UNIX, Linuxe Windows ricoprono il ruolo di XA Resource Manager. Un coordinatore del punto di sincronizzazione XA può gestire una serie di gestori risorse XA e sincronizzare il commit o il backout delle transazioni in entrambi i gestori risorse. Questo è il modo in cui funziona per un gestore risorse registrato staticamente:

1. Un'applicazione notifica al coordinatore del punto di sincronizzazione che desidera avviare una transazione.
2. Il coordinatore del punto di sincronizzazione invia una chiamata a tutti i gestori risorse di cui è a conoscenza, per notificare loro la transazione corrente.
3. L'applicazione emette chiamate per aggiornare le risorse gestite dai gestori risorse associati alla transazione corrente.
4. L'applicazione richiede che il coordinatore del punto di sincronizzazione esegua il commit o il rollback della transazione.
5. Il coordinatore del punto di sincronizzazione emette chiamate a ciascun gestore risorse utilizzando protocolli di commit a due fasi per completare la transazione come richiesto.

La specifica XA richiede ogni Resource Manager per fornire una struttura denominata *Switch XA*. Questa struttura dichiara le funzioni del Resource Managere le funzioni che devono essere richiamate dal coordinatore del punto di sincronizzazione.

Esistono due versioni di questa struttura:

MQRMIXASwitch	Gestione delle risorse XA statici
MQRMIXASwitchDynamic	Gestione dinamica delle risorse XA

Per un elenco delle librerie contenenti questa struttura, consultare "La struttura dello switch IBM WebSphere MQ XA" a pagina 70.

Il metodo che deve essere utilizzato per collegarli a un coordinatore del punto di sincronizzazione XA è definito dal coordinatore; consultare la documentazione fornita da tale coordinatore per stabilire come abilitare WebSphere MQ a cooperare con il coordinatore del punto di sincronizzazione XA.

La struttura *xa_info* che viene passata su qualsiasi chiamata *xa_open* dal coordinatore del punto di sincronizzazione può essere il nome del gestore code che deve essere gestito. Questo assume lo stesso formato del nome del gestore code passato a MQCONN o MQCONNX e può essere vuoto se deve essere utilizzato il gestore code predefinito. Tuttavia, è possibile utilizzare i due parametri aggiuntivi TPM e AXLIB

TPM consente di specificare per WebSphere MQ il nome del gestore transazioni, ad esempio CICS. AXLIB consente di specificare il nome effettivo della libreria nel gestore transazioni in cui si trovano i punti di ingresso XA AX.

Se si utilizza uno di questi parametri o un gestore code non predefinito, è necessario specificare il nome del gestore code utilizzando il parametro QMNAME. Per ulteriori informazioni, consultare I parametri CHANNEL, TRPTYPE, CONNAME e QMNAME della stringa xa_open.

Limitazioni

1. Le unità di lavoro globali non sono consentite con un Hconn condiviso (come descritto in "Connessioni condivise (indipendenti dal thread) con MQCONNX" a pagina 213.
2. Su sistemi Windows , tutte le funzioni dichiarate nello switch XA sono dichiarate come funzioni _cdecl.
3. Un coordinatore del punto di sincronizzazione esterno può gestire solo un gestore code alla volta. Ciò è dovuto al fatto che il coordinatore ha una connessione effettiva a ciascun gestore code ed è pertanto soggetto alla regola che è consentita solo una connessione alla volta.

   Nota: Nota: un'applicazione client JMS (applicazione CLIENT JEE) in esecuzione su un server JEE, non ha questa limitazione, quindi una singola transazione gestita dal server JEE può coordinare più gestori code nella stessa transazione. Tuttavia, un'applicazione server JMS, in esecuzione in modalità bind, è ancora soggetta alla regola che consente una sola connessione alla volta.
4. Tutte le applicazioni che vengono eseguite utilizzando il coordinatore del punto di sincronizzazione possono connettersi solo al gestore code gestito dal coordinatore perché sono già effettivamente connesse a tale gestore code. Devono emettere MQCONN o MQCONNX per ottenere un handle di connessione e devono emettere MQDISC prima di uscire. In alternativa, possono utilizzare l'uscita UE014015 per TXSeries CICS.

Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger

Informazioni sui trigger e su come avviare le applicazioni IBM WebSphere MQ utilizzando i trigger.

Alcune applicazioni WebSphere MQ che servono le code vengono eseguite continuamente, in modo che siano sempre disponibili per richiamare i messaggi che arrivano sulle code. Tuttavia, è possibile che ciò non sia necessario quando il numero di messaggi in arrivo nelle code è imprevedibile. In questo caso, le applicazioni potrebbero utilizzare le risorse di sistema anche quando non ci sono messaggi da recuperare.

WebSphere MQ fornisce una funzione che abilita l'avvio automatico di un'applicazione quando sono disponibili messaggi da richiamare. Questa funzione è nota come *attivazione*.

Per informazioni sull'attivazione dei canali, vedere Trigger dei canali .

Cos' è l'attivazione?

Il gestore code definisce alcune condizioni che costituiscono *eventi trigger*.

Se il trigger è abilitato per una coda e si verifica un evento trigger, il gestore code invia un messaggio di *trigger* a una coda denominata *coda di iniziazione*. La presenza del messaggio trigger sulla coda di iniziazione indica che si è verificato un evento trigger.

I messaggi trigger generati dal gestore code non sono persistenti. Ciò riduce la registrazione (migliorando le prestazioni) e riducendo i duplicati durante il riavvio, migliorando il tempo di riavvio.

Il programma che elabora la coda di iniziazione viene chiamato *applicazione trigger - monitor*e la sua funzione è quella di leggere il messaggio del trigger e intraprendere l'azione appropriata, in base alle informazioni contenute nel messaggio del trigger. Di solito, questa azione consente di avviare un'altra applicazione per elaborare la coda che ha generato il messaggio trigger. Dal punto di vista del gestore code, non c'è nulla di speciale nell'applicazione di controllo dei trigger; si tratta semplicemente di un'altra applicazione che legge i messaggi da una coda (la coda di avvio).

Se il trigger è abilitato per una coda, è possibile creare un *oggetto di definizione del processo* ad esso associato. Questo oggetto contiene informazioni sull'applicazione che elabora il messaggio che ha causato l'evento trigger. Se l'oggetto di definizione del processo viene creato, il gestore code estrae queste informazioni e le inserisce nel messaggio del trigger, per l'utilizzo da parte dell'applicazione di controllo dei trigger. Il nome della definizione del processo associato a una coda viene fornito dall'attributo della coda locale *ProcessName* . Ogni coda può specificare una definizione di processo differente oppure diverse code possono condividere la stessa definizione.

Se si desidera attivare l'avvio di un canale, non è necessario definire un oggetto definizione processo. Viene utilizzata invece la definizione della coda di trasmissione.

Il trigger è supportato da client WebSphere MQ in esecuzione nei seguenti ambienti:

- Sistemi UNIX and Linux
- Sistemi Windows

Un'applicazione in esecuzione in un ambiente client è uguale a quella in esecuzione in un ambiente WebSphere MQ completo, ad eccezione del fatto che viene collegata alle librerie client. Tuttavia, il controllo trigger e l'applicazione da avviare devono essere entrambi nello stesso ambiente.

L'attivazione implica:

Coda applicazione
Una *coda dell'applicazione* è una coda locale che, quando ha il trigger impostato e quando le condizioni vengono soddisfatte, richiede la scrittura dei messaggi del trigger.

Definizione di processo
Una coda dell'applicazione può avere un *oggetto definizione processo* associato ad essa che contiene i dettagli dell'applicazione che acquisirà i messaggi dalla coda dell'applicazione. (Consultare Attributi per definizioni di processi per un elenco di attributi.)

Si ricordi che se si desidera che un trigger avvii un canale, non è necessario definire un oggetto definizione processo.

Coda di trasmissione
È necessaria una coda di trasmissione se si desidera che un trigger avvii un canale.

Per una coda di trasmissione su sistemi AIX, HP-UX, IBM i, Solaris, z/OSo Windows , l'attributo *TriggerData* della coda di trasmissione può specificare il nome del canale da avviare. Può sostituire la definizione del processo per i canali di attivazione, ma viene utilizzata solo quando non viene creata una definizione del processo.

Evento trigger
Un *evento trigger* è un evento che causa la generazione di un messaggio trigger da parte del gestore code. Questo è di solito un messaggio che arriva su una coda dell'applicazione, ma può verificarsi anche in altri momenti (consultare “Condizioni per un evento trigger” a pagina 338). WebSphere MQ

ha una gamma di opzioni che consentono di controllare le condizioni che causano un evento trigger (consultare “Controllo degli eventi trigger” a pagina 341).

messaggio di trigger
Il gestore code crea un *messaggio trigger* quando riconosce un evento trigger (consultare “Condizioni per un evento trigger” a pagina 338). Copia nel messaggio trigger le informazioni sull'applicazione da avviare. Queste informazioni provengono dalla coda dell'applicazione e dall'oggetto di definizione processo associato alla coda dell'applicazione. I messaggi trigger hanno un formato fisso (consultare “Formato dei messaggi di trigger” a pagina 348).

Inizializzazione coda
Una *coda di inizializzazione* è una coda locale in cui il gestore code inserisce i messaggi del trigger. Si noti che una coda di iniziazione non può essere una coda alias o una coda modello. Un gestore code può essere proprietario di più di una coda di iniziazione e ciascuna di esse è associata a una o più code dell'applicazione. Una coda condivisa, una coda locale accessibile dai gestori code in un gruppo di condivisione code, può essere una coda di avvio su WebSphere MQ per z/OS.

Controllo trigger
Un *controllo trigger* è un programma in esecuzione continua che serve una o più code di iniziazione. Quando un messaggio di trigger arriva su una coda di iniziazione, il controllo di trigger richiama il messaggio. Il controllo trigger utilizza le informazioni nel messaggio trigger. Immette un comando per avviare l'applicazione che deve richiamare i messaggi in arrivo sulla coda dell'applicazione, passando le informazioni contenute nell'intestazione del messaggio trigger, che include il nome della coda dell'applicazione.

Su tutte le piattaforme, uno speciale controllo trigger noto come iniziatore di canali è responsabile dell'avvio dei canali. Su z/OS, l'iniziatore di canali viene generalmente avviato manualmente oppure può essere eseguito automaticamente quando un gestore code viene avviato modificando CSQINP2 nel JCL di avvio del gestore code. Su altre piattaforme, viene avviato automaticamente all'avvio del gestore code o può essere avviato manualmente con il comando runmqchi.

(Per ulteriori informazioni, consultare “Elaborazione della coda di iniziazione da parte dei controlli trigger” a pagina 345.)

Per comprendere il funzionamento del trigger, considerare Figura 68 a pagina 334, che è un esempio di tipo di trigger FIRST (MQTT_FIRST).

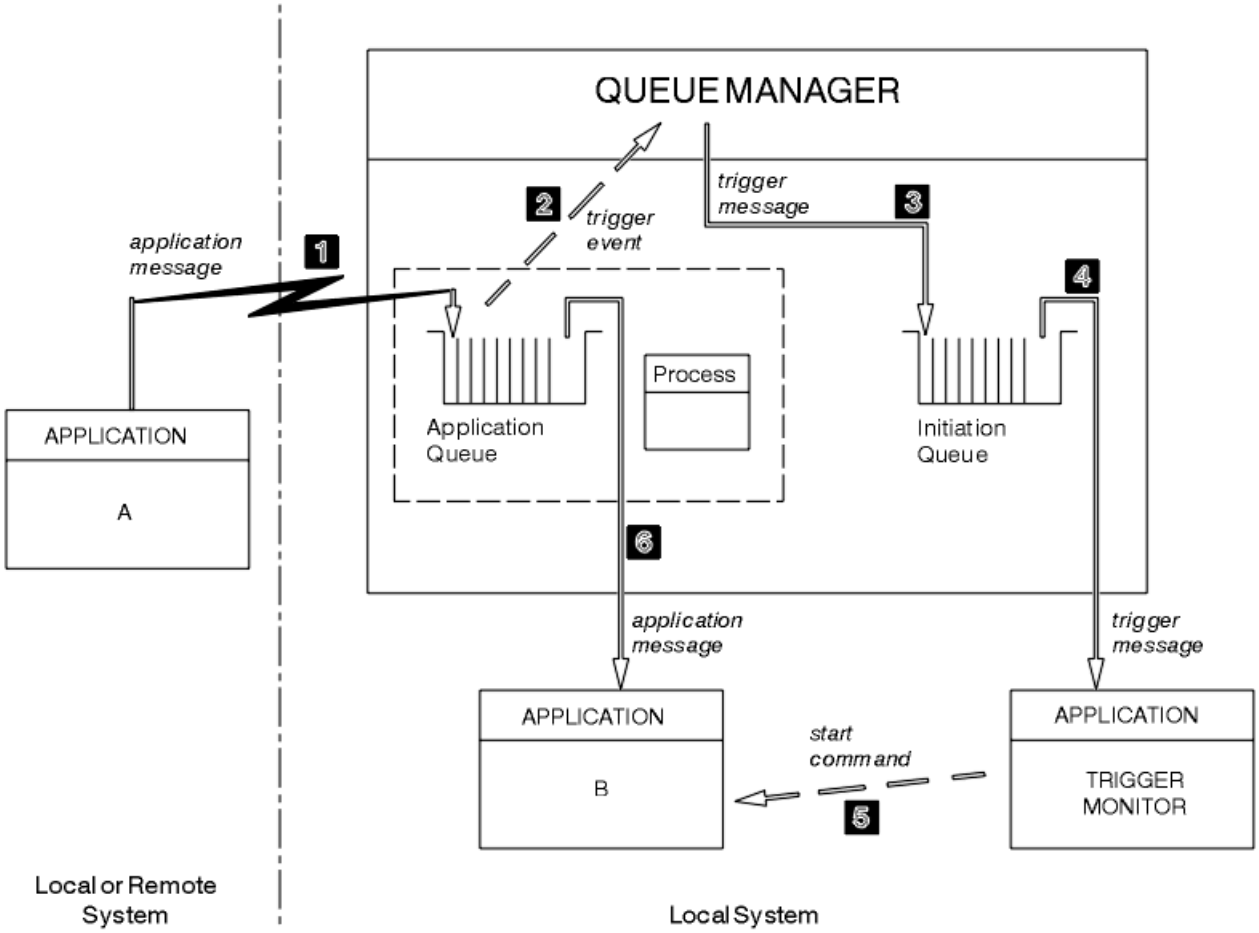


Figura 68. Flusso di messaggi di applicazione e trigger

In Figura 68 a pagina 334, la sequenza di eventi è:

1. L'applicazione A, che può essere locale o remota per il gestore code, inserisce un messaggio nella coda dell'applicazione. Nessuna applicazione ha questa coda aperta per l'immissione. Tuttavia, questo fatto è rilevante solo per attivare il tipo FIRST e DEPTH.
2. Il gestore code controlla se sono soddisfatte le condizioni in base alle quali deve generare un evento trigger. Lo sono e viene generato un evento trigger. Le informazioni contenute nell'oggetto di definizione processo associato vengono utilizzate durante la creazione del messaggio trigger.
3. Il gestore code crea un messaggio trigger e lo inserisce nella coda di iniziazione associata a questa coda dell'applicazione, ma solo se un'applicazione (controllo trigger) ha la coda di iniziazione aperta per l'input.
4. Il controllo trigger richiama il messaggio trigger dalla coda di iniziazione.
5. Il controllo trigger emette un comando per avviare l'applicazione B (l'applicazione server).
6. L'applicazione B apre la coda dell'applicazione e richiama i messaggi.

Nota:

1. Se la coda dell'applicazione è aperta per l'input, da qualsiasi programma, e ha il trigger impostato per FIRST o DEPTH, non si verifica alcun evento trigger perché la coda è già servita.
2. Se la coda di iniziazione non è aperta per l'input, il gestore code non genera alcun messaggio trigger; attende fino a quando un'applicazione non apre la coda di iniziazione per l'input.
3. Quando si utilizza il trigger per i canali, utilizzare il tipo di trigger FIRST o DEPTH.

4. Le applicazioni attivate vengono eseguite con l'ID utente e il gruppo dell'utente che ha avviato il controllo dei trigger, l'utente CICS o l'utente che ha avviato il gestore code.

Finora, la relazione tra le code all'interno del trigger è stata solo su una base uno - a - uno. Considerare Figura 69 a pagina 335.

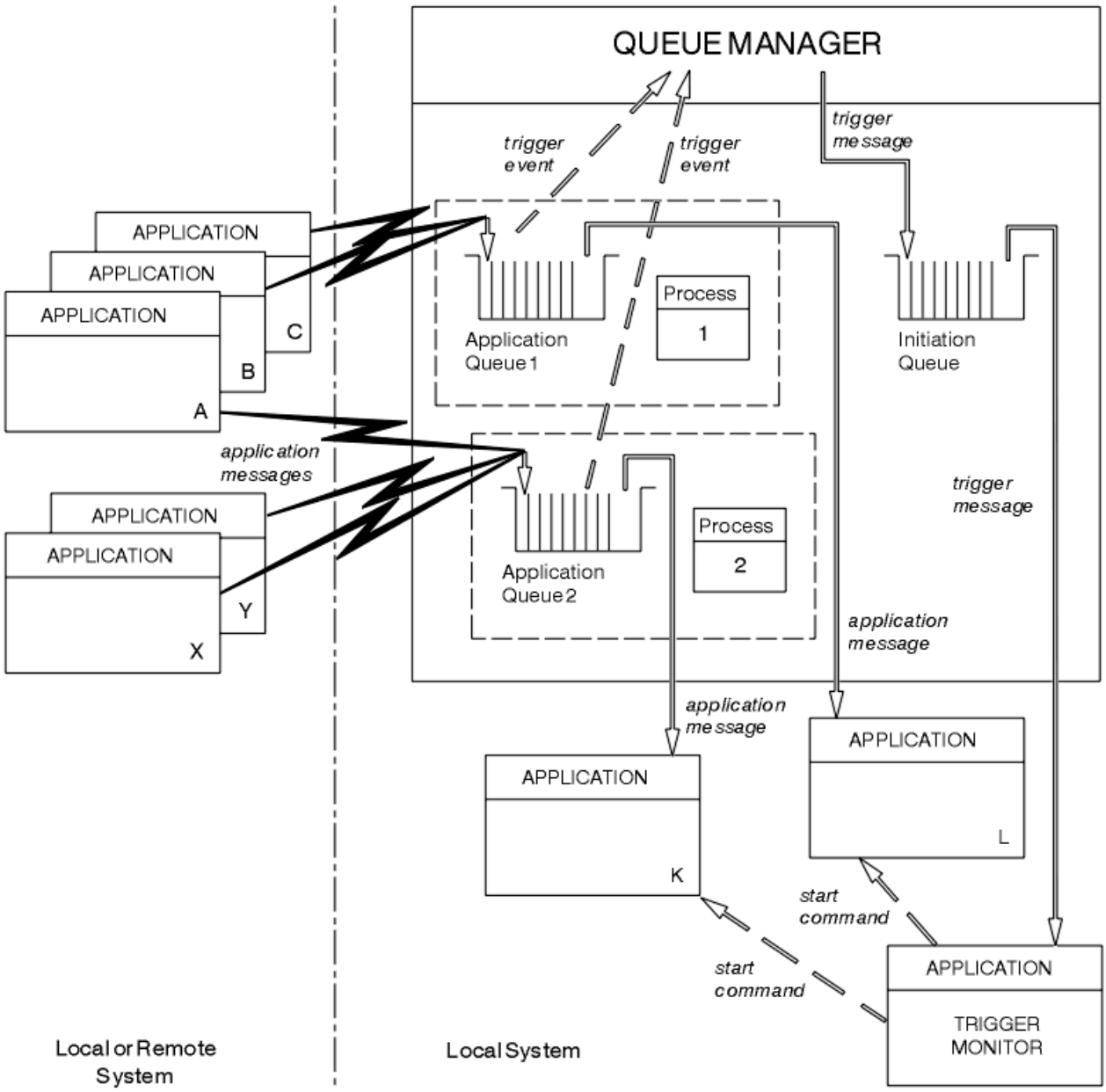


Figura 69. Relazione delle code all'interno del trigger

Una coda dell'applicazione ha un oggetto di definizione processo associato che contiene i dettagli dell'applicazione che elaborerà il messaggio. Il gestore code inserisce le informazioni nel messaggio del trigger, pertanto è necessaria solo una coda di avvio. Il controllo dei trigger estrae queste informazioni dal messaggio del trigger e avvia l'applicazione pertinente per gestire il messaggio su ciascuna coda dell'applicazione.

Tenere presente che, se si desidera attivare l'avvio di un canale, non è necessario definire un oggetto definizione processo. La definizione della coda di trasmissione può determinare il canale da attivare.

Utilizzare i seguenti collegamenti per ulteriori informazioni sull'avvio delle applicazioni WebSphere MQ mediante i trigger:

- “Prerequisiti per l'attivazione” a pagina 336

- “Condizioni per un evento trigger” a pagina 338
- “Controllo degli eventi trigger” a pagina 341
- “Progettazione di un'applicazione che utilizza code attivate” a pagina 344
- “Elaborazione della coda di iniziazione da parte dei controlli trigger” a pagina 345
- “Proprietà dei messaggi trigger” a pagina 347
- “Quando l'attivazione non funziona” a pagina 349

Concetti correlati

“Panoramica su Message Queue Interface” a pagina 195
Informazioni sui componenti MQI (Message Queue Interface).

“Connessione e disconnessione da un gestore code” a pagina 207
Per utilizzare i servizi di programmazione WebSphere MQ , un programma deve disporre di una connessione a un gestore code. Utilizzare queste informazioni per informazioni su come connettersi e disconnettersi da un gestore code.

“Apertura e chiusura di oggetti” a pagina 215
Queste informazioni forniscono informazioni sull'apertura e la chiusura degli oggetti WebSphere MQ .

“Inserimento di messaggi in una coda” a pagina 225
Utilizzare queste informazioni per informazioni su come inserire i messaggi su una coda.

“Richiamo dei messaggi da una coda” a pagina 241
Utilizzare queste informazioni per ottenere messaggi da una coda.

“Richiesta di informazioni e impostazione degli attributi dell'oggetto” a pagina 322
Gli attributi sono proprietà che definiscono le caratteristiche di un oggetto WebSphere MQ .

“Commit e backout delle unità di lavoro” a pagina 325
Queste informazioni descrivono come eseguire il commit e il backout di tutte le operazioni get e put recuperabili che si sono verificate in un'unità di lavoro.

“Utilizzo di MQI e cluster” a pagina 349
Esistono opzioni speciali sulle chiamate e sui codici di ritorno relativi al cluster.

Prerequisiti per l'attivazione

Utilizzare queste informazioni per informazioni sui passi da eseguire prima di utilizzare il trigger.

Prima che la tua applicazione possa sfruttare l'attivazione, completa la seguente procedura:

1. Le alternative sono:
   a. Creare una coda di iniziazione per la coda dell'applicazione. Ad esempio:

```
DEFINE QLOCAL (initiation.queue) REPLACE      +
       LIKE (SYSTEM.DEFAULT.INITIATION.QUEUE)      +
       DESCR ('initiation queue description')
```

   o
   b. Determinare il nome di una coda locale esistente e che può essere utilizzata dall'applicazione (di solito, questo nome è SYSTEM.DEFAULT.INITIATION.QUEUE o, se si stanno avviando canali con trigger, SYSTEM.CHANNEL.INITQ) e specificarne il nome nel campo *InitiationQName* della coda dell'applicazione.

2. Associare la coda di iniziazione alla coda dell'applicazione. Un gestore code può possedere più di una coda di iniziazione. È possibile che si desideri che alcune delle code dell'applicazione vengano servite da programmi differenti, in tal caso, è possibile utilizzare una coda di avvio per ciascun programma

di utilizzo, anche se non è necessario. Di seguito è riportato un esempio di come creare una coda dell'applicazione:

```
DEFINE QLOCAL (application.queue) REPLACE   +
LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE)           +
DESCR ('appl queue description')    +
INITQ ('initiation.queue')          +
PROCESS ('process.name')            +
TRIGGER                                     +
TRIGTYPE (FIRST)
```

3. Se si sta attivando un'applicazione, creare un oggetto di definizione del processo per contenere le informazioni relative all'applicazione che deve servire la coda dell'applicazione. Ad esempio, per attivare - avviare una transazione di retribuzione CICS denominata PAYR:

```
DEFINE PROCESS (process.name) +
       REPLACE +
       DESCR ('process description') +
       APPLICID ('PAYR') +
       APPLTYPE (CICS) +
       USERDATA ('Payroll data')
```

   Quando il gestore code crea un messaggio trigger, copia le informazioni dagli attributi dell'oggetto definizione del processo nel messaggio trigger.

Piattaforma	**Per creare un oggetto definizione processo**
Sistemi UNIX, Linuxe Windows	Utilizzare DEFINE PROCESS o SYSTEM.DEFAULT.PROCESS e modifica utilizzando ALTER PROCESS

4. Facoltativo: creare una definizione di coda di trasmissione e utilizzare spazi vuoti per l'attributo *ProcessName* .

   L'attributo *TrigData* può contenere il nome del canale da attivare o può essere lasciato vuoto. Tranne su IBM WebSphere MQ for z/OS, se viene lasciato vuoto, l'iniziatore del canale ricerca i file di definizione del canale fino a quando non trova un canale associato alla coda di trasmissione denominata. Quando il gestore code crea un messaggio trigger, copia le informazioni dall'attributo *TrigData* della definizione della coda di trasmissione nel messaggio trigger.

5. Se è stato creato un oggetto di definizione del processo per specificare le proprietà dell'applicazione che deve servire la coda dell'applicazione, associare l'oggetto del processo alla coda dell'applicazione denominandolo nell'attributo *ProcessName* della coda.

Piattaforma	**Utilizza comandi**
Sistemi UNIX, Linuxe Windows	MODIFICA QLOCAL

6. Avviare le istanze dei controlli dei trigger che servono le code di iniziazione definite. Per ulteriori informazioni, fare riferimento a “Elaborazione della coda di iniziazione da parte dei controlli trigger” a pagina 345.

Se si desidera essere a conoscenza di eventuali messaggi di trigger non recapitati, assicurarsi che il gestore code disponga di una coda di messaggi non recapitabili (non recapitabile) definita. Specificare il nome della coda nel campo Gestore code *DeadLetterQName* .

È quindi possibile impostare le condizioni trigger richieste, utilizzando gli attributi dell'oggetto coda che definisce la coda dell'applicazione. Per ulteriori informazioni, vedere “Controllo degli eventi trigger” a pagina 341.

Condizioni per un evento trigger

I riferimenti alle code condivise in questo argomento indicano code condivise in un gruppo di condivisione code, disponibili solo su WebSphere MQ per z/OS.

Il gestore code crea un messaggio trigger quando vengono soddisfatte le seguenti condizioni:

1. Un messaggio è *inserito* su una coda.
2. Il messaggio ha una priorità maggiore o uguale alla priorità del trigger di soglia della coda. Questa priorità è impostata nell'attributo della coda locale *TriggerMsgPriority* ; se è impostata su zero, qualsiasi messaggio è qualificato.
3. Il numero di messaggi sulla coda con priorità maggiore o uguale a *TriggerMsgPriority* era in precedenza, a seconda di *TriggerType*:
   - Zero (per il tipo di trigger MQTT_FIRST)
   - Qualsiasi numero (per il tipo di trigger MQTT_EVERY)
   - *TriggerDepth* meno 1 (per il tipo di trigger MQTT_DEPTH)

   Nota:

   a. Per le code locali non condivise, il gestore code conta i messaggi di cui è stato eseguito il commit e di cui non è stato eseguito il commit quando valuta se esistono le condizioni per un evento trigger. Di conseguenza, un'applicazione potrebbe essere avviata quando non ci sono messaggi da richiamare poiché non è stato eseguito il commit dei messaggi sulla coda. In questa situazione, considerare l'utilizzo dell'opzione di attesa con un *WaitInterval*adatto, in modo che l'applicazione attenda l'arrivo dei messaggi.

   b. Per le code condivise locali, il gestore code conta solo i messaggi di cui è stato eseguito il commit.
4. Per l'attivazione di tipo FIRST o DEPTH, nessun programma ha la coda dell'applicazione aperta per la rimozione dei messaggi (ossia, l'attributo della coda locale *OpenInputCount* è zero).

   Nota:

   a. Per le code condivise, si applicano condizioni speciali quando più gestori code dispongono di controlli trigger in esecuzione su una coda. In questa situazione, se uno o più gestori code hanno la coda aperta per l'input condiviso, i criteri di trigger sugli altri gestori code vengono considerati come *TriggerType* MQTT_FIRST e *TriggerMsgPriority* zero. Quando tutti i gestori code chiudono la coda per l'input, le condizioni di trigger ritornano a quelle specificate nella definizione della coda.

      Uno scenario di esempio interessato da questa condizione è rappresentato da più gestori code QM1, QM2e QM3 con un controllo trigger in esecuzione per una coda dell'applicazione A. Un messaggio arriva su A che soddisfa le condizioni per l'attivazione e viene generato un messaggio di trigger sulla coda di iniziazione. Il controllo trigger su QM1 richiama il messaggio trigger e attiva un'applicazione. L'applicazione attivata apre la coda dell'applicazione per l'input condiviso. Da questo punto in poi le condizioni di trigger per la coda dell'applicazione A vengono valutate come *TriggerType* MQTT_FIRST e *TriggerMsgPriority* zero sui gestori code QM2 e QM3, fino a quando QM1 non chiude la coda dell'applicazione.

   b. Per le code condivise, questa condizione viene applicata per ciascun gestore code. In altre parole, il valore *OpenInputCount* di un gestore code per una coda deve essere zero affinché un messaggio trigger venga generato per la coda da tale gestore code. Tuttavia, se un qualsiasi gestore code nel gruppo di condivisione code ha la coda aperta utilizzando l'opzione MQOO_INPUT_EXCLUSIVE, non viene generato alcun messaggio trigger per tale coda da uno dei gestori code nel gruppo di condivisione code.

      La modifica nella modalità di valutazione delle condizioni di trigger si verifica quando l'applicazione attivata apre la coda per l'input. Negli scenari in cui è in esecuzione un solo controllo trigger, altre applicazioni possono avere lo stesso effetto perché aprono in modo simile la coda dell'applicazione per l'input. Non importa se la coda dell'applicazione è stata aperta da

un'applicazione avviata da un controllo dei trigger o da un'altra applicazione; è il fatto che la coda è aperta per l'input su un altro gestore code che causa la modifica nei criteri dei trigger.

5. Su WebSphere MQ per z/OS, se la coda dell'applicazione è una con un attributo *Usage* MQUS_NORMAL, le richieste get per essa non sono inibite (ossia, l'attributo della coda *InhibitGet* è MQQA_GET_ALLOWED). Inoltre, se la coda dell'applicazione attivata è una coda con un attributo *Usage* di MQUS_XMITQ, le richieste get per essa non sono inibite.
6. Le alternative sono:
   - L'attributo della coda locale *ProcessName* per la coda non è vuoto e l'oggetto di definizione del processo identificato da tale attributo è stato creato oppure
   - L'attributo della coda locale *ProcessName* per la coda è vuoto, ma la coda è una coda di trasmissione. Poiché la definizione del processo è facoltativa, l'attributo *TriggerData* potrebbe contenere anche il nome del canale da avviare. In tal caso, il messaggio trigger contiene attributi con i seguenti valori:
     - *QName*: nome coda
     - *ProcessName*: spazi
     - *TriggerData*: dati trigger
     - *ApplType*: MQAT_SCONOSCIUTO
     - *ApplId*: spazi
     - *EnvData*: spazi
     - *UserData*: spazi
7. È stata creata una coda di iniziazione ed è stata specificata nell'attributo della coda locale *InitiationQName* . Inoltre:
   - Le richieste di richiamo non sono inibite per la coda di avvio (ossia, l'attributo della coda *InhibitGet* è MQQA_GET_ALLOWED).
   - Le richieste di inserimento non devono essere inibite per la coda di avvio (ovvero, l'attributo della coda *InhibitPut* deve essere MQQA_PUT_ALLOWED).
   - L'attributo *Usage* della coda di iniziazione deve essere MQUS_NORMAL.
   - In ambienti in cui sono supportate code dinamiche, la coda di iniziazione non deve essere una coda dinamica contrassegnata come eliminata logicamente.
8. Un controllo trigger attualmente ha la coda di iniziazione aperta per la rimozione dei messaggi (ovvero, l'attributo della coda locale *OpenInputCount* è maggiore di zero).
9. Il controllo trigger (attributo della coda locale*TriggerControl*) per la coda dell'applicazione è impostato su MQTC_ON. Per eseguire questa operazione, impostare l'attributo *trigger* quando si definisce la coda oppure utilizzare il comando ALTER QLOCAL.
10. Il tipo di trigger (attributo coda locale*TriggerType*) non è MQTT_NONE.

    Se vengono soddisfatte tutte le condizioni richieste e il messaggio che ha causato la condizione del trigger viene inserito come parte di un'unità di lavoro, il messaggio del trigger non diventa disponibile per il richiamo da parte dell'applicazione di controllo del trigger fino a quando l'unità di lavoro non viene completata, se è stato eseguito il commit dell'unità di lavoro o, per il tipo di trigger MQTT_FIRST o MQTT_DEPTH, è stato eseguito il backout.
11. Un messaggio adatto viene inserito nella coda, per un *TriggerType* di MQTT_FIRST o MQTT_DEPTH e la coda:
    - Non era precedentemente vuoto (MQTT_FIRST) o
    - Aveva *TriggerDepth* o più messaggi (MQTT_DEPTH)

    e le condizioni da “2” a pagina 338 a “10” a pagina 339 (escluso “3” a pagina 338) sono soddisfatte, se nel caso di MQTT_FIRST è trascorso un intervallo sufficiente (attributo gestore code*TriggerInterval*) da quando è stato scritto l'ultimo messaggio trigger per questa coda.

Ciò consente a un server di coda di terminare prima di elaborare tutti i messaggi sulla coda. Lo scopo dell'intervallo di trigger è quello di ridurre il numero di messaggi di trigger duplicati generati.

Nota: Se si arresta e si riavvia il gestore code, il *TriggerInterval timer* viene reimpostato. C'è una piccola finestra durante la quale è possibile produrre due messaggi trigger. La finestra esiste quando l'attributo trigger della coda è impostato su abilitato contemporaneamente all'arrivo di un messaggio e la coda non era precedentemente vuota (MQTT_FIRST) o aveva *TriggerDepth* o più messaggi (MQTT_DEPTH).

12. L'unica applicazione che serve una coda emette una chiamata MQCLOSE, per un *TriggerType* di MQTT_FIRST o MQTT_DEPTH, ed è presente almeno:

    - Uno (MQTT_FIRST) o
    - *TriggerDepth* (MQTT_DEPTH)

    i messaggi sulla coda di priorità sufficiente (condizione “2” a pagina 338) e le condizioni da a “10” a pagina 339 sono soddisfatte.

    Ciò consente un server di coda che emette una chiamata MQGET, trova la coda vuota e quindi termina; tuttavia, nell'intervallo tra le chiamate MQGET e MQCLOSE, arrivano uno o più messaggi.

    Nota:

    a. Se il programma che serve la coda dell'applicazione non richiama tutti i messaggi, ciò può causare un loop chiuso. Ogni volta che il programma chiude la coda, il gestore code crea un altro messaggio trigger che fa sì che il controllo dei trigger avvii di nuovo il programma server.

    b. Se il programma che serve la coda dell'applicazione esegue il backout della richiesta di richiamo (o se il programma termina in modo anomalo) prima di chiudere la coda, lo stesso accade. Tuttavia, se il programma chiude la coda prima di eseguire il backout della richiesta di richiamo e la coda è altrimenti vuota, non viene creato alcun messaggio trigger.

    c. Per evitare che si verifichi un loop di questo tipo, utilizzare il campo *BackoutCount* di MQMD per rilevare i messaggi di cui viene ripetutamente eseguito il backout. Per ulteriori informazioni, vedere “Messaggi di cui è stato eseguito il backout” a pagina 38.

13. Le seguenti condizioni vengono soddisfatte utilizzando MQSET o un comando:

    a. • *TriggerControl* è stato modificato in MQTC_ON oppure
    • *TriggerControl* è già MQTC_ON e il valore di *TriggerType*, *TriggerMsgPriority*o *TriggerDepth* (se pertinente) viene modificato,

    e ci sono almeno:

    - Uno (MQTT_FIRST o MQTT_EVERY) o
    - *TriggerDepth* (MQTT_DEPTH)

    i messaggi sulla coda con priorità sufficiente (condizione “2” a pagina 338) e le condizioni da “4” a pagina 338 a “10” a pagina 339 (escluso “8” a pagina 339) vengono soddisfatte.

    Ciò consente a un'applicazione o a un operatore di modificare i criteri di attivazione, quando le condizioni per il verificarsi di un trigger sono già soddisfatte.

    b. L'attributo della coda *InhibitPut* di una coda di iniziazione cambia da MQQA_PUT_INIBITED a MQQA_PUT_ALLOWED e c'è almeno:

    - Uno (MQTT_FIRST o MQTT_EVERY) o
    - *TriggerDepth* (MQTT_DEPTH)

    messaggi di priorità sufficiente (condizione “2” a pagina 338) su una qualsiasi delle code per cui questa è la coda di iniziazione e le condizioni “4” a pagina 338 tramite “10” a pagina 339 vengono soddisfatte. (Viene generato un messaggio trigger per ciascuna coda che soddisfa le condizioni.)

    Ciò consente ai messaggi trigger di non essere generati a causa della condizione MQQA_PUT_INIITED sulla coda di avvio, ma questa condizione ora è stata modificata.

c. L'attributo della coda *InhibitGet* di una coda dell'applicazione viene modificato da MQQA_GET_INIBITED a MQQA_GET_ALLOWED ed è presente almeno:

   - Uno (MQTT_FIRST o MQTT_EVERY) o
   - *TriggerDepth* (MQTT_DEPTH)

   i messaggi con priorità sufficiente (condizione “2” a pagina 338) sulla coda e le condizioni “4” a pagina 338 tramite “10” a pagina 339, escluso “5” a pagina 339, vengono soddisfatte.

   Ciò consente alle applicazioni di essere attivati solo quando possono richiamare i messaggi dalla coda dell'applicazione.

d. Un'applicazione di controllo dei trigger emette una chiamata MQOPEN per l'immissione da una coda di avvio e vi è almeno:

   - Uno (MQTT_FIRST o MQTT_EVERY) o
   - *TriggerDepth* (MQTT_DEPTH)

   i messaggi di priorità sufficiente (condizione “2” a pagina 338) su qualsiasi coda dell'applicazione per cui questa è la coda di iniziazione e le condizioni “4” a pagina 338 tramite “10” a pagina 339 (escluso “8” a pagina 339) sono soddisfatte e nessun' altra applicazione ha la coda di iniziazione aperta per l'input (viene generato un messaggio trigger per ogni coda che soddisfa le condizioni).

   Ciò consente ai messaggi in arrivo sulle code mentre non è in esecuzione il controllo dei trigger e al gestore code di riavviare e attivare i messaggi (non persistenti) che vengono persi.

14. MSGDLVSQ è impostato correttamente. Se si imposta MSGDLVSQ=FIFO, i messaggi vengono consegnati alla coda in base al primo in uscita. La priorità del messaggio viene ignorata e la priorità predefinita della coda viene assegnata al messaggio. Se *TriggerMsgPriority* è impostato su un valore superiore rispetto alla priorità predefinita della coda, non viene attivato alcun messaggio. Se *TriggerMsgPriority* è impostato su un valore uguale o inferiore alla priorità predefinita della coda, l'attivazione si verifica per il tipo FIRST, EVERY e DEPTH. Per informazioni su questi tipi, consultare la descrizione del campo *TriggerType* in “Controllo degli eventi trigger” a pagina 341.

    Se si imposta MSGDLVSQ=PRIORITY e la priorità del messaggio è maggiore o uguale al campo *TriggerMsgPriority* , solo i messaggi *contano* per un evento trigger. In questo caso, il trigger si verifica per il tipo FIRST, EVERY e DEPTH. Ad esempio, se si immettono 100 messaggi di priorità inferiore rispetto a *TriggerMsgPriority*, la profondità della coda effettiva per l'attivazione è ancora zero. Se si inserisce un altro messaggio sulla coda, ma questa volta la priorità è maggiore o uguale a *TriggerMsgPriority*, la profondità effettiva della coda aumenta da zero a uno e la condizione per *TriggerType* FIRST viene soddisfatta.

Nota:

1. Dal passo “12” a pagina 340 (dove i messaggi trigger vengono generati come risultato di un evento diverso da un messaggio in arrivo sulla coda dell'applicazione), il messaggio trigger non viene inserito come parte di un'unità di lavoro. Inoltre, se *TriggerType* è MQTT_EVERY e se vi sono uno o più messaggi nella coda dell'applicazione, viene generato solo un messaggio trigger.
2. Se WebSphere MQ segmenta un messaggio durante MQPUT, un evento trigger non verrà elaborato fino a quando tutti i segmenti non saranno stati posizionati correttamente nella coda. Tuttavia, una volta che i segmenti di messaggio si trovano nella coda, WebSphere MQ li tratta come singoli messaggi per scopi di attivazione. Ad esempio, un singolo messaggio logico suddiviso in tre parti fa sì che venga elaborato solo un evento trigger quando viene prima MQPUT e segmentato. Tuttavia, ciascuno dei tre segmenti determina l'elaborazione dei propri eventi trigger man mano che vengono spostati nella rete WebSphere MQ .

Controllo degli eventi trigger

Gli eventi trigger vengono controllati utilizzando alcuni attributi che definiscono la coda dell'applicazione. Queste informazioni forniscono anche esempi di utilizzo dei tipi di trigger: EVERY, FIRST e DEPTH.

È possibile abilitare e disabilitare il trigger ed è possibile selezionare il numero o la priorità dei messaggi che contano per un evento trigger. Esiste una descrizione completa di questi attributi in Attributi degli oggetti.

Gli attributi rilevanti sono:

TriggerControl
Utilizzare questo attributo per abilitare e disabilitare il trigger per una coda di applicazioni.

TriggerMsgPriority
La priorità minima che un messaggio deve avere per essere conteggiato per un evento trigger. Se un messaggio con priorità inferiore a *TriggerMsgPriority* arriva sulla coda dell'applicazione, il gestore code ignora il messaggio quando determina se creare un messaggio trigger. Se *TriggerMsgPriority* è impostato su zero, tutti i messaggi vengono conteggiati per un evento trigger.

TriggerType
Oltre al tipo di trigger NONE (che disabilita il trigger come l'impostazione di *TriggerControl* su OFF), è possibile utilizzare i seguenti tipi di trigger per impostare la sensibilità di una coda per attivare gli eventi:

ogni	Un evento trigger si verifica ogni volta che un messaggio arriva sulla coda dell'applicazione. Utilizzare questo tipo di trigger se si desidera avviare più istanze di un'applicazione.
PRIMO	Un evento trigger si verifica solo quando il numero di messaggi sulla coda dell'applicazione cambia da zero a uno. Utilizzare questo tipo di trigger se si desidera che un programma di utilizzo venga avviato quando il primo messaggio arriva su una coda, continuare fino a quando non ci sono più messaggi da elaborare, quindi terminare. È sempre necessario elaborare la coda finché non è vuota. Consultare anche “Caso speciale del tipo di trigger FIRST” a pagina 343 .
PROFOND	Un evento trigger si verifica solo quando il numero di messaggi nella coda dell'applicazione raggiunge il valore dell'attributo *TriggerDepth* . Un uso tipico di questo tipo di attivazione è quello di avviare un programma quando vengono ricevute tutte le risposte a una serie di richieste. **Attivazione per profondità:** Con l'attivazione per profondità, il gestore code disabilita l'attivazione (utilizzando l'attributo < xph> < pv>TriggerControl< /pv> < /xph>) dopo aver creato un messaggio trigger. L'applicazione deve riabilitare l'attivazione stessa (utilizzando la chiamata MQSET) dopo che ciò si è verificato. L'azione di disabilitazione del trigger non è sotto il controllo del syncpoint, quindi il trigger non può essere riabilitato ripristinando un'unità di lavoro. Se un programma esegue il backout di una richiesta put che ha causato un evento trigger o se il programma termina in modo anomalo, è necessario riabilitare il trigger utilizzando la chiamata MQSET o il comando ALTER QLOCAL.

TriggerDepth
Il numero di messaggi su una coda che causa un evento trigger quando si utilizza l'attivazione per profondità.

Le condizioni che devono essere soddisfatte per un gestore code per creare un messaggio trigger sono descritte in “Condizioni per un evento trigger” a pagina 338 .

Esempio di utilizzo del tipo di trigger EVERY

Si consideri un'applicazione che genera richieste di assicurazione autoveicoli. L'applicazione potrebbe inviare messaggi di richiesta a un certo numero di compagnie di assicurazione, specificando la stessa

coda di risposta ogni volta. Potrebbe impostare un trigger di tipo EVERY su questa coda di risposta in modo che ogni volta che arriva una risposta, la risposta potrebbe attivare un'istanza del server per elaborare la risposta.

Esempio di utilizzo del tipo di trigger FIRST

Considerare un'organizzazione con un numero di filiali che trasmettono i dettagli dei giorni lavorativi alla sede principale. Lo fanno tutti allo stesso tempo, alla fine della giornata lavorativa, e presso la sede centrale c'è una applicazione che elabora i dettagli da tutte le filiali. Il primo messaggio che arriva alla sede principale potrebbe causare un evento trigger che avvia questa applicazione. Questa applicazione continuerà l'elaborazione fino a quando non ci saranno più messaggi sulla sua coda.

Esempio di utilizzo del tipo di trigger DEPTH

Prendere in considerazione un'applicazione dell'agenzia di viaggi che crea una singola richiesta per confermare una prenotazione di volo, per confermare una prenotazione per una camera d'albergo, per noleggiare un'auto e per ordinare alcuni controlli dei viaggiatori. L'applicazione potrebbe separare questi elementi in quattro messaggi di richiesta, inviando ciascuno a una destinazione separata. Potrebbe impostare un trigger di tipo DEPTH sulla relativa coda di risposta (con la profondità impostata sul valore 4), in modo che venga riavviato solo quando tutte e quattro le risposte sono arrivate.

Se un altro messaggio (probabilmente proveniente da una richiesta diversa) arriva sulla coda di risposta prima dell'ultima delle quattro risposte, l'applicazione richiedente viene attivata in anticipo. Per evitare ciò, quando si utilizza il trigger DEPTH per raccogliere più risposte a una richiesta, utilizzare sempre una nuova coda di risposta per ogni richiesta.

Caso speciale del tipo di trigger FIRST

Con il tipo di trigger FIRST, se è già presente un messaggio sulla coda dell'applicazione quando arriva un altro messaggio, il gestore code in genere non crea un altro messaggio trigger.

Tuttavia, l'applicazione che serve la coda potrebbe non aprire effettivamente la coda (ad esempio, l'applicazione potrebbe terminare, probabilmente a causa di un problema di sistema). Se è stato inserito un nome applicazione non corretto nell'oggetto di definizione del processo, l'applicazione che serve la coda non raccoglierà nessuno dei messaggi. In queste situazioni, se un altro messaggio arriva sulla coda dell'applicazione, non c'è alcun server in esecuzione per elaborare questo messaggio (e qualsiasi altro messaggio sulla coda).

Per risolvere questo problema, il gestore code crea ulteriori messaggi trigger nelle seguenti circostanze:

- Se un altro messaggio arriva sulla coda dell'applicazione, ma solo se è trascorso un intervallo di tempo predefinito da quando il gestore code ha creato l'ultimo messaggio trigger per quella coda. Questo intervallo di tempo è definito nell'attributo gestore code *TriggerInterval*. Il valore predefinito è 999 999 999 millisecondi.
- In WebSphere MQ per z/OS, le code dell'applicazione che denominano una coda di avvio aperta vengono sottoposte periodicamente a scansione. Se *TRIGINT* millisecondi sono trascorsi da quando è stato inviato l'ultimo messaggio trigger e la coda soddisfa le condizioni per un evento trigger e CURDEPTH è maggiore di zero, viene generato un messaggio trigger. Questo processo viene chiamato trigger di backstop.

Considerare i seguenti punti quando si decide un valore per l'intervallo di trigger da utilizzare nella propria applicazione:

- Se si imposta *TriggerInterval* su un valore basso e non c'è alcuna applicazione che serve la coda dell'applicazione, il tipo di trigger FIRST potrebbe comportarsi come tipo di trigger EVERY. Ciò dipende dalla frequenza con cui i messaggi vengono inseriti nella coda dell'applicazione, che a sua volta potrebbe dipendere da un'altra attività del sistema. Questo perché, se l'intervallo di trigger è molto piccolo, viene generato un altro messaggio di trigger ogni volta che un messaggio viene inserito nella coda dell'applicazione, anche se il tipo di trigger è FIRST, non EVERY. (Il tipo di trigger FIRST con un intervallo di trigger di zero è equivalente al tipo di trigger EVERY.)

- Su WebSphere MQ per z/OS se si imposta *TRIGINT* su un valore basso e non vi è alcuna applicazione che serve la coda dell'applicazione FIRST di tipo trigger, il trigger di backstop genererà un messaggio di trigger ogni volta che si verifica la scansione periodica delle code dell'applicazione che denominano le code di iniziazione aperte.
- Se viene eseguito il backout di un'unità di lavoro (consultare Messaggi di trigger e unità di lavoro) e l'intervallo di trigger è stato impostato su un valore elevato (o il valore predefinito), viene generato un messaggio di trigger quando viene eseguito il backout dell'unità di lavoro. Tuttavia, se l'intervallo di trigger è stato impostato su un valore basso o su zero (causando il comportamento del tipo di trigger FIRST come tipo di trigger EVERY), possono essere generati molti messaggi di trigger. Se viene eseguito il backout dell'unità di lavoro, tutti i messaggi trigger vengono ancora resi disponibili. Il numero di messaggi di trigger generati dipende dall'intervallo di trigger. Se l'intervallo di trigger è impostato su zero, viene generato il numero massimo di messaggi.

Progettazione di un'applicazione che utilizza code attivate

Si è visto come impostare e controllare l'attivazione per le applicazioni. Di seguito sono riportati alcuni consigli da considerare quando si progetta l'applicazione.

Attiva messaggi e unità di lavoro

I messaggi trigger creati a causa di eventi trigger che non fanno parte di un'unità di lavoro vengono inseriti nella coda di avvio, al di fuori di qualsiasi unità di lavoro, senza alcuna dipendenza da altri messaggi e sono immediatamente disponibili per il richiamo da parte del controllo trigger.

I messaggi trigger creati a causa di eventi trigger che fanno parte di un'unità di lavoro vengono resi disponibili sulla coda di iniziazione quando la UOW viene risolta, se è stato eseguito il commit o il backout dell'unità di lavoro

Se il gestore code non riesce a inserire un messaggio trigger in una coda di iniziazione, verrà inserito nella coda di messaggi non recapitabili (messaggio non recapito).

Nota:

1. Il gestore code conta sia i messaggi di cui è stato eseguito il commit che quelli di cui non è stato eseguito il commit quando valuta se esistono le condizioni per un evento trigger.

   Con l'attivazione di tipo FIRST o DEPTH, i messaggi di trigger vengono resi disponibili anche se viene eseguito il backout dell'unità di lavoro, in modo che un messaggio di trigger sia sempre disponibile quando vengono soddisfatte le condizioni richieste. Ad esempio, considerare una richiesta di inserimento all'interno di un'unità di lavoro per una coda attivata con il tipo di trigger FIRST. Ciò fa sì che il gestore code crei un messaggio trigger. Se si verifica un'altra richiesta di inserimento, da un'altra unità di lavoro, ciò non causa un altro evento trigger poiché il numero di messaggi nella coda dell'applicazione è ora cambiato da uno a due, il che non soddisfa le condizioni per un evento trigger. Ora, se viene eseguito il backout della prima unità di lavoro, ma viene eseguito il commit della seconda, viene comunque creato un messaggio di trigger.

   Tuttavia, ciò significa che i messaggi trigger vengono a volte creati quando le condizioni per un evento trigger *non sono* soddisfatte. Le applicazioni che utilizzano il trigger devono sempre essere preparate a gestire questa situazione. Si consiglia di utilizzare l'opzione di attesa con la chiamata MQGET, impostando *WaitInterval* su un valore appropriato.

   I messaggi trigger creati vengono sempre resi disponibili, indipendentemente dal fatto che l'unità di lavoro sia ripristinata o sottoposta a commit.
2. Per le code condivise locali (ossia, le code condivise in un gruppo di condivisione code) il gestore code conta solo i messaggi di cui è stato eseguito il commit.

Ricezione di messaggi da una coda attivata

Quando si progettano le applicazioni che utilizzano il trigger, tenere presente che potrebbe verificarsi un ritardo tra un controllo trigger che avvia un programma e altri messaggi che diventano disponibili sulla

coda dell'applicazione. Ciò può verificarsi quando il messaggio che causa l'evento trigger viene sottoposto a commit prima degli altri.

Per consentire l'arrivo dei messaggi, utilizzare sempre l'opzione di attesa quando si utilizza la chiamata MQGET per rimuovere i messaggi da una coda per cui sono impostate condizioni di trigger. Il *WaitInterval* deve essere sufficiente per consentire il tempo ragionevole più lungo tra l'inserimento di un messaggio e il commit della chiamata di inserimento. Se il messaggio arriva da un gestore code remoto, questa ora è influenzata da:

- Il numero di messaggi immessi prima del commit
- La velocità e la disponibilità del collegamento di comunicazione
- Le dimensioni dei messaggi

Per un esempio di una situazione in cui è necessario utilizzare la chiamata MQGET con l'opzione di attesa, considerare lo stesso esempio utilizzato quando si descrivono le unità di lavoro. Si trattava di una richiesta di inserimento all'interno di un'unità di lavoro per una coda attivata con trigger di tipo FIRST. Questo evento fa sì che il gestore code crei un messaggio trigger. Se si verifica un'altra richiesta di inserimento, da un'altra unità di lavoro, ciò non causa un altro evento trigger poiché il numero di messaggi sulla coda dell'applicazione non è stato modificato da zero a uno. Ora, se viene eseguito il backout della prima unità di lavoro, ma viene eseguito il commit della seconda, viene comunque creato un messaggio di trigger. Quindi, il messaggio di trigger viene creato nel momento in cui viene eseguito il backout della prima unità di lavoro. Se si verifica un ritardo significativo prima che venga eseguito il commit del secondo messaggio, l'applicazione attivata potrebbe dover attendere.

Con l'attivazione di tipo DEPTH, è possibile che si verifichi un ritardo anche se viene eseguito il commit di tutti i messaggi pertinenti. Si supponga che l'attributo della coda *TriggerDepth* abbia il valore 2. Quando due messaggi arrivano sulla coda, il secondo causa la creazione di un messaggio trigger. Tuttavia, se il secondo messaggio è il primo di cui eseguire il commit, è in quel momento che il messaggio del trigger diventa disponibile. Il controllo trigger avvia il programma del server, ma il programma può richiamare solo il secondo messaggio fino a quando non viene eseguito il commit del primo. Quindi il programma potrebbe dover attendere che il primo messaggio sia reso disponibile.

Progettare l'applicazione in modo che termini se nessun messaggio è disponibile per il richiamo quando l'intervallo di attesa scade. Se uno o più messaggi arrivano in un secondo momento, fai affidamento sul fatto che la tua applicazione venga riattivata per elaborarli. Questo metodo impedisce alle applicazioni di essere inattive e di utilizzare inutilmente le risorse.

Elaborazione della coda di iniziazione da parte dei controlli trigger

Per un gestore code, un controllo trigger è come qualsiasi altra applicazione che serve una coda. Tuttavia, un controllo trigger serve le code di iniziazione.

Un controllo trigger è di solito un programma in esecuzione continua. Quando un messaggio trigger arriva su una coda di iniziazione, il controllo trigger richiama tale messaggio. Utilizza le informazioni nel messaggio per emettere un comando per avviare l'applicazione che deve elaborare i messaggi sulla coda dell'applicazione.

Il controllo trigger deve passare informazioni sufficienti al programma che sta avviando in modo che il programma possa eseguire le azioni corrette sulla corretta coda dell'applicazione.

Un iniziatore di canali è un esempio di un tipo speciale di controllo trigger per gli agent del canale dei messaggi. In questa situazione, tuttavia, è necessario utilizzare il tipo di trigger FIRST o DEPTH.

Controlli dei trigger su sistemi UNIX e Windows

Questo argomento contiene informazioni sui controlli dei trigger forniti su sistemi UNIX e Windows .

I seguenti controlli trigger vengono forniti per l'ambiente del server:

amqstrg0
Questo è un controllo trigger di esempio che fornisce un sottoinsieme della funzione fornita da **runmqtrm**. Per ulteriori informazioni su amqstrg0, consultare "Programmi di esempio per piattaforme distribuite" a pagina 97 .

runmqtrm
La sintassi di questo comando è **runmqtrm** [*-m QMgrName*] [*-q InitQ*], dove QMgrName è il gestore code e InitQ è la coda di iniziazione. La coda predefinita è SYSTEM.DEFAULT.INITIATION.QUEUE sul gestore code predefinito. Richiama i programmi per i messaggi trigger appropriati. Questo controllo trigger supporta il tipo di applicazione predefinito.

La stringa di comando trasmessa dal controllo trigger al sistema operativo viene creata come segue:

1. Il *ApplId* dalla definizione PROCESS pertinente (se creato)
2. La struttura MQTMC2 , racchiusa tra doppi apici
3. Il *EnvData* dalla definizione PROCESS pertinente (se creato)

dove *ApplId* è il nome del programma da eseguire come verrebbe immesso sulla riga comandi.

Il parametro passato è la struttura di caratteri MQTMC2 . Viene richiamata una stringa di comando che contiene questa stringa, esattamente come fornita, tra virgolette, in modo che il comando di sistema la accetti come un parametro.

Il controllo dei trigger non controlla se è presente un altro messaggio sulla coda di iniziazione fino al completamento dell'applicazione appena avviata. Se l'applicazione ha molta elaborazione da fare, il controllo trigger potrebbe non essere in grado di tenere il passo con il numero di messaggi trigger in arrivo. Hai due opzioni:

- Avere più controlli trigger in esecuzione
- Eseguire le applicazioni avviate in background

Se si dispone di più controlli dei trigger in esecuzione, è possibile controllare il numero massimo di applicazioni che possono essere eseguite contemporaneamente. Se si eseguono applicazioni in background, non vi è alcuna limitazione imposta da WebSphere MQ sul numero di applicazioni che possono essere eseguite.

Per eseguire l'applicazione avviata in background su sistemi Windows , all'interno del campo *ApplId* , aggiungere un comando START al nome dell'applicazione. Ad esempio:

```
START ?B AMQSECHA
```

Per eseguire l'applicazione avviata in background su sistemi UNIX , inserire una & alla fine della *EnvData* definizione PROCESS.

Nota: Quando un percorso Windows contiene spazi come parte del nome percorso, questi devono essere racchiusi tra virgolette (") per assicurarsi che sia gestito come un singolo argomento. Ad esempio, `" C:\Program Files\Application Directory\Application.exe"`.

Di seguito viene riportato un esempio di stringa APPLICID in cui il nome file include spazi come parte del percorso:

```
START "" /B "C:\Program Files\Application Directory\Application.exe"
```

La sintassi del comando Windows START nell'esempio include una stringa vuota racchiusa tra virgolette doppie. START specifica che il primo argomento tra virgolette verrà considerato come titolo del nuovo comando. Per assicurarsi che Windows non sbagli il percorso dell'applicazione per un argomento 'title', aggiungere una stringa del titolo racchiusa tra virgolette al comando prima del nome dell'applicazione.

I seguenti controlli trigger vengono forniti per il client WebSphere MQ :

runmqtmc
È uguale a runmqtrm tranne che si collega alle librerie client MQI WebSphere MQ .

Per CICS
Il controllo trigger amqltmc0 viene fornito per CICS. Funziona allo stesso modo del controllo trigger standard, runmqtrm, ma viene eseguito in un modo diverso e vengono attivate transazioni CICS .

Questo argomento si applica solo a sistemi Windows, UNIXe Linux .

Viene fornito come programma di CICS ; definirlo con un nome transazione di 4 caratteri. Immettere il nome di 4 caratteri per avviare il controllo trigger. Utilizza il gestore code predefinito (come indicato nel file qm.ini o, su WebSphere MQ per Windows, il registro) e SYSTEM.CICS.INITIATION.QUEUE.

Se si desidera utilizzare un gestore code o una coda differenti, creare la struttura del controllo dei trigger MQTMC2 : ciò richiede la scrittura di un programma utilizzando la chiamata EXEC CICS START, poiché la struttura è troppo lunga per essere aggiunta come parametro. Quindi, passare la struttura MQTMC2 come dati alla richiesta START per il controllo trigger.

Quando si utilizza la struttura MQTMC2 , è necessario fornire solo i parametri *StrucId*, *Version*, *QName*e *QMgrName* al controllo trigger poiché non fa riferimento ad altri campi.

I messaggi vengono letti dalla coda di iniziazione e utilizzati per avviare le transazioni CICS , utilizzando EXEC CICS START, presumendo che APPL_TYPE nel messaggio trigger sia MQAT_CICS. La lettura dei messaggi dalla coda di iniziazione viene eseguita sotto il controllo del punto di sincronizzazione CICS .

I messaggi vengono generati quando il monitoraggio viene avviato e arrestato e quando si verificano errori. Questi messaggi vengono inviati alla coda di dati temporanei CSMT.

Di seguito sono riportate le versioni disponibili del controllo trigger:

Versione	**Utilizza**
amqltmc0	TXSeries per AIX, HP-UXe Sun Solaris versione 5.1
amqltmc4	TXSeries per Windows, Versione 5.1
amqltmcc	Versione di collegamento client del controllo trigger CICS

Se è necessario un controllo trigger per altri ambienti, scrivere un programma che possa elaborare i messaggi trigger che il gestore code inserisce nelle code di iniziazione. Tale programma deve eseguire le azioni riportate di seguito:

1. Utilizzare la chiamata MQGET per attendere l'arrivo di un messaggio sulla coda di avvio.
2. Esaminare i campi della struttura di MQTM del messaggio trigger per individuare il nome dell'applicazione da avviare e l'ambiente in cui viene eseguito.
3. Immettere un comando di avvio specifico dell'ambiente.
4. Convertire la struttura MQTM nella struttura MQTMC2 , se necessario.
5. Passare la struttura MQTMC2 o MQTM all'applicazione avviata. Può contenere dati utente.
6. Associare alla coda dell'applicazione l'applicazione che deve servire tale coda. A tal fine, denominare l'oggetto di definizione del processo (se creato) nell'attributo *ProcessName* della coda.

Per ulteriori informazioni sull'interfaccia di controllo del trigger, consultare MQTMC2.

Proprietà dei messaggi trigger

I seguenti argomenti descrivono alcune altre proprietà dei messaggi trigger.

- “Persistenza e priorità dei messaggi trigger” a pagina 348
- “Messaggi di trigger e riavvio del gestore code” a pagina 348
- “Attiva messaggi e modifiche agli attributi dell'oggetto” a pagina 348
- “Formato dei messaggi di trigger” a pagina 348

Persistenza e priorità dei messaggi trigger

I messaggi di trigger non sono persistenti perché non è necessario che lo siano.

Tuttavia, le condizioni per la generazione di eventi di attivazione persistono, quindi i messaggi di trigger vengono generati ogni volta che queste condizioni vengono soddisfatte. Se un messaggio di trigger viene perso, l'esistenza continua del messaggio dell'applicazione sulla coda dell'applicazione garantisce che il gestore code genera un messaggio di trigger non appena vengono soddisfatte tutte le condizioni.

Se viene eseguito il rollback di un'unità di lavoro, tutti i messaggi di trigger generati vengono sempre consegnati.

I messaggi trigger assumono la priorità predefinita della coda di avvio.

Messaggi di trigger e riavvio del gestore code

In seguito al riavvio di un gestore code, quando una coda di iniziazione viene successivamente aperta per l'input, è possibile inserire un messaggio trigger in questa coda di iniziazione se una coda di applicazione ad essa associata contiene messaggi ed è definita per l'attivazione.

Attiva messaggi e modifiche agli attributi dell'oggetto

I messaggi trigger vengono creati in base ai valori degli attributi trigger in vigore al momento dell'evento trigger.

Se il messaggio trigger non viene reso disponibile per un controllo trigger fino a un momento successivo (poiché il messaggio che ne ha causato la creazione è stato inserito all'interno di un'unità di lavoro), qualsiasi modifica agli attributi trigger nel frattempo non ha alcun effetto sul messaggio trigger. In particolare, la disabilitazione del trigger non impedisce che un messaggio trigger venga reso disponibile una volta creato. Inoltre, la coda dell'applicazione potrebbe non esistere più nel momento in cui il messaggio trigger viene reso disponibile.

Formato dei messaggi di trigger

Il formato di un messaggio trigger è definito dalla struttura MQTM.

Sono presenti i seguenti campi, che il gestore code compila quando crea il messaggio trigger, utilizzando le informazioni nelle definizioni di oggetto della coda dell'applicazione e del processo associato a tale coda:

StrucId
L'identificativo della struttura.

Version
La versione della struttura.

QName
Il nome della coda dell'applicazione in cui si è verificato l'evento trigger. Quando il gestore code crea un messaggio trigger, riempie questo campo utilizzando l'attributo *QName* della coda dell'applicazione.

ProcessName
Il nome dell'oggetto di definizione processo associato alla coda dell'applicazione. Quando il gestore code crea un messaggio trigger, riempie questo campo utilizzando l'attributo *ProcessName* della coda dell'applicazione.

TriggerData
Un campo a formato libero per l'utilizzo da parte del controllo trigger. Quando il gestore code crea un messaggio trigger, riempie questo campo utilizzando l'attributo *TriggerData* della coda dell'applicazione. Su qualsiasi prodotto WebSphere MQ ad eccezione di WebSphere MQ per z/OS, questo campo può essere utilizzato per specificare il nome del canale da attivare.

ApplType
Il tipo di applicazione che il controllo trigger deve avviare. Quando il gestore code crea un messaggio trigger, riempie questo campo utilizzando l'attributo *ApplType* dell'oggetto definizione processo identificato in *ProcessName*.

ApplId
Una stringa di caratteri che identifica l'applicazione che il controllo trigger deve avviare. Quando il gestore code crea un messaggio trigger, riempie questo campo utilizzando l'attributo *ApplId* dell'oggetto definizione processo identificato in *ProcessName*. Quando si utilizza il controllo trigger CKTI o CSQQTRMN fornito da WebSphere MQ per z/OS, l'attributo *ApplId* dell'oggetto di definizione processo è un identificativo di transazione CICS o IMS .

EnvData
Un campo di caratteri che contiene i dati relativi all'ambiente per l'utilizzo da parte del controllo trigger. Quando il gestore code crea un messaggio trigger, riempie questo campo utilizzando l'attributo *EnvData* dell'oggetto definizione processo identificato in *ProcessName*. I controlli trigger forniti da WebSphere MQ for z/OS(CKTI o CSQQTRMN) non utilizzano questo campo, ma altri controlli trigger potrebbero scegliere di utilizzarlo.

UserData
Un campo di caratteri contenente i dati utente che possono essere utilizzati dal controllo trigger. Quando il gestore code crea un messaggio trigger, riempie questo campo utilizzando l'attributo *UserData* dell'oggetto definizione processo identificato in *ProcessName*. Questo campo può essere utilizzato per specificare il nome del canale da attivare.

È disponibile una descrizione completa della struttura del messaggio trigger in MQTM.

Quando l'attivazione non funziona

Un programma non viene attivato se il controllo trigger non può avviare il programma o il gestore code non può consegnare il messaggio trigger. Ad esempio, l'applid nell'oggetto processo deve specificare che il programma deve essere avviato in background; altrimenti, il controllo trigger non può avviare il programma.

Se un messaggio trigger viene creato ma non può essere inserito nella coda di iniziazione (ad esempio, perché la coda è piena o la lunghezza del messaggio trigger è maggiore della lunghezza massima del messaggio specificata per la coda di iniziazione), il messaggio trigger viene inserito nella coda di messaggi non recapitabili (messaggio non consegnato).

Se l'operazione di inserimento nella coda di messaggi non recapitabili non può essere completata correttamente, il messaggio di trigger viene eliminato e viene inviato un messaggio di avvertenza alla console z/OS o all'operatore di sistema oppure viene inserito nel log degli errori.

L'inserimento del messaggio trigger nella coda di messaggi non instradabili potrebbe generare un messaggio trigger per tale coda. Questo secondo messaggio di trigger viene eliminato se aggiunge un messaggio alla coda di messaggi non recapitabili.

Se il programma viene attivato correttamente ma si interrompe prima di ricevere il messaggio dalla coda, utilizzare un programma di utilità di traccia (ad esempio, CICS AUXTRACE se il programma è in esecuzione in CICS) per individuare la causa dell'errore.

Utilizzo di MQI e cluster

Esistono opzioni speciali sulle chiamate e sui codici di ritorno relativi al cluster.

Utilizzare i seguenti link per ulteriori informazioni sulle opzioni disponibili sulle chiamate e sui codici di ritorno da utilizzare con i cluster:

- “MQOPEN e cluster” a pagina 350
- “MQPUT, MQPUT1 e cluster” a pagina 351
- “MQINQ e cluster” a pagina 352
- “MQSET e cluster” a pagina 352

- “Codici di ritorno” a pagina 352

Concetti correlati

“Panoramica su Message Queue Interface” a pagina 195
Informazioni sui componenti MQI (Message Queue Interface).

“Connessione e disconnessione da un gestore code” a pagina 207
Per utilizzare i servizi di programmazione WebSphere MQ , un programma deve disporre di una connessione a un gestore code. Utilizzare queste informazioni per informazioni su come connettersi e disconnettersi da un gestore code.

“Apertura e chiusura di oggetti” a pagina 215
Queste informazioni forniscono informazioni sull'apertura e la chiusura degli oggetti WebSphere MQ .

“Inserimento di messaggi in una coda” a pagina 225
Utilizzare queste informazioni per informazioni su come inserire i messaggi su una coda.

“Richiamo dei messaggi da una coda” a pagina 241
Utilizzare queste informazioni per ottenere messaggi da una coda.

“Richiesta di informazioni e impostazione degli attributi dell'oggetto” a pagina 322
Gli attributi sono proprietà che definiscono le caratteristiche di un oggetto WebSphere MQ .

“Commit e backout delle unità di lavoro” a pagina 325
Queste informazioni descrivono come eseguire il commit e il backout di tutte le operazioni get e put recuperabili che si sono verificate in un'unità di lavoro.

“Avvio delle applicazioni IBM WebSphere MQ utilizzando i trigger” a pagina 331
Informazioni sui trigger e su come avviare le applicazioni IBM WebSphere MQ utilizzando i trigger.

MQOPEN e cluster

La coda in cui viene inserito un messaggio o da cui viene letto, quando viene aperta una coda cluster, dipende dalla chiamata MQOPEN .

Selezione della coda di destinazione

Se non si fornisce un nome gestore code nel descrittore dell'oggetto, MQOD, il gestore code seleziona il gestore code a cui inviare il messaggio. Se si fornisce un nome gestore code nel descrittore dell'oggetto, i messaggi vengono sempre inviati al gestore code selezionato.

Se il gestore code sta selezionando il gestore code di destinazione, la selezione dipende dalle opzioni di bind MQOO_BIND_* e se esiste una coda locale. Se è presente un'istanza locale della coda, questa viene sempre aperta in preferenza a un'istanza remota, a meno che l'attributo CLWLUSEQ non sia impostato su ANY. Altrimenti, la selezione dipende dalle opzioni di collegamento. È necessario specificare MQOO_BIND_ON_OPEN o MQOO_BIND_ON_GROUP quando si utilizzano gruppi di messaggi con cluster per garantire che tutti i messaggi nel gruppo vengano elaborati alla stessa destinazione.

Se il gestore code sta selezionando il gestore code di destinazione, lo fa in modo round - robin, utilizzando l'algoritmo di gestione del carico di lavoro; consultare Bilanciamento del carico di lavoro.

MQOO_BIND_ON_OPEN
L'opzione MQOO_BIND_ON_OPEN sulla chiamata MQOPEN specifica che il gestore code di destinazione deve essere corretto. Utilizzare l'opzione MQOO_BIND_ON_OPEN se vi sono più istanze della stessa coda all'interno di un cluster. Tutti i messaggi immessi nella coda specificando l'handle dell'oggetto restituito dalla chiamata MQOPEN vengono indirizzati allo stesso gestore code.

- Utilizzare l'opzione MQOO_BIND_ON_OPEN se i messaggi hanno affinità. Ad esempio, se un batch di messaggi deve essere tutti elaborato dallo stesso gestore code, specificare MQOO_BIND_ON_OPEN quando si apre la coda. IBM WebSphere MQ corregge il gestore code e l'instradamento che deve essere utilizzato da tutti i messaggi inseriti in tale coda.
- Se viene specificata l'opzione MQOO_BIND_ON_OPEN , la coda deve essere riaperta per poter selezionare una nuova istanza della coda.

MQOO_BIND_NOT_FIXED
L'opzione MQOO_BIND_NOT_FIXED nella chiamata MQOPEN specifica che il gestore code di destinazione non è fisso. I messaggi scritti nella coda che specificano l'handle dell'oggetto restituito dalla chiamata MQOPEN vengono instradati a un gestore code MQPUT in base al messaggio. Utilizzare l'opzione MQOO_BIND_NOT_FIXED se non si desidera forzare la scrittura di tutti i messaggi nella stessa destinazione.

- Non specificare MQOO_BIND_NOT_FIXED e MQMF_SEGMENTATION_ALLOWED contemporaneamente. In tal caso, i segmenti del messaggio potrebbero essere consegnati a gestori code differenti, sparsi in tutto il cluster.

MQOO_BIND_ON_GROUP
Consente a una applicazione di richiedere che un gruppo di messaggi sia assegnato alla stessa istanza di destinazione. Questa opzione è valida solo per le code e interessa solo le code cluster. Se specificata per una coda che non è una coda cluster, l'opzione viene ignorata.

- I gruppi vengono instradati solo a una destinazione singola quando MQPMO_LOGICAL_ORDER viene specificato su MQPUT. Quando viene specificato MQOO_BIND_ON_GROUP, ma un messaggio non fa parte di un gruppo, viene invece utilizzato il comportamento BIND_NOT_FIXED.

MQOO_BIND_AS_Q_DEF
Se non si specifica MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED o MQOO_BIND_ON_GROUP, l'opzione predefinita è MQOO_BIND_AS_Q_DEF. L'utilizzo di MQOO_BIND_AS_Q_DEF fa in modo che il bind utilizzato per l'handle della coda venga preso dall'attributo della coda DefBind .

Pertinenza delle opzioni MQOPEN

Le MQOPEN opzioni MQOO_BROWSE, MQOO_INPUT_*o MQOO_SET richiedono un'istanza locale della coda del cluster perché MQOPEN abbia esito positivo.

Le MQOPEN opzioni MQOO_OUTPUT, MQOO_BIND_*o MQOO_INQUIRE non richiedono un'istanza locale della coda del cluster per avere esito positivo.

Nome del gestore code risolto

Quando un nome gestore code viene risolto in MQOPEN , il nome risolto viene restituito all'applicazione. Se l'applicazione tenta di utilizzare questo nome in una chiamata MQOPEN successiva, potrebbe scoprire che non è autorizzata ad accedere al nome.

MQPUT, MQPUT1 e cluster

Se MQOO_BIND_NOT_FIXED è specificato su MQOPEN le routine di gestione del carico di lavoro scelgono quale destinazione MQPUT o MQPUT1 selezionare.

Se MQOO_BIND_NOT_FIXED viene specificato su una chiamata MQOPEN , ogni chiamata MQPUT successiva richiama la routine di gestione del carico di lavoro per stabilire a quale gestore code inviare il messaggio. La destinazione e l'instradamento da prendere vengono selezionati messaggio per messaggio. La destinazione e l'instradamento potrebbero cambiare dopo che il messaggio è stato inserito se cambiano le condizioni nella rete. La chiamata MQPUT1 funziona sempre come se MQOO_BIND_NOT_FIXED fosse attiva, ossia richiama sempre la routine di gestione del carico di lavoro.

Quando la routine di gestione del carico di lavoro ha selezionato un gestore code, il gestore code locale completa l'operazione di inserimento. Il messaggio può essere inserito in code differenti:

1. Se la destinazione è l'istanza locale della coda, il messaggio viene inserito nella coda locale.
2. Se la destinazione è un gestore code in un cluster, il messaggio viene inserito in una coda di trasmissione cluster.
3. Se la destinazione è un gestore code esterno a un cluster, il messaggio viene inserito in una coda di trasmissione con lo stesso nome del gestore code di destinazione.

Se MQOO_BIND_ON_OPEN è specificato nella chiamata MQOPEN , le chiamate MQPUT non richiamano la routine di gestione del workload perché la destinazione e l'instradamento sono già stati selezionati.

MQINQ e cluster

La coda cluster da interrogare dipende dalle opzioni che si combinano con MQOO_INQUIRE.

Prima di poter analizzare una coda, aprirla utilizzando la chiamata MQOPEN e specificare MQOO_INQUIRE.

Per analizzare una coda cluster, utilizzare la chiamata MQOPEN e combinare altre opzioni con MQOO_INQUIRE. Gli attributi che possono essere interrogati dipendono dalla presenza di un'istanza locale della coda del cluster e dalla modalità di apertura della coda:

- La combinazione di MQOO_BROWSE, MQOO_INPUT_*o MQOO_SET con MQOO_INQUIRE richiede un'istanza locale della coda del cluster perché l'apertura abbia esito positivo. In questo caso, è possibile esaminare tutti gli attributi validi per le code locali.
- Combinando MQOO_OUTPUT con MQOO_INQUIREe non specificando alcuna delle opzioni precedenti, l'istanza aperta è:
  - L'istanza sul gestore code locale, se presente. In questo caso, è possibile esaminare tutti gli attributi validi per le code locali.
  - Un'istanza altrove nel cluster, se non è presente alcuna istanza del gestore code locale. In questo caso è possibile interrogare solo i seguenti attributi. In questo caso l'attributo QType ha il valore MQQT_CLUSTER .
    - DefBind
    - DefPersistence
    - DefPriority
    - InhibitPut
    - QDesc
    - QName
    - QTYPE

Per analizzare l'attributo DefBind di una coda cluster, utilizzare la chiamata MQINQ con il selettore MQIA_DEF_BIND. Il valore restituito è MQBND_BIND_ON_OPEN o MQBND_BIND_NOT_FIXEDo MQBND_BIND_ON_GROUP. È necessario specificare MQBND_BIND_ON_OPEN o MQBND_BIND_ON_GROUP quando si utilizzano gruppi con cluster.

Per analizzare gli attributi CLUSTER e CLUSNL dell'istanza locale di una coda, utilizzare la chiamata MQINQ con il selettore MQCA_CLUSTER_NAME o il selettore MQCA_CLUSTER_NAMELIST.

Nota: Se si apre una coda cluster senza correggere la coda a cui MQOPEN ha eseguito il bind, le successive chiamate MQINQ potrebbero interrogare diverse istanze della coda cluster.

Concetti correlati

“Opzione MQOPEN per la coda cluster” a pagina 221
Il bind utilizzato per l'handle della coda viene preso dall'attributo della coda *DefBind* , che può assumere il valore MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXEDo MQBND_BIND_ON_GROUP.

MQSET e cluster

L'opzione MQOPEN option MQOO_SET richiede che ci sia un'istanza locale di una coda cluster perché MQSET abbia esito positivo.

Non è possibile utilizzare la chiamata MQSET per impostare gli attributi di una coda altrove nel cluster.

È possibile aprire un alias locale o una coda remota definita con l'attributo cluster e utilizzare la chiamata MQSET . È possibile impostare gli attributi dell'alias locale o della coda remota. Non importa se la coda di destinazione è una coda cluster definita su un gestore code differente.

Codici di ritorno

Codici di ritorno specifici per i cluster

MQRC_CLUSTER_EXIT_ERROR (2266 X'8DA')

Viene emessa una chiamata MQOPEN, MQPUTo MQPUT1 per aprire una coda cluster o inserire un messaggio su di essa. L'uscita del carico di lavoro del cluster, definita dall'attributo ClusterWorkloadExit di un gestore code, ha esito negativo in modo imprevisto o non risponde in tempo.

Un messaggio viene scritto nel log di sistema su WebSphere MQ per z/OS fornendo ulteriori informazioni su questo errore.

Le successive chiamate MQOPEN, MQPUTe MQPUT1 per questo handle di coda vengono elaborate come se l'attributo ClusterWorkloadExit fosse vuoto.

MQRC_CLUSTER_EXIT_LOAD_ERROR (2267 X'8DB')

Su z/OS, non è possibile caricare l'exit del carico di lavoro del cluster.

Un messaggio viene scritto nel log di sistema e l'elaborazione continua come se l'attributo ClusterWorkloadExit fosse vuoto.

Su piattaforme diverse da z/OS, viene emessa una chiamata MQCONN o MQCONNX per connettersi a un gestore code. La chiamata ha esito negativo perché non è possibile caricare l'uscita del carico di lavoro cluster, definita dall'attributo ClusterWorkloadExit del gestore code.

MQRC_CLUSTER_PUT_INHIBITED (2268 X'8DC')

Viene emessa una chiamata MQOPEN con le opzioni MQOO_OUTPUT e MQOO_BIND_ON_OPEN attive per una coda cluster. Tutte le istanze della coda nel cluster sono attualmente inibite dall'inserimento, poiché l'attributo InhibitPut è impostato su MQQA_PUT_INHIBITED. Poiché non ci sono istanze di coda disponibili per ricevere messaggi, la chiamata MQOPEN ha esito negativo.

Questo codice di errore si verifica solo quando si verificano entrambe le seguenti condizioni:

- Non esiste alcuna istanza locale della coda. Se è presente un'istanza locale, la chiamata MQOPEN ha esito positivo, anche se l'istanza locale è di tipo put - inibito.
- Non esiste alcuna uscita del carico di lavoro del cluster per la coda oppure esiste un'uscita del carico di lavoro del cluster ma non sceglie un'istanza della coda. (Se l'uscita del carico di lavoro del cluster sceglie un'istanza della coda, la chiamata MQOPEN ha esito positivo, anche se tale istanza è inibita dall'inserimento.)

Se l'opzione MQOO_BIND_NOT_FIXED viene specificata nella chiamata MQOPEN , la chiamata può avere esito positivo anche se tutte le code nel cluster sono bloccate dall'inserimento. Tuttavia, una chiamata MQPUT successiva potrebbe non riuscire se tutte le code sono ancora immesse al momento di tale chiamata.

MQRC_CLUSTER_RESOLUTION_ERROR (2189 X'88D')

1. Viene emessa una chiamata MQOPEN, MQPUTo MQPUT1 per aprire una coda cluster o inserire un messaggio su di essa. La definizione della coda non può essere risolta correttamente perché è richiesta una risposta dal gestore code del repository completo, ma non ne è disponibile alcuna.
2. Viene emessa una chiamata MQOPEN, MQPUT, MQPUT1 o MQSUB per un oggetto argomento che specifica PUBSCOPE(ALL) o SUBSCOPE(ALL). La definizione dell'argomento del cluster non può essere risolta correttamente perché è richiesta una risposta dal gestore code del repository completo, ma non è disponibile.

MQRC_CLUSTER_RESOURCE_ERROR (2269 X'8DD')

Viene emessa una chiamata MQOPEN, MQPUTo MQPUT1 per una coda cluster. Si è verificato un errore durante il tentativo di utilizzare una risorsa richiesta per il clustering.

MQRC_NO_DESTINATIONS_AVAILABLE (2270 X'8DE')

Viene emessa una chiamata MQPUT o MQPUT1 per inserire un messaggio in una coda cluster. Al momento della chiamata, non ci sono più istanze della coda nel cluster. MQPUT ha esito negativo e il messaggio non viene inviato.

L'errore può verificarsi se MQOO_BIND_NOT_FIXED viene specificata sulla chiamata MQOPEN che apre la coda oppure se viene utilizzato MQPUT1 per inserire il messaggio.

MQRC_STOPPED_BY_CLUSTER_EXIT (2188 X'88C')

Viene emessa una chiamata MQOPEN, MQPUTo MQPUT1 per aprire o inserire un messaggio in una coda cluster. L'uscita del carico di lavoro del cluster rifiuta la chiamata.

Scrittura delle applicazioni client

Informazioni necessarie per scrivere le applicazioni client su WebSphere MQ.

Le applicazioni possono essere create ed eseguite nell'ambiente client WebSphere MQ . L'applicazione deve essere creata e collegata al client MQI WebSphere MQ utilizzato. Il modo in cui le applicazioni vengono create e collegate varia in base alla piattaforma e al linguaggio di programmazione utilizzato. Per informazioni su come creare applicazioni client, consultare “Creazione di applicazioni per client MQI WebSphere MQ” a pagina 360.

È possibile eseguire un'applicazione WebSphere MQ in un ambiente WebSphere MQ completo e in un ambiente client WebSphere MQ MQI senza modificare il proprio codice, a condizione che vengano soddisfatte determinate condizioni. Per ulteriori informazioni sull'esecuzione delle proprie applicazioni nell'ambiente client WebSphere MQ , consultare “Esecuzione di applicazioni nell'ambiente client IBM WebSphere MQ MQI” a pagina 362.

Se si utilizza l'interfaccia MQI (message queue interface) per scrivere applicazioni da eseguire in un ambiente client WebSphere MQ MQI, esistono alcuni controlli aggiuntivi da imporre durante una chiamata MQI per garantire che l'elaborazione dell'applicazione WebSphere MQ non venga interrotta. Per ulteriori informazioni su questi controlli, consultare “Utilizzo dell'interfaccia della coda messaggi (MQI) in un'applicazione client” a pagina 355.

Consultare i seguenti argomenti per informazioni sulla preparazione ed esecuzione di altri tipi di applicazione come applicazioni client:

- “Preparazione ed esecuzione di applicazioni CICS e Tuxedo” a pagina 373
- “Preparazione ed esecuzione di applicazioni Microsoft Transaction Server” a pagina 41
- “Preparazione ed esecuzione delle applicazioni WebSphere MQ JMS” a pagina 376

Concetti correlati

“Concetti dello sviluppo di applicazioni” a pagina 8
È possibile utilizzare una scelta di linguaggi procedurali o orientati agli oggetti per scrivere applicazioni IBM WebSphere MQ . Utilizzare i link in questo argomento per informazioni sui concetti di IBM WebSphere MQ che sono utili per sviluppatori di applicazioni.

“Scelta del linguaggio di programmazione da utilizzare” a pagina 79
Utilizza queste informazioni per scoprire i linguaggi di programmazione e i framework supportati da IBM WebSphere MQe alcune considerazioni per utilizzarli.

“Progettazione di applicazioni IBM WebSphere MQ” a pagina 90
Una volta stabilito il modo in cui le applicazioni possono sfruttare le piattaforme e gli ambienti disponibili, è necessario decidere come utilizzare le funzioni offerte da WebSphere MQ.

“Programmi di WebSphere MQ di esempio” a pagina 97
Utilizzare questa raccolta di argomenti per informazioni sui programmi WebSphere MQ di esempio su piattaforme differenti.

“Scrittura di un'applicazione di accodamento” a pagina 195
Utilizzare queste informazioni per informazioni sulla scrittura delle applicazioni di accodamento, la connessione e la disconnessione da un gestore code, la pubblicazione / sottoscrizione e l'apertura e la chiusura di oggetti.

“Utilizzo dei servizi Web in WebSphere MQ” a pagina 948
È possibile sviluppare applicazioni IBM WebSphere MQ per servizi Web utilizzando il trasporto IBM WebSphere MQ per SOAP o il bridge IBM WebSphere MQ per HTTP.

“Scrittura di applicazioni di pubblicazione / sottoscrizione” a pagina 279
Iniziare a scrivere applicazioni WebSphere MQ di pubblicazione / sottoscrizione.

“Creazione di un'applicazione IBM WebSphere MQ” a pagina 431
Utilizzare queste informazioni per informazioni sulla generazione di un'applicazione IBM WebSphere MQ su piattaforme differenti.

“Gestione degli errori del programma” a pagina 550
Queste informazioni spiegano gli errori associati alle chiamate MQI delle applicazioni quando effettua una chiamata o quando il suo messaggio viene consegnato alla destinazione finale.

Utilizzo dell'interfaccia della coda messaggi (MQI) in un'applicazione client

In questa raccolta di argomenti vengono considerate le differenze tra la scrittura dell'applicazione WebSphere MQ da eseguire in un ambiente client WebSphere MQ MQI e da eseguire nell'ambiente gestore code WebSphere MQ completo.

Quando si progetta un'applicazione, considerare quali controlli è necessario imporre durante una chiamata MQI per assicurarsi che l'elaborazione dell'applicazione WebSphere MQ non venga interrotta.

Limitazione della dimensione di un messaggio in una applicazione client

Un gestore code ha una lunghezza massima del messaggio, ma la dimensione massima del messaggio che è possibile trasmettere da una applicazione client è limitata dalla definizione del canale.

L'attributo lunghezza massima del messaggio (MaxMsgLength) di un gestore code è la lunghezza massima di un messaggio che può essere gestito da tale gestore code.

Su piattaforme diverse da z/OS, è possibile incrementare l'attributo della lunghezza massima del messaggio di un gestore code. I dettagli vengono forniti in ALTER QMGR.

È possibile rilevare il valore di MaxMsgLength per un gestore code utilizzando la chiamata MQINQ.

Se l'attributo di lunghezza MaxMsgviene modificato, non viene effettuato alcun controllo che non vi siano già code e nemmeno messaggi con una lunghezza superiore al nuovo valore. Dopo aver modificato questo attributo, riavviare le applicazioni e i canali per essere certi che la modifica sia diventata effettiva. Non è quindi possibile generare nuovi messaggi che superano la lunghezza MaxMsgdel gestore code o della coda (a meno che non sia consentita la segmentazione del gestore code).

La lunghezza massima del messaggio in una definizione del canale limita la dimensione di un messaggio che è possibile trasmettere lungo una connessione client. Se un'applicazione WebSphere MQ tenta di utilizzare la chiamata MQPUT o la chiamata MQGET con un messaggio più grande di questo, viene restituito un codice di errore all'applicazione. Il parametro della dimensione massima del messaggio della definizione del canale non influenza la dimensione massima del messaggio che può essere utilizzata utilizzando MQBC su una connessione client.

Scelta del CCSID (coded character set identifier) client o server

Utilizzare il CCSID locale per il client. Il gestore code esegue la conversione necessaria. Utilizzare la variabile di ambiente MQCCSID per sovrascrivere il CCSID. Se l'applicazione esegue più PUT, i campi CCSID e di codifica di MQMD possono essere sovrascritti dopo il completamento del primo PUT.

I dati passati attraverso MQI dall'applicazione allo stub client devono essere nel CCSID locale, codificati per il client WebSphere MQ . Se il gestore code connesso richiede la conversione dei dati, la conversione viene eseguita dal codice di supporto client sul gestore code.

Il client Java in V7, tuttavia, può eseguire la conversione se il gestore code non è in grado di farlo. Vedi “Classi WebSphere MQ per connessioni client Java” a pagina 670

Il codice client presuppone che i dati carattere che attraversano l'MQI nel client si trovano nel CCSID configurato per quella workstation. Se questo CCSID è un CCSID non supportato o non è il CCSID richiesto, può essere sovrascritto con la variabile di ambiente MQCCSID utilizzando uno dei seguenti comandi:

- Su Windows:

```
SET MQCCSID=850
```

- Su sistemi UNIX:

```
export MQCCSID=850
```

Se questo parametro è impostato nel profilo, si presuppone che tutti i dati MQI si trovano nella codepage 850.

Nota: Il presupposto relativo alla codepage 850 non si applica ai dati dell'applicazione nel messaggio.

Se l'applicazione sta eseguendo più PUT che includono intestazioni WebSphere MQ dopo il descrittore del messaggio (MQMD), tenere presente che i campi CCSID e di codifica di MQMD vengono sovrascritti dopo il completamento del primo PUT.

Dopo il primo PUT, questi campi contengono il valore utilizzato dal gestore code connesso per convertire le intestazioni WebSphere MQ . Verificare che l'applicazione reimposti i valori sui valori richiesti.

Utilizzo di MQINQ in un'applicazione client

Alcuni valori interrogati utilizzando MQINQ vengono modificati dal codice client.

CCSID
è impostato sul CCSID client, non su quello del gestore code.

MaxMsgLunghezza
viene ridotto se è limitato dalla definizione di canale. Questo sarà il valore più basso tra:

- Il valore definito nella definizione della coda oppure
- Il valore definito nella definizione di canale

Per ulteriori informazioni, consultare MQINQ.

Utilizzo del coordinamento del punto di sincronizzazione in un'applicazione client

Un'applicazione in esecuzione su un client di base può emettere MQCMIT e MQBACK, ma l'ambito del controllo del punto di sincronizzazione è limitato alle risorse MQI. È possibile utilizzare un gestore transazioni esterno con un client transazionale esteso.

In WebSphere MQ, uno dei ruoli del gestore code è il controllo del punto di sincronizzazione all'interno di un'applicazione. Se un'applicazione viene eseguita su un client di base WebSphere MQ , può emettere MQCMIT e MQBACK, ma l'ambito del controllo del punto di sincronizzazione è limitato alle risorse MQI. Il comando WebSphere MQ MQBEGIN non è valido in un ambiente client di base.

Le applicazioni in esecuzione nell'ambiente del gestore code completo sul server possono coordinare più risorse (ad esempio i database) tramite un monitoraggio delle transazioni. Sul server è possibile utilizzare il monitoraggio transazioni fornito con i prodotti WebSphere MQ o un altro monitoraggio transazioni, ad esempio CICS. Non è possibile utilizzare un monitoraggio transazioni con un'applicazione client base.

È possibile utilizzare un gestore transazioni esterno con un client transazionale esteso WebSphere MQ . Consultare Cos' è un client transazionale esteso? per ulteriori dettagli.

Utilizzo della lettura anticipata in un'applicazione client

È possibile utilizzare la lettura anticipata su un client per consentire l'invio di messaggi non persistenti a un client senza che l'applicazione client debba richiedere i messaggi.

Quando un client richiede un messaggio da un server, invia una richiesta al server. Invia una richiesta separata per ciascuno dei messaggi che utilizza. Per migliorare le prestazioni di un client che utilizza messaggi non persistenti evitando di dover inviare questi messaggi di richiesta, è possibile configurare

un client per utilizzare la lettura anticipata. La lettura anticipata consente l'invio di messaggi a un cliente senza che un'applicazione debba richiederli.

L'utilizzo della lettura anticipata può migliorare le prestazioni quando si utilizzano messaggi non persistenti da un'applicazione client. Questo miglioramento delle performance è disponibile sia per le applicazioni MQI che JMS. Le applicazioni client che utilizzano MQGET o l'utilizzo asincrono traggono vantaggio dai miglioramenti delle prestazioni quando si utilizzano messaggi non persistenti.

Quando si richiama MQOPEN con MQOO_READ_AHEAD, il client WebSphere MQ abilita la lettura anticipata solo se vengono soddisfatte determinate condizioni. Queste condizioni includono:

- Sia il client che il gestore code remoto devono trovarsi in WebSphere MQ Versione 7 o successive.
- L'applicazione client deve essere compilata e collegata rispetto alle librerie client MQI WebSphere MQ con thread.
- Il canale del client deve utilizzare il protocollo TCP/IP
- Il canale deve avere un'impostazione SharingConversations (SHARECNV) diversa da zero nelle definizioni di canale del client e del server.

Quando la lettura anticipata è abilitata, i messaggi vengono inviati a un buffer di memoria sul client denominato buffer di lettura anticipata. Il client dispone di un buffer di lettura anticipata per ogni coda aperta con lettura anticipata abilitata. I messaggi nel buffer di lettura anticipata non sono persistenti. Il client aggiorna periodicamente il server con informazioni sulla quantità di dati che ha utilizzato.

Non tutte le progettazioni di applicazioni client sono adatte per l'utilizzo della lettura anticipata poiché non tutte le opzioni sono supportate per l'utilizzo. È necessario che alcune opzioni siano congruenti tra le chiamate MQGET quando la lettura anticipata è abilitata. Se un client modifica i criteri di selezione tra le chiamate MQGET, i messaggi memorizzati nel buffer di lettura anticipata rimangono bloccati nel buffer di lettura anticipata del client. Per ulteriori informazioni, vedi “Miglioramento delle prestazioni dei messaggi non persistenti” a pagina 259

La configurazione read ahead è controllata da tre attributi, MaximumSize, PurgeTimee UpdatePercentage, specificati nella stanza MessageBuffer del file di configurazione del client WebSphere MQ .

Utilizzo dell'inserimento asincrono in un'applicazione client

Utilizzando l'inserimento asincrono, un'applicazione può inserire un messaggio in una coda senza attendere una risposta dal gestore code. È possibile utilizzarlo per migliorare le prestazioni della messaggistica in alcune situazioni.

Di solito, quando un'applicazione inserisce uno o più messaggi in una coda, utilizzando MQPUT o MQPUT1, l'applicazione deve attendere che il gestore code confermi di aver elaborato la richiesta MQI. È possibile migliorare le prestazioni della messaggistica, in modo particolare per le applicazioni che utilizzano i bind del client e per le applicazioni che inserono un numero elevato di messaggi di piccole dimensioni in una coda, scegliendo invece di inserire i messaggi in modo asincrono. Quando un'applicazione inserisce un messaggio in maniera asincrona, il gestore code non restituisce l'esito positivo o negativo di ciascuna chiamata, ma è possibile controllare periodicamente la presenza di errori.

Per inserire un messaggio in una coda in modo asincrono, utilizzare l'opzione MQPMO_ASYNC_RESPONSE nel campo *Options* della struttura MQPMO.

Se un messaggio non è idoneo per l'inserimento asincrono, viene inserito in una coda in modo sincrono.

Quando si richiede una risposta di inserimento asincrono per MQPUT o MQPUT1, un CompCode e motivo di MQCC_OK e MQRC_NONE non significa necessariamente che il messaggio è stato inserito correttamente in una coda. Anche se l'esito positivo o negativo di ogni singola chiamata MQPUT o MQPUT1 potrebbe non essere restituito immediatamente, il primo errore che si è verificato in una chiamata asincrona può essere determinato successivamente tramite una chiamata a MQSTAT.

Per ulteriori dettagli su MQPMO_ASYNC_RESPONSE, consultare Opzioni MQPMO.

Il programma di esempio Put asincrono dimostra alcune funzioni disponibili. Per i dettagli sulle caratteristiche e la progettazione del programma e su come eseguirlo, consultare “Il programma di esempio Put asincrono” a pagina 116.

Utilizzo delle conversazioni di condivisione in un'applicazione client

In un ambiente in cui la condivisione delle conversazioni è consentita, le conversazioni possono condividere un'istanza del canale MQI.

La condivisione delle conversazioni è controllata da due campi, entrambi denominati SharingConversations, uno dei quali fa parte della struttura MQCD (channel definition) e uno dei quali fa parte della struttura MQCXP (channel exit parameter). Il campo SharingConversations in MQCD è un valore intero che determina il numero massimo di conversazioni che possono condividere un'istanza del canale associata al canale. Il campo SharingConversations in MQCXP è un valore booleano, che indica se l'istanza del canale è attualmente condivisa.

In un ambiente in cui la condivisione delle conversazioni non è consentita, le nuove connessioni client che specificano MQCD identici non condivideranno un'istanza del canale.

Una nuova connessione dell'applicazione client condividerà l'istanza del canale quando si verificano le seguenti condizioni:

- Le estremità di connessione client e server dell'istanza del canale sono configurate per la condivisione delle conversazioni e questi valori non vengono sovrascritti dalle uscite del canale.
- Il valore MQCD della connessione client (fornito sulla chiamata MQCONNX del client o dalla tabella di definizione del canale client (CCDT)) corrisponde esattamente al valore MQCD della connessione client fornito sulla chiamata MQCONNX del client o dalla CCDT quando è stata stabilita per la prima volta l'istanza del canale esistente. Notare che l'MQCD originale potrebbe essere stato successivamente modificato dalle uscite o dalla negoziazione del canale, ma che la corrispondenza viene effettuata rispetto al valore fornito al sistema client prima che queste modifiche siano state apportate.
- Il limite di conversazioni di condivisione sul lato server non è stato superato.

Se una nuova connessione dell'applicazione client corrisponde ai criteri per eseguire la condivisione di un'istanza del canale con altre conversazioni, questa decisione viene presa prima che vengano richiamate le uscite su tale conversazione. Le uscite su una conversazione di questo tipo non possono alterare il fatto che sta condividendo l'istanza del canale con altre conversazioni. Se non ci sono istanze di canale esistenti che corrispondono alla nuova definizione di canale, viene connessa una nuova istanza di canale.

La negoziazione del canali si verifica solo per la prima conversazione su un'istanza del canale; i valori negoziati per l'istanza del canale sono fissi in quella fase e non possono essere modificati all'avvio delle conversazioni successive. L'autenticazione TLS/SSL si verifica anche per la prima conversazione.

Se il valore di MQCD SharingConversations viene modificato durante l'inizializzazione di qualsiasi uscita di sicurezza, di invio o di ricezione per la prima conversazione sul socket alla connessione client o all'estremità di connessione server dell'istanza del canale, il nuovo valore che ha dopo l'inizializzazione di tutte queste uscite viene utilizzato per determinare il valore delle conversazioni di condivisione per l'istanza del canale (il valore più basso ha la precedenza).

Se il valore negoziato per la condivisione delle conversazioni è zero, l'istanza del canale non viene mai condivisa. Ulteriori programmi di uscita che impostano questo campo su zero vengono eseguiti in modo simile sulla propria istanza del canale.

Se il valore negoziato per la condivisione delle conversazioni è maggiore di zero, MQCXP SharingConversations è impostato su TRUE per le chiamate successive alle uscite, indicando che altri programmi di uscita su questa istanza del canale possono essere immessi contemporaneamente a questo.

Quando si scrive un programma di uscita del canale, considerare se verrà eseguito su un'istanza del canale che potrebbe implicare la condivisione delle conversazioni. Se l'istanza del canale potrebbe implicare la condivisione di conversazioni, considerare l'effetto sulle altre istanze dell'uscita del canale della modifica dei campi MQCD; tutti i campi MQCD hanno valori comuni in tutte le conversazioni di condivisione. Una volta stabilita l'istanza del canale, se i programmi di uscita tentano di modificare i campi MQCD, potrebbero riscontrare dei problemi perché altre istanze di programmi di uscita in esecuzione sull'istanza del canale potrebbero tentare di modificare gli stessi campi contemporaneamente. Se questa situazione può verificarsi con i programmi di uscita, è necessario serializzare l'accesso al MQCD nel codice di uscita.

Se si utilizza un canale definito per condividere le conversazioni, ma non si desidera che la condivisione avvenga su una particolare istanza del canale, impostare il valore MQCD di SharingConversations su 1 o 0 quando si inizializza un'uscita del canale sulla prima conversazione sull'istanza del canale. Consultare SharingConversations per una spiegazione dei valori di SharingConversations.

Esempio

La condivisione delle conversazioni è abilitata.

Si sta utilizzando una definizione di canale di collegamento client che specifica un programma di uscita.

La prima volta che questo canale viene avviato, il programma di uscita modifica alcuni parametri MQCD quando viene inizializzato. Questi sono gestiti dal canale, quindi la definizione con cui è in esecuzione il canale è ora differente da quella fornita originariamente. Il parametro MQCXP SharingConversations è impostato su TRUE.

La volta successiva in cui l'applicazione si connette utilizzando questo canale, la conversazione viene eseguita sull'istanza del canale che è stata avviata precedentemente, perché ha la stessa definizione di canale originale. L'istanza del canale a cui l'applicazione si connette la seconda volta è la stessa istanza della prima volta che si connette. Di conseguenza, utilizza le definizioni che sono state modificate dal programma di uscita. Quando il programma di uscita viene inizializzato per la seconda conversazione, sebbene possa modificare i campi MQCD, *non* vengono utilizzati dal canale. Queste stesse caratteristiche si applicano a tutte le conversazioni successive che condividono l'istanza del canale.

Utilizzo di MQCONNX

È possibile utilizzare la chiamata MQCONNX per specificare una struttura MQCD (channel definition) nella struttura MQCNO.

Ciò consente all'applicazione client chiamante di specificare la definizione del canale di connessione client al runtime. Per ulteriori informazioni, consultare Utilizzo della struttura MQCNO su una chiamata MQCONNX. Quando si utilizza MQCONNX, la chiamata emessa sul server dipende dal livello server e dalla configurazione del listener.

Quando si utilizza MQCONNX da un client, le seguenti opzioni vengono ignorate:

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING

La struttura MQCD che è possibile utilizzare dipende dal numero di versione MQCD che si sta utilizzando. Per informazioni sulle versioni MQCD (MQCD_VERSION), consultare Versione MQCD. È possibile utilizzare la struttura MQCD, ad esempio, per passare i programmi di uscita canale al server. Se si utilizza MQCD Versione 3 o successiva, è possibile utilizzare la struttura per passare un array di uscite al server. È possibile utilizzare questa funzione per eseguire più di un'operazione sullo stesso messaggio, come la codifica e la compressione, aggiungendo un'uscita per ciascuna operazione, piuttosto che modificare un'uscita esistente. Se non si specifica un array nella struttura MQCD, verranno controllati i singoli campi di uscita. Per ulteriori informazioni sui programmi di uscita del canale, consultare “Programmi di uscita canale per canali di messaggistica” a pagina 399.

Handle di connessione condivisi su MQCONNX

È possibile condividere handle tra thread differenti all'interno dello stesso processo, utilizzando handle di connessione condivisi.

Quando si specifica un handle di connessione condiviso, l'handle di connessione restituito dalla chiamata MQCONNX può essere passato nelle chiamate MQI successive su qualsiasi thread nel processo.

Nota: È possibile utilizzare un handle di connessione condiviso su un client WebSphere MQ MQI per connettersi a un gestore code del server che non supporta gli handle di connessione condivisi.

Per ulteriori informazioni, vedere “Utilizzo di MQCONNX” a pagina 359.

Creazione di applicazioni per client MQI WebSphere MQ

Le applicazioni possono essere create ed eseguite in WebSphere MQ ambiente client MQI. L'applicazione deve essere creata e collegata al client MQI WebSphere MQ utilizzato. Il modo in cui le applicazioni vengono create e collegate varia in base alla piattaforma e al linguaggio di programmazione utilizzato.

Se un'applicazione deve essere eseguita in un ambiente client, è possibile scriverla nelle lingue mostrate nella seguente tabella:

Tabella 47. Linguaggi di programmazione supportati in ambienti client

Piattaforma client	C	C++	COBOL	pTAL	RPG	Visual Basic
AIX	Sì	Sì	Sì			
HP Integrity NonStop Server	Sì		Sì	Sì		
HP-UX	Sì	Sì	Sì			
Linux	Sì	Sì	Sì			
Solaris	Sì	Sì	Sì			
Windows	Sì	Sì	Sì			Sì

Consultare gli argomenti correlati per istruzioni per il collegamento o la creazione di applicazioni client in queste lingue.

Collegamento di applicazioni C con il WebSphere MQ codice client MQI

Dopo aver scritto l'applicazione WebSphere MQ che si desidera eseguire sul client WebSphere MQ MQI, è necessario collegarlo a un gestore code.

È possibile collegare l'applicazione a un gestore code in due modi:

1. Direttamente, nel qual caso il gestore code deve essere sulla stessa stazione di lavoro dell'applicazione
2. In un file di libreria client, che fornisce l'accesso ai gestori code sulla stessa o su una stazione di lavoro diversa

WebSphere MQ fornisce un file della libreria client per ciascun ambiente:

AIX
La libreria libmqic.a per le applicazioni non con thread o la libreria libmqic_r.a per le applicazioni con thread.

HP-UX
La libreria libmqic.sl per le applicazioni non con thread o la libreria libmqic_r.sl per le applicazioni con thread.

Linux
La libreria libmqic.so per le applicazioni senza thread o la libreria libmqic_r.so per le applicazioni con thread.

Solaris
libmqic.so.

Se si desidera utilizzare i programmi su una workstation su cui è installato solo il client WebSphere MQ MQI per Solaris, è necessario ricompilare i programmi per collegarli alla libreria del client:

```
$ /opt/SUNWspro/bin/cc -o <prog> <prog> c -mt -lmqic \
-lsocket -lc -lnsl -ldl
```

I parametri devono essere immessi nell'ordine corretto, come mostrato.

Finestre
MQIC32.LIB.

Collegamento di applicazioni C+ + con il codice client MQI WebSphere MQ

È possibile scrivere le applicazioni da eseguire sul client in C + +. I metodi di creazione variano a seconda dell'ambiente.

Per informazioni su come collegare le applicazioni C++, consultare Creazione di programmi C++ WebSphere MQ.

Per i dettagli completi di tutti gli aspetti relativi all'utilizzo di C + +, consultare Utilizzo di C + +

Collegamento delle applicazioni COBOL con il codice del client MQI IBM WebSphere MQ

Dopo aver scritto un'applicazione COBOL che si desidera eseguire sul client IBM WebSphere MQ MQI, è necessario collegarla a una libreria appropriata.

IBM WebSphere MQ fornisce un file di libreria client per ciascun ambiente:

AIX
Collegare l'applicazione COBOL non sottoposta a thread con la libreria libmqicb.a o l'applicazione COBOL sottoposta a thread con libmqicb_r.a.

HP-UX
Collegare l'applicazione COBOL senza thread con la libreria libmqicb.sl o l'applicazione COBOL con thread con libmqicb_r.sl.

Linux
Collegare l'applicazione COBOL non sottoposta a thread con la libreria libmqicb.so o l'applicazione COBOL sottoposta a thread con libmqicb_r.so.

Solaris
Collegare l'applicazione COBOL non sottoposta a thread con la libreria libmqicb.so o l'applicazione COBOL sottoposta a thread con libmqicb_r.so.

Windows
Collegare il codice dell'applicazione alla libreria MQICCBB per COBOL a 32 bit. Il client IBM WebSphere MQ MQI per Windows non supporta COBOL a 16 bit.

Collegamento delle applicazioni Visual Basic con il codice client MQI WebSphere MQ

È possibile collegare le applicazioni Visual Basic con il codice client WebSphere MQ MQI su Windows .

Collegare l'applicazione Visual Basic con i seguenti file di inclusione:

CMQB.bas
MQI

CMQBB.bas
MQAI

CMQCFB.bas
Comandi PCF

CMQXB.bas
Canali

Impostare `mqtype=2` per il client nel compilatore Visual Basic, per garantire la selezione automatica corretta della dll client:

MQIC32.dll
Windows 2000, Windows XP e Windows 2003

Esecuzione di applicazioni nell'ambiente client IBM WebSphere MQ MQI

È possibile eseguire un'applicazione IBM WebSphere MQ sia in ambiente IBM WebSphere MQ completo che in ambiente client IBM WebSphere MQ MQI senza modificare il codice, purché vengano soddisfatte determinate condizioni.

Queste condizioni sono:

- Non è necessario che l'applicazione si connetta a più di un gestore code contemporaneamente.
- Il nome del gestore code non ha come prefisso un asterisco (*) su una chiamata `MQCONN` o `MQCONNX` .
- L'applicazione non deve utilizzare alcuna delle eccezioni elencate in Quali applicazioni vengono eseguite su un client IBM WebSphere MQ MQI?

Nota: Le librerie utilizzate in fase di modifica del collegamento determinano l'ambiente in cui deve essere eseguita l'applicazione.

Quando si utilizza l'ambiente client IBM WebSphere MQ MQI, tenere presente che:

- Ogni applicazione in esecuzione nell'ambiente client IBM WebSphere MQ MQI ha le proprie connessioni ai server. Un'applicazione stabilisce una connessione a un server ogni volta che emette una chiamata `MQCONN` o `MQCONNX` .
- Un'applicazione invia e riceve i messaggi in modo sincrono. Ciò implica un'attesa tra il momento in cui la chiamata viene emessa sul client e il ritorno di un codice di completamento e di un codice motivo sulla rete.
- Tutta la conversione dei dati viene effettuata dal server, ma consultare anche MQCCSID per informazioni sulla sostituzione del CCSID configurato della macchina.

Connessione delle applicazioni client IBM WebSphere MQ MQI ai gestori code

Un'applicazione in esecuzione in un ambiente client MQI IBM WebSphere MQ può connettersi ad un gestore code in vari modi. È possibile utilizzare le variabili di ambiente, la struttura MQCNO o una tabella di definizione client.

Quando un'applicazione in esecuzione in un ambiente client IBM WebSphere MQ emette una chiamata MQCONN o MQCONNX, il client identifica come effettuare la connessione. Quando una chiamata MQCONNX viene emessa da un'applicazione su un client IBM WebSphere MQ , la libreria del client MQI ricerca le informazioni sul canale client nel seguente ordine:

1. Utilizzo del contenuto dei campi *`ClientConnOffset`* o *`ClientConnPtr`* della struttura MQCNO (se fornita). Questi campi identificano la struttura di definizioni del canale (MQCD) da utilizzare come definizione del canale di connessione client. I dettagli di connessione possono essere sovrascritti utilizzando un'uscita di preconnessione. Per ulteriori informazioni, consultare “Riferimento alle definizioni di connessione mediante un'uscita di pre - connessione da un repository” a pagina 425.
2. Se la variabile di ambiente MQSERVER è impostata, viene utilizzato il canale definito.
3. Se un file `mqclient.ini` è definito e contiene un parametro ServerConnection, viene utilizzato il canale che definisce. Per ulteriori informazioni, consultare Configurazione di un client utilizzando un file di configurazione e la stanza CHANNELS del file di configurazione del client.
4. Se le variabili di ambiente MQCHLLIB e MQCHLTAB sono impostate, viene utilizzata la tabella di definizione del canale client a cui puntano.
5. Se un file `mqclient.ini` è definito e contiene gli attributi Directory ChannelDefinitione File ChannelDefinition, questi attributi vengono utilizzati per individuare la tabella di definizione del canale client. Per ulteriori informazioni, consultare Configurazione di un client utilizzando un file di configurazione e la stanza CHANNELS del file di configurazione del client.
6. Infine, se le variabili di ambiente non sono impostate, il client ricerca una tabella di definizione del canale client con un percorso e un nome stabiliti dal `DefaultPrefix` del file `mqs.ini` . Se la ricerca di una tabella di definizione client ha esito negativo, il client utilizza i seguenti percorsi:
   - Sistemi UNIX and Linux : `/var/mqm/AMQCLCHL.TAB`
   - Windows: `C:\Program Files\IBM\Websphere MQ\amqclchl.tab`

La prima delle opzioni descritte nell'elenco precedente (utilizzando i campi *ClientConnOffset* o *ClientConnPtr* di MQCNO) è supportata solo dalla chiamata MQCONNX. Se l'applicazione utilizza MQCONN anziché MQCONNX, le informazioni sul canale vengono ricercate nei restanti cinque modi nell'ordine mostrato nell'elenco. Se il client non riesce a trovare le informazioni sul canale, la chiamata MQCONN o MQCONNX non riesce.

Il nome del canale (per la connessione client) deve corrispondere al nome del canale di connessione server definito sul server perché la chiamata MQCONN o MQCONNX abbia esito positivo.

Se si riceve un codice di ritorno MQRC_Q_MGR_NOT_AVAILABLE dall'applicazione con un messaggio di errore nel file di log degli errori `AMQ9517 - File danneggiato`, consultare Migration and client channel definition tables (CCDT).

Concetti correlati

Tabella definizione canale client

Attività correlate

Configurazione delle connessioni tra il server e il client

Riferimenti correlati

SERVER MQT

MQCHLLIB

MQCHLTAB

Connessione delle applicazioni client ai gestori code utilizzando le variabili di ambiente

Le informazioni sul canale del client possono essere fornite a una applicazione in esecuzione in un ambiente client dalle variabili di ambiente MQSERVER, MQCHLLIB e MQCHLTAB.

Per dettagli su queste variabili, consultare MQSERVER, MQCHLLIB e MQCHLTAB .

Connessione delle applicazioni client ai gestori code utilizzando la struttura MQCNO

È possibile specificare la definizione del canale in una struttura di definizione del canale (MQCD), che viene fornito utilizzando la struttura MQCNO della chiamata MQCONNX.

Per ulteriori informazioni, consultare Utilizzo della struttura MQCNO su una chiamata MQCONNX .

Connessione di applicazioni client ai gestori code utilizzando una tabella di definizione del canale client

Se si utilizza il comando MQSC DEFINE CHANNEL, i dettagli forniti vengono inseriti nella tabella di definizione del canale client (ccdt). Il contenuto del parametro *QMgrName* della chiamata MQCONN o MQCONNX determina il gestore code a cui si connette il client.

Il client accede a questo file per determinare il canale che verrà utilizzato da un'applicazione. Se è presente più di una definizione di canale adatta, la scelta del canale è influenzata dagli attributi del canale client (CLNTWGHT) e di affinità di connessione (AFFINITY).

Ruolo della tabella di definizione del canale client

La tabella di definizione del canale client (CCDT) contiene le definizioni dei canali di connessione client. È particolarmente utile se le applicazioni client potrebbero dover connettersi a diversi gestori code alternativi.

La tabella di definizione del canale client viene creata quando si definisce un gestore code.

Nota: Lo stesso file può essere utilizzato da più di un client IBM WebSphere MQ . È possibile accedere a diverse versioni di questo file utilizzando le variabili di ambiente MQCHLLIB e MQCHLTAB IBM WebSphere MQ . Consultare Utilizzo di variabili di ambiente WebSphere MQ per informazioni sulle variabili di ambiente.

Gruppi di gestori code in CCDT

È possibile definire una serie di connessioni nella tabella di definizione del canale client (CCDT) come un gruppo di gestori code . È possibile connettere un'applicazione a un gestore code che fa parte di un

gruppo di gestori code. Questa operazione può essere eseguita prefissando il nome del gestore code su una chiamata MQCONN o MQCONNX con un asterisco.

È possibile scegliere di definire le connessioni a più di una macchina server perché:

- Si desidera connettere un client a uno qualsiasi di una serie di gestori code in esecuzione, per migliorare la disponibilità.
- Si desidera riconnettere un client allo stesso gestore code a cui si è connesso correttamente l'ultima volta, ma connettersi a un altro gestore code se la connessione non riesce.
- Si desidera essere in grado di ritentare una connessione client a un gestore code differente se la connessione non riesce, immettendo nuovamente MQCONN nel programma client.
- Se la connessione non riesce, si desidera riconnettere automaticamente una connessione client a un altro gestore code, senza scrivere alcun codice client.
- Si desidera riconnettere automaticamente una connessione client a un'altra istanza di un gestore code a più istanze se un'istanza in standby assume il controllo, senza scrivere alcun codice client.
- Si desidera bilanciare le connessioni client tra diversi gestori code, con più client che si collegano ad alcuni gestori code rispetto ad altri.
- Si desidera estendere la riconnessione di molte connessioni client su più gestori code e nel corso del tempo, nel caso in cui l'elevato volume di connessioni causi un malfunzionamento.
- Si desidera spostare i gestori code senza modificare il codice dell'applicazione client.
- Si desidera scrivere programmi di applicazione client che non devono conoscere i nomi dei gestori code.

Non è sempre appropriato connettersi a gestori code differenti. Un client transazionale esteso o un client Java in WebSphere Application Server, ad esempio, potrebbe dover connettersi a un'istanza del gestore code prevedibile. La riconnessione automatica del client non è supportata dalle classi WebSphere MQ per Java.

Un gruppo di gestori code è una serie di connessioni definite nella tabella di definizione del canale client (CCDT). La serie è definita dai membri che hanno lo stesso valore dell'attributo **QMNAME** nelle loro definizioni di canale.

Figura 70 a pagina 365 è una rappresentazione grafica di una tabella di connessione client, che mostra tre gruppi di gestori code, due gruppi di gestori code denominati scritti in CCDT come **QMNAME**(QM1) e **QMNAME**(QMGrp1)e un gruppo vuoto o predefinito scritto come **QMNAME**(' ').

1. Il gruppo di gestori code QM1 dispone di tre canali di connessione client, che lo collegano ai gestori code QM1 e QM2. QM1 potrebbe essere un gestore code a più istanze ubicato su due server differenti.
2. Il gruppo di gestori code predefinito ha sei canali di connessione client che lo collegano a tutti i gestori code.
3. QMGrp1 dispone di canali di connessione client a due gestori code, QM4 e QM5.

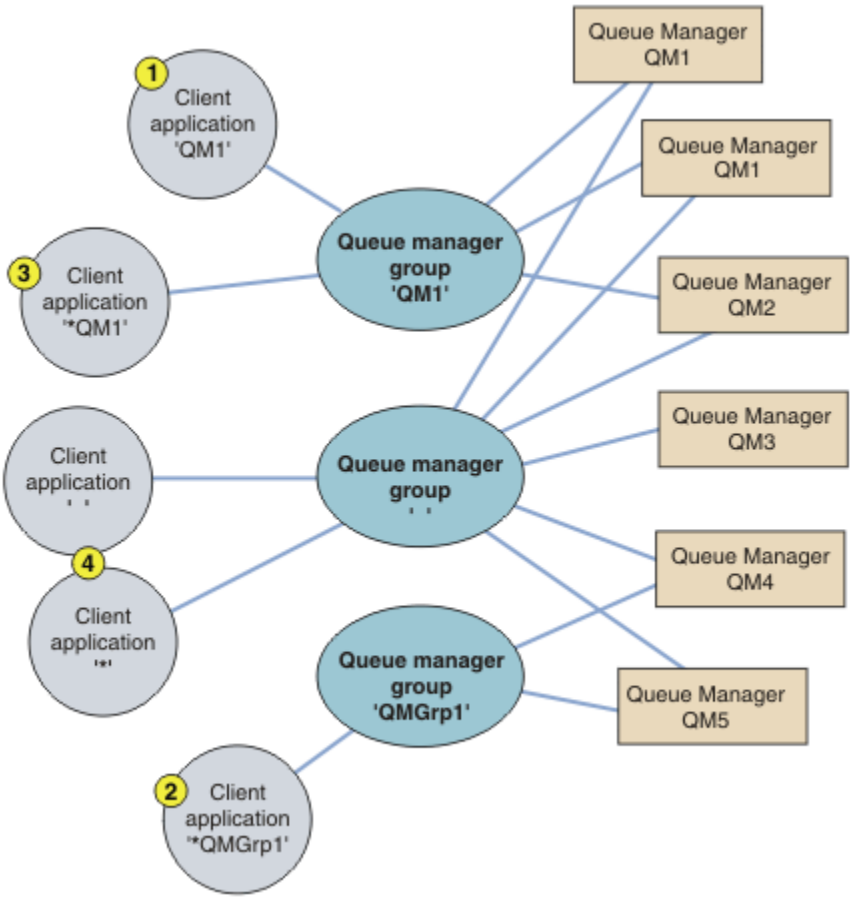


Figura 70. Gruppi gestore code

Quattro esempi di utilizzo di questa tabella di connessione client sono descritti con la guida delle applicazioni client numerate in Figura 70 a pagina 365.

1. Nel primo esempio, l'applicazione client passa un nome gestore code, `QM1`, come parametro **`QmgrName`** alla sua chiamata `MQCONN` o `MQCONNX` MQI. Il codice client WebSphere MQ seleziona il gruppo di gestori code corrispondente, QM1. Il gruppo contiene tre canali di collegamento e il client WebSphere MQ MQI tenta di connettersi a `QM1` utilizzando ciascuno di tali canali, finché non trova un listener WebSphere MQ per la connessione collegata a un gestore code in esecuzione denominato `QM1`.

   L'ordine dei tentativi di connessione dipende dal valore dell'attributo AFFINITY della connessione client e dalle ponderazioni del canale client. All'interno di questi vincoli, l'ordine dei tentativi di connessione viene randomizzato, sia sulle tre connessioni possibili, sia nel tempo, al fine di distribuire il carico di effettuare connessioni.

   La chiamata `MQCONN` o `MQCONNX` emessa dall'applicazione client ha esito positivo quando viene stabilita una connessione a un'istanza in esecuzione di `QM1`.
2. Nel secondo esempio, l'applicazione client passa un nome gestore code con un prefisso con un asterisco, `*QMGrp1` come parametro **`QmgrName`** alla sua chiamata MQI `MQCONN` o `MQCONNX` . Il client WebSphere MQ seleziona il gruppo di gestori code corrispondente, QMGrp1. Questo gruppo contiene due canali di connessione client e il client WebSphere MQ MQI tenta di connettersi a *qualsiasi* gestore code utilizzando ciascun canale a turno. In questo esempio, il client WebSphere MQ MQI deve stabilire una connessione corretta; il nome del gestore code a cui si connette non è importante.

   La regola per l'ordine dei tentativi di connessione è la stessa di prima. L'unica differenza è che prefissando il nome del gestore code con un asterisco, il client indica che il nome del gestore code non è rilevante.

La chiamata MQCONN o MQCONNX emessa dall'applicazione client ha esito positivo quando viene stabilita una connessione a un'istanza in esecuzione di qualsiasi gestore code connesso dai canali nel gruppo di gestori code QMGrp1 .

3. Il terzo esempio è essenzialmente lo stesso del secondo perché il parametro **QmgrName** è preceduto da un asterisco, *QM1. L'esempio illustra che non è possibile determinare a quale gestore code si connetterà una connessione del canale client ispezionando l'attributo QMNAME in un'unica definizione di canale. Il fatto che l'attributo **QMNAME** della definizione di canale sia QM1, non è sufficiente per richiedere una connessione a un gestore code denominato QM1. Se l'applicazione client antepone il parametro **QmgrName** con un asterisco, qualsiasi gestore code è una destinazione di connessione possibile.

   In questo caso, le chiamate MQCONN o MQCONNX emesse dall'applicazione client hanno esito positivo quando viene stabilita una connessione a un'istanza in esecuzione di QM1 o QM2.

4. Il quarto esempio illustra l'utilizzo del gruppo predefinito. In questo caso, l'applicazione client passa un asterisco, '*' o uno spazio vuoto ' ', come parametro **QmgrName** alla sua chiamata MQI MQCONN o MQCONNX . Per convenzione nella definizione di canale client, un attributo **QMNAME** vuoto indica il gruppo di gestori code predefinito e un parametro **QmgrName** vuoto o asterisco corrisponde a un attributo **QMNAME** vuoto.

   In questo esempio, il gruppo di gestori code predefinito ha connessioni del canale client a tutti i gestori code. Selezionando il gruppo di gestori code predefinito, l'applicazione potrebbe essere connessa a qualsiasi gestore code del gruppo.

   La chiamata MQCONN o MQCONNX emessa dall'applicazione client ha esito positivo quando viene stabilita una connessione a un'istanza in esecuzione di qualsiasi gestore code.

   Nota: Il gruppo predefinito è diverso da un gestore code predefinito, anche se un'applicazione utilizza un parametro **QmgrName** vuoto per connettersi al gruppo di gestori code predefinito o al gestore code predefinito. Il concetto di gruppo di gestori code predefiniti è rilevante solo per un'applicazione client e per un gestore code predefinito per un'applicazione server.

Definire i canali di connessione client solo su un gestore code, inclusi i canali che si connettono a un secondo o terzo gestore code. *Non* definirli su due gestori code e provare quindi ad unire le due tabelle di definizione del canale client. Il client può accedere a una sola tabella di definizione del canale client.

Esempi

Esaminare nuovamente l' elenco dei motivi per cui si utilizzano i gruppi di gestori code all'inizio dell'argomento. In che modo l'uso di un gruppo di gestori code fornisce tali funzioni?

Connettersi a uno qualsiasi di una serie di gestori code.
Definire un gruppo di gestori code con le connessioni a tutti i gestori code nel set e connettersi al gruppo utilizzando il parametro **QmgrName** preceduto da un asterisco.

Riconnettersi allo stesso gestore code, ma connettersi a uno differente, se il gestore code connesso all'ultima volta non è disponibile.
Definire un gruppo di gestori code come prima, ma impostare l'attributo **AFFINITY**(PREFERRED) su ciascuna definizione di canale client.

Riprovare una connessione a un altro gestore code se una connessione non riesce.
Connettersi a un gruppo di gestori code ed emettere nuovamente la chiamata MQCONN o MQCONNX MQI se la connessione è interrotta o se il gestore code ha esito negativo.

Riconnettersi automaticamente ad un altro gestore code se una connessione non riesce.
Connettersi a un gruppo di gestori code utilizzando l'opzione MQCONNX **MQCNO** MQCNO_RECONNECT.

Riconnettersi automaticamente a un'altra istanza di un gestore code a più istanze.
Eseguire la stessa operazione dell'esempio precedente. In questo caso, se si desidera limitare il gruppo di gestori code per connettersi alle istanze di un particolare gestore code a più istanze, definire il gruppo con le connessioni solo alle istanze del gestore code a più istanze.

È anche possibile richiedere all'applicazione client di emettere la relativa chiamata MQCONN o MQCONNX MQI senza un asterisco come prefisso al parametro **QmgrName** . In questo modo l'applicazione client può connettersi solo a un gestore code denominato. Infine, è possibile impostare

l'opzione **MQCNO** su MQCNO_RECONNECT_Q_MGR. Questa opzione accetta le riconnessioni allo stesso gestore code precedentemente connesso. È anche possibile utilizzare questo valore per limitare le riconnessioni alla stessa istanza di un gestore code normale.

Bilanciare le connessioni client tra i gestori code, con più client connessi ad alcuni gestori code rispetto ad altri.
Definire un gruppo di gestori code e impostare l'attributo **CLNTWGHT** su ciascuna definizione di canale client per distribuire le connessioni in modo non uniforme.

Distribuire il carico di riconnessione del client in modo non uniforme, e diffonderlo nel tempo, dopo un errore di connessione o di gestore code.
Eseguire la stessa operazione dell'esempio precedente. Il client WebSphere MQ MQI randomizza le riconnessioni tra i gestori code e diffonde le riconnessioni nel tempo.

Spostare i gestori code senza modificare il codice client.
CDT isola l'applicazione client dall'ubicazione del gestore code.

È possibile scegliere se distribuire la tabella di connessione client a ciascun client o posizionare CCDT su un file system condiviso per ciascun client a cui fare riferimento. In alternativa, utilizzare la versione programmatica della CCDT supportata nella chiamata MQI MQCONNX e richiamare un servizio per passare la CCDT all'applicazione client.

Scrivere un'applicazione client che non conosca i nomi dei gestori code.
Utilizzare i nomi dei gruppi di gestori code e definire una convenzione di denominazione per i nomi dei gruppi di gestori code che sia rilevante per le applicazioni client nella propria organizzazione e che rifletta l'architettura delle soluzioni piuttosto che la denominazione dei gestori code.

Connessione ai gruppi di condivisione code
È possibile connettere l'applicazione a un gestore code che fa parte di un gruppo di condivisione code. Questa operazione può essere eseguita utilizzando il nome del gruppo di condivisione code invece del nome del gestore code nella chiamata MQCONN o MQCONNX.

I gruppi di condivisione code hanno nomi formati di massimo quattro caratteri. Il nome deve essere univoco nella rete e diverso da qualsiasi altro nome di gestore code.

La definizione di canale client deve utilizzare l'interfaccia generica del gruppo di condivisione code per connettersi a un gestore code disponibile nel gruppo. Per ulteriori informazioni, consultare Connessione di un client a un gruppo di condivisione code. Viene eseguito un controllo per assicurarsi che il gestore code a cui si connette il listener sia un membro del gruppo di condivisione code.

Esempi di importanza e affinità del canale
Questi esempi illustrano come vengono selezionati i canali di connessione client quando vengono utilizzati ClientChannelWeights diversi da zero.

Gli attributi ClientChannelPeso e ConnectionAffinity controllano il modo in cui vengono selezionati i canali di connessione client quando è disponibile più di un canale adatto per una connessione. Questi canali sono configurati per connettersi a diversi gestori code in modo da fornire maggiore disponibilità, bilanciamento del carico di lavoro o entrambi. Le chiamate MQCONN che potrebbero risultare in una connessione a uno dei diversi gestori code devono anteporre un asterisco al nome del gestore code, come descritto in:Esempi di chiamate MQCONN: Esempio 1. Il nome del gestore code include un asterisco (*).

I canali candidati applicabili per una connessione sono quelli in cui l'attributo QMNAME corrisponde al nome gestore code specificato nella chiamata MQCONN. Se tutti i canali applicabili per una connessione hanno un ClientChannelWeight uguale a zero (impostazione predefinita), vengono selezionati in ordine alfabetico come nell'esempio: Esempi di chiamate MQCONN: Esempio 1. Il nome del gestore code include un asterisco (*).

I seguenti esempi illustrano cosa accade quando vengono utilizzati ClientChannelWeights diversi da zero. Si noti che, poiché questa funzione implica la selezione di canali pseudo - casuali, gli esempi mostrano una sequenza di azioni che potrebbero verificarsi piuttosto che ciò che sicuramente accadrà.

Esempio 1. Selezione dei canali quando ConnectionAffinity è impostata su PREFERRED
Questo esempio illustra il modo in cui un client MQI WebSphere MQ seleziona un canale da una CCDT, dove ConnectionAffinity è impostata su PREFERRED.

In questo esempio, diverse macchine client utilizzano una CCDT (Client Channel Definition Table) fornita da un gestore code. CDT include canali di connessione client con i seguenti attributi (visualizzati utilizzando la sintassi del comando 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)
```

L'applicazione emette MQCONN (*CORE)

Il canale A non è un candidato per questa connessione, perché l'attributo QMNAME non corrisponde. I canali B, C e D sono identificati come candidati e sono posti in ordine di preferenza in base alla loro ponderazione. In questo esempio l'ordine potrebbe essere C, B, D. Il client tenta di collegarsi al gestore code in core2.ops.company.example. Il nome del gestore code a tale indirizzo non viene controllato perché la chiamata MQCONN includeva un asterisco nel nome gestore code.

È importante notare che, con `AFFINITY(PREFERRED)`, ogni volta che questa particolare macchina client si collega, posizionerà i canali nello stesso ordine iniziale di preferenza. Ciò si applica anche quando le connessioni provengono da processi diversi o in momenti diversi.

In questo esempio, non è possibile raggiungere il gestore code in core.2.ops.company.example . Il client tenta di connettersi a core1.ops.company.example perché il canale B è il successivo nell'ordine di preferenza. Inoltre, il canale C viene degradato per diventare il meno preferito.

Una seconda chiamata MQCONN (*CORE) viene emessa dalla stessa applicazione. Il canale C è stato retrocesso dalla connessione precedente, quindi il canale più preferito è ora B. Questa connessione viene effettuata a core1.ops.company.example.

Una seconda macchina che condivide la stessa tabella di definizione canale client posiziona i canali in un ordine di preferenza iniziale diverso. Ad esempio, D, B, C. In circostanze normali, con tutti i canali che funzionano, le applicazioni su questa macchina sono connesse a core3.ops.company.example mentre quelle sulla prima macchina sono connesse a core2.ops.company.example. Ciò consente il bilanciamento del carico di lavoro di un numero elevato di client su più gestori code, consentendo a ciascun client di connettersi allo stesso gestore code, se disponibile.

Esempio 2. Selezione dei canali quando ConnectionAffinity è impostata su NONE
Questo esempio illustra in che modo un client MQI WebSphere MQ seleziona un canale da una CCDT, dove ConnectionAffinity è impostato su NONE.

In questo esempio, alcuni client utilizzano una CCDT (Client Channel Definition Table) fornita da un gestore code. CDT include canali di connessione client con i seguenti attributi (visualizzati utilizzando la sintassi del comando 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)
```

L'applicazione emette MQCONN (*CORE). Come nell'esempio precedente, il canale A non viene considerato perché QMNAME non corrisponde. I canali B, C o D sono selezionati in base al loro peso, con probabilità del 50%, 30% o 20%. In questo esempio, il canale B potrebbe essere selezionato. Non è stato creato alcun ordine di preferenza persistente.

Viene effettuata una seconda chiamata MQCONN (*CORE). Ancora una volta, viene selezionato uno dei tre canali applicabili, con le stesse probabilità. In questo esempio, viene scelto il canale C. Tuttavia,

core2.ops.company.example non risponde, quindi viene effettuata un'altra scelta tra i rimanenti canali candidati. Il canale B è selezionato e l'applicazione è connessa a core1.ops.company.example.

Con AFFINITY (NONE), ogni chiamata MQCONN è indipendente da qualsiasi altra. Pertanto, quando questa applicazione di esempio effettua un terzo MQCONN (*CORE), potrebbe ancora una volta tentare la connessione tramite il canale interrotto C, prima di scegliere uno tra B o D.

Esempi di chiamate MQCONN

Esempi di utilizzo di MQCONN per la connessione a un gestore code specifico o a uno di un gruppo di gestori code.

In ognuno dei seguenti esempi, la rete è la stessa; esiste una connessione definita per due server dallo stesso client WebSphere MQ MQI. In questi esempi, è possibile utilizzare la chiamata MQCONNX al posto della chiamata MQCONN.

Esistono due gestori code in esecuzione sulle macchine server, uno denominato SALE e l'altro denominato SALE_BACKUP.

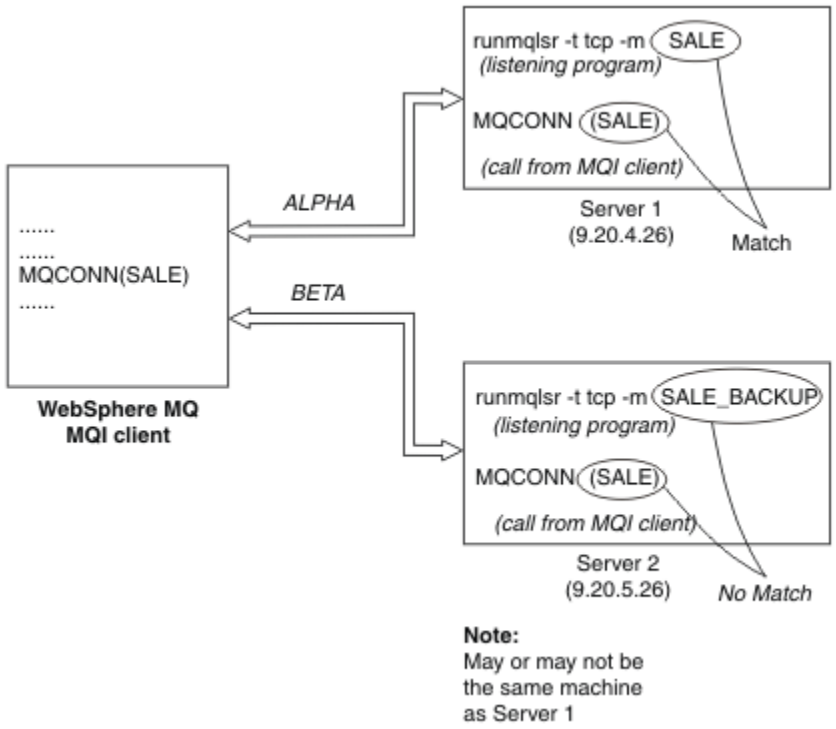


Figura 71. Esempio MQCONN

Le definizioni per i canali in questi esempi sono:

Definizioni SALDI:

```
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)
```

Definizione SALE_BACKUP:

```
DEFINE CHANNEL(BETA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to WebSphere MQ MQI client')
```

Le definizioni del canale client possono essere riepilogate come segue:

Nome	CHLTYPE	TRPTYPE	CONNAME	QMNAME
ALPHA	CLNTCONN	TCP	9.20.4.26	Vendita
BETA	CLNTCONN	TCP	9.20.5.26	Vendita

Cosa dimostrano gli esempi MQCONN
Gli esempi mostrano l'utilizzo di più gestori code come sistema di backup.

Si supponga che il collegamento di comunicazione al server 1 sia temporaneamente interrotto. Viene dimostrato l'utilizzo di più gestori code come sistema di backup.

Ogni esempio copre una chiamata MQCONN differente e fornisce una spiegazione di ciò che accade nell'esempio specifico presentato, applicando le seguenti regole:

1. La CCDT (client channel definition table) viene scansionata in ordine alfabetico per un nome gestore code (campo QMNAME) corrispondente a quello fornito nella chiamata MQCONN.
2. Se viene trovata una corrispondenza, viene utilizzata la definizione di canale.
3. Si è tentato di avviare il canale sulla macchina identificata dal nome connessione (CONNAME). Se l'operazione ha esito positivo, l'applicazione continua. Richiede:
   - Un listener in esecuzione sul server.
   - Il listener da connettere allo stesso gestore code di quello a cui il client desidera connettersi (se specificato).
4. Se il tentativo di avviare il canale non riesce e c'è più di una voce nella tabella di definizione del canale client (in questo esempio ci sono due voci), viene ricercata un'ulteriore corrispondenza nel file. Se viene trovata una corrispondenza, l'elaborazione continua al passo 1.
5. Se non viene trovata alcuna corrispondenza o non ci sono più voci nella tabella di definizione del canale client e il canale non è stato avviato, l'applicazione non è in grado di connettersi. Nella chiamata MQCONN vengono restituiti un codice motivo e un codice di completamento appropriati. L'applicazione può intraprendere un'azione in base ai codici motivo e di completamento restituiti.

Esempio 1. Il nome del gestore code include un asterisco (*)
In questo esempio, l'applicazione non è interessata a quale gestore code si connette. L'applicazione emette una chiamata MQCONN per un nome gestore code che include un asterisco. Viene scelto un canale adatto.

L'applicazione ha i seguenti problemi:

```
MQCONN (*SALE)
```

Seguendo le regole, questo è ciò che accade in questa istanza:

1. La CCDT (client channel definition table) viene sottoposta a scansione per il nome del gestore code SALE, corrispondente alla chiamata MQCONN dell'applicazione.
2. Sono state trovate definizioni di canale per ALPHA e BETA .
3. Se un canale ha un valore CLNTWGHT pari a 0, questo canale viene selezionato. Se entrambi hanno un valore CLNTWGHT pari a 0, il canale ALPHA viene selezionato perché è il primo in sequenza alfabetica. Se entrambi i canali hanno un valore CLNTWGHT diverso da zero, un canale viene selezionato in modo casuale, in base al suo peso.
4. Viene effettuato un tentativo di avviare il canale.
5. Se è stato selezionato il canale BETA , il tentativo di avviarlo ha esito positivo.

6. Se è stato selezionato il canale ALPHA , il tentativo di avviarlo NON ha esito positivo perché il collegamento di comunicazione è interrotto. Si applicano quindi le seguenti operazioni:
   a. L'unico altro canale per il nome gestore code SALE è BETA.
   b. È stato effettuato un tentativo di avviare questo canale - l'operazione ha esito positivo.
7. Un controllo per verificare che un listener sia in esecuzione indica che ne esiste uno in esecuzione. Non è connesso al gestore code SALE , ma poiché il parametro della chiamata MQI contiene un asterisco (*), non viene eseguito alcun controllo. L'applicazione è connessa al gestore code SALE_BACKUP e continua l'elaborazione.

Esempio 2. Nome gestore code specificato
In questo esempio l'applicazione deve connettersi a uno specifico gestore code. L'applicazione emette una chiamata MQCONN per tale nome gestore code. Viene scelto un canale adatto.

L'applicazione richiede una connessione a un gestore code specifico, denominato SALE, come visualizzato nella chiamata MQI:

```
MQCONN (SALE)
```

Seguendo le regole, questo è ciò che accade in questa istanza:

1. La tabella di definizione del canale client (CCDT) viene scansionata in ordine alfabetico per il nome del gestore code SALE, corrispondente alla chiamata MQCONN dell'applicazione.
2. La prima definizione di canale trovata per la corrispondenza è ALPHA.
3. Viene effettuato un tentativo di avviare il canale - *non* è stato eseguito correttamente perché il collegamento di comunicazione è interrotto.
4. La tabella di definizione del canale client viene nuovamente sottoposta a scansione per il nome gestore code SALE e viene trovato il nome canale BETA .
5. È stato effettuato un tentativo di avviare il canale - l'operazione ha esito positivo.
6. Un controllo per verificare che un listener sia in esecuzione mostra che ne esiste uno in esecuzione, ma non è connesso al gestore code SALE .
7. Non ci sono ulteriori voci nella tabella di definizione del canale client. L'applicazione non può continuare e riceve il codice di ritorno MQRC_Q_MGR_NOT_AVAILABLE.

Esempio 3. Il nome del gestore code è vuoto o un asterisco (*)
In questo esempio, l'applicazione non è interessata a quale gestore code si connette. L'applicazione emette un MQCONN specificando un nome gestore code vuoto o un asterisco. Viene scelto un canale adatto.

Questo viene trattato come “Esempio 1. Il nome del gestore code include un asterisco (*)” a pagina 370.

Nota: Se questa applicazione fosse in esecuzione in un ambiente diverso da WebSphere MQ client MQI e il nome fosse vuoto, tenterebbe di connettersi al gestore code predefinito. Questo *non* è il caso quando viene eseguito da un ambiente client; il gestore code a cui si accede è quello associato al listener a cui si connette il canale.

L'applicazione ha i seguenti problemi:

```
MQCONN ("")
```

o

```
MQCONN (*)
```

Seguendo le regole, questo è ciò che accade in questa istanza:

1. La tabella di definizione del canale client (CCDT) viene sottoposta a scansione in ordine alfabetico in base alla sequenza dei nomi dei canali, alla ricerca di un nome gestore code vuoto, corrispondente alla chiamata MQCONN dell'applicazione.
2. La voce per il nome del canale ALPHA ha un nome gestore code nella definizione di SALE. *Non* corrisponde al parametro della chiamata MQCONN, che richiede che il nome del gestore code sia vuoto.
3. La voce successiva è per il nome canale BETA.
4. Il `queue manager name` nella definizione è SALE. Ancora una volta, *non* corrisponde al parametro della chiamata MQCONN, che richiede che il nome del gestore code sia vuoto.
5. Non ci sono ulteriori voci nella tabella di definizione del canale client. L'applicazione non può continuare e riceve il codice di ritorno MQRC_Q_MGR_NOT_AVAILABLE.

Attivazione nell'ambiente client

I messaggi inviati dalle applicazioni WebSphere MQ in esecuzione su client WebSphere MQ MQI contribuiscono all'attivazione esattamente come qualsiasi altro messaggio e possono essere utilizzati per attivare programmi sia sul server che sul client.

L'attivazione viene spiegata in dettaglio in Trigger canali.

Il controllo trigger e l'applicazione da avviare devono trovarsi sullo stesso sistema.

Le caratteristiche predefinite della coda attivata sono le stesse dell'ambiente del server. In particolare, se non vengono specificate opzioni di controllo del punto di sincronizzazione MQPMO in un'applicazione client che inserisce i messaggi in una coda attivata locale di un gestore code z/OS , i messaggi vengono inseriti all'interno di un'unità di lavoro. Se la condizione di attivazione viene quindi soddisfatta, il messaggio del trigger viene inserito nella coda di avvio all'interno della stessa unità di lavoro e non può essere richiamato dal controllo del trigger fino a quando l'unità di lavoro non termina. Il processo che deve essere attivato non viene avviato fino al termine dell'unità di lavoro.

Definizione di processo

È necessario definire la definizione di processo sul server, poiché è associata alla coda su cui è impostato il trigger.

L'oggetto processo definisce cosa deve essere attivato. Se il client e il server non sono in esecuzione sulla stessa piattaforma, qualsiasi processo avviato dal controllo trigger deve definire *ApplType*, altrimenti il server utilizza le definizioni predefinite (ovvero, il tipo di applicazione normalmente associato alla macchina server) e causa un errore.

Ad esempio, se il controllo dei trigger è in esecuzione su un client Windows e si desidera inviare una richiesta a un server su un altro sistema operativo, è necessario definire MQAT_WINDOWS_NT altrimenti l'altro sistema operativo utilizzerà le definizioni predefinite e il processo avrà esito negativo.

Controllo trigger

Il controllo dei trigger fornito da prodotti nonz/OS WebSphere MQ viene eseguito negli ambienti client per sistemi UNIX, Linux e Windows .

Per eseguire il controllo trigger, immettere uno dei seguenti comandi:

- 

Su piattaforme Windows, UNIX e Linux :

```
runmqtmc [-m QMgrName] [-q InitQ]
```

La coda di avvio predefinita è SYSTEM.DEFAULT.INITIATION.QUEUE sul gestore code predefinito. La coda di iniziazione è il punto in cui il controllo del trigger cerca i messaggi del trigger. Successivamente, richiama i programmi per i messaggi trigger appropriati. Questo controllo trigger supporta il tipo di applicazione predefinito ed è uguale a `runmqtrm` ad eccezione del fatto che collega le librerie client.

La stringa di comando, creata dal controllo trigger, è la seguente:

1. Il *ApplicId* dalla definizione del processo pertinente. *ApplicId* è il nome del programma da eseguire, come dovrebbe essere immesso sulla riga comandi.
2. La struttura MQTMC2 , racchiusa tra virgolette, ottenuta dalla coda di avvio. Viene avviata una stringa di comando che ha questa stringa, esattamente come fornita, tra virgolette in modo che il comando di sistema la accetti come un parametro.
3. Il *EnvrData* dalla definizione del processo pertinente.

Il controllo dei trigger non controlla se è presente un altro messaggio sulla coda di iniziazione fino al completamento dell'applicazione che ha avviato. Se l'applicazione ha molta elaborazione da fare, il controllo trigger potrebbe non tenere il passo con il numero di messaggi trigger in arrivo. Ci sono due modi per affrontare questa situazione:

1. Avere più controlli trigger in esecuzione

   Se si sceglie di avere più controlli trigger in esecuzione, è possibile controllare il numero massimo di applicazioni che possono essere eseguite contemporaneamente.
2. Eseguire le applicazioni avviate in background

   Se si sceglie di eseguire le applicazioni in background, WebSphere MQ non impone alcuna limitazione sul numero di applicazioni che possono essere eseguite.

Per eseguire l'applicazione avviata in background sui sistemi UNIX and Linux , è necessario inserire una & (e commerciale) alla fine del *EnvrData* della definizione del processo.

Applicazioni CICS (nonz/OS)

Un programma applicativo nonz/OS CICS che emette una chiamata MQCONN o MQCONNX deve essere definito in CEDA come RESIDENT. Se si ricollglia un'applicazione server CICS come client, si rischia di perdere il supporto del punto di sincronizzazione.

Un programma applicativo nonz/OS CICS che emette una chiamata MQCONN o MQCONNX deve essere definito in CEDA come RESIDENT. Per rendere il codice residente il più piccolo possibile, è possibile collegarsi a un programma separato per emettere la chiamata MQCONN o MQCONNX .

Se la variabile di ambiente MQSERVER viene utilizzata per definire la connessione client, deve essere specificata nel CICSENV CICSENV.CMD .

WebSphere Le applicazioni MQ possono essere eseguite in un ambiente server WebSphere MQ o su un client WebSphere MQ senza modificare il codice. Tuttavia, in un ambiente server WebSphere MQ , CICS può agire come coordinatore del punto di sincronizzazione e si utilizza EXEC CICS SYNCPOINT e EXEC CICS SYNCPOINT ROLLBACK anziché MQCMIT e MQBACK. Se un'applicazione CICS viene semplicemente ricollegata come client, il supporto del punto di sincronizzazione viene perso. MQCMIT e MQBACK devono essere utilizzati per l'applicazione in esecuzione su un client MQI WebSphere MQ .

Preparazione ed esecuzione di applicazioni CICS e Tuxedo

Per eseguire le applicazioni CICS e Tuxedo come applicazioni client, si utilizzano librerie diverse da quelle utilizzate con le applicazioni server. Anche l'ID utente con cui viene eseguita l'applicazione è diverso.

Per preparare le applicazioni CICS e Tuxedo all'esecuzione come applicazioni client WebSphere MQ MQI, attenersi alle istruzioni in Configurazione di un client transazionale esteso.

Si noti, tuttavia, che le informazioni che riguardano specificamente la preparazione delle applicazioni CICS e Tuxedo, inclusi i programmi di esempio forniti con WebSphere MQ, presuppongono che si stiano preparando le applicazioni da eseguire su un sistema server WebSphere MQ . Di conseguenza, le informazioni si riferiscono solo a librerie WebSphere MQ che sono destinate all'utilizzo su un sistema server. Durante la preparazione delle applicazioni client, è necessario effettuare le seguenti operazioni:

- Utilizzare la libreria di sistema client appropriata per i bind di lingua utilizzati dall'applicazione. Ad esempio, per le applicazioni scritte in C su AIX, HP-UXo Solaris, utilizzare la libreria libmqic invece di libmqm. su sistemi Windows , utilizzare la libreria mqic.lib invece di mqm.lib.
- Invece delle librerie di sistema del server mostrate in Tabella 48 a pagina 374, per AIX, HP-UXe Solaris e Tabella 49 a pagina 374, per i sistemi Windows , utilizzare le librerie di sistema client equivalenti. Se

una libreria di sistema server non è elencata in queste tabelle, utilizzare la stessa libreria su un sistema client.

Tabella 48. Librerie di sistema client su AIX, HP-UXe Solaris

Libreria per un sistema server WebSphere MQ	**Libreria equivalente da utilizzare su un sistema client WebSphere MQ**
libmqmxa	libmqcxa

Tabella 49. Librerie di sistema client su sistemi Windows

Libreria per un sistema server WebSphere MQ	**Libreria equivalente da utilizzare su un sistema client WebSphere MQ**
mqmxa.lib	mqcxa.lib
mqmtux.lib	mqcxa.lib
mqmenc.lib	mqcxa.lib
mqmcics4.lib	mqccics4.lib

L'ID utente utilizzato da un'applicazione client

Quando si esegue un'applicazione server WebSphere MQ in CICS, normalmente passa dall'utente CICS all'ID utente della transazione. Tuttavia, quando si esegue un'applicazione client MQI WebSphere MQ in CICS, essa conserva l'autorità privilegiata CICS.

Programmi di esempio CICS e Tuxedo

Programmi di esempio CICS e Tuxedo da utilizzare su sistemi AIX, HP-UX, Solaris e Windows.

Tabella 50 a pagina 374 elenca i programmi di esempio di CICS e Tuxedo forniti per l'utilizzo sui sistemi client AIX, HP-UXe Solaris. Tabella 51 a pagina 374 elenca le informazioni equivalenti per i sistemi client Windows . Le tabelle elencano anche i file utilizzati per preparare ed eseguire i programmi. Per una descrizione dei programmi di esempio, vedere “L'esempio di transazione CICS” a pagina 119 e “Esempi TUXEDO” a pagina 157.

Tabella 50. Programmi di esempio per sistemi client AIX, HP-UXe Solaris

Descrizione	**Sorgente**	**Modulo eseguibile**
Programma CICS	amqscic0.ccs	amqscicc
File di intestazione per il programma CICS	amqscih0.h	-
Programma client Tuxedo per inserire messaggi	amqstxpx.c	-
Programma client Tuxedo per richiamare i messaggi	amqstxgx.c	-
Programma del server Tuxedo per i due programmi client	amqstxsx.c	-
File UBBCONFIG per i programmi Tuxedo	ubbstxcx.cfg	-
File di tabella dei campi per i programmi Tuxedo	amqstxvx.flds	-
Visualizza file di descrizione per i programmi Tuxedo	amqstxvx.v	-

Tabella 51. Programmi di esempio per sistemi client Windows

Descrizione	**Sorgente**	**Modulo eseguibile**
Transazione CICS	amqscic0.ccs	amqscicc
File di intestazione per la transazione CICS	amqscih0.h	-

Tabella 51. Programmi di esempio per sistemi client Windows (Continua)		
Descrizione	**Sorgente**	**Modulo eseguibile**
Programma client Tuxedo per inserire messaggi	amqstxpx.c	-
Programma client Tuxedo per richiamare i messaggi	amqstxgx.c	-
Programma del server Tuxedo per i due programmi client	amqstxsx.c	-
File UBBCONFIG per i programmi Tuxedo	ubbstxcx.cfg	-
File di tabella dei campi per i programmi Tuxedo	amqstxvx.fld	-
Visualizza file di descrizione per i programmi Tuxedo	amqstxvx.v	-
Makefile per i programmi Tuxedo	amqstxmc.mak	-
File ENVFILE per i programmi Tuxedo	amqstxen.env	-

Messaggio di errore AMQ5203, come modificato per le applicazioni CICS e Tuxedo

Quando si eseguono applicazioni CICS o Tuxedo che utilizzano un client transazionale esteso, è possibile che vengano visualizzati messaggi di diagnostica standard. Uno di questi è stato modificato per essere utilizzato con un client transazionale esteso

I messaggi che potrebbero essere visualizzati nei file di log degli errori di WebSphere MQ sono documentati in Messaggi diagnostici: AMQ4000-9999. Il messaggio AMQ5203 è stato modificato per l'utilizzo con un client transazionale esteso. Di seguito è riportato il testo del messaggio modificato:

AMQ5203: Si è verificato un errore durante il richiamo dell'interfaccia XA.

Spiegazione

Il numero di errore è & 2 dove il valore 1 indica che il valore degli indicatori forniti di & 1 non era valido, 2 indica che si è tentato di utilizzare le librerie con thread e non con thread nello stesso processo, 3 indica che si è verificato un errore con il nome del gestore code fornito '& 3', 4 indica che l'ID del gestore risorse di & 1 non era valido, 5 indica che si è tentato di utilizzare un secondo gestore code denominato '& 3 'quando un altro gestore code era già connesso, 6 indica che il Gestore transazioni è stato richiamato quando l'applicazione non è connessa a un gestore code, 7 indica che la chiamata XA è stata effettuata mentre era in corso un'altra chiamata, 8 indica che la stringa xa_info' & 4 'nella chiamata xa_open conteneva un valore di parametro non valido per il nome parametro' & 5 'e 9 indica che alla stringa xa_info' & 4 'nella chiamata xa_open manca un parametro obbligatorio, il nome parametro' & 5 '.

Risposta utente

Correggere l'errore e ripetere l'operazione.

Preparazione ed esecuzione di applicazioni Microsoft Transaction Server

Per preparare un'applicazione MTS da eseguire come applicazione client WebSphere MQ MQI, seguire queste istruzioni in base al proprio ambiente.

Per informazioni generali su come sviluppare applicazioni Microsoft Transaction Server (MTS) che accedono alle risorse WebSphere MQ , consultare la sezione relativa a MTS nel Centro assistenza WebSphere MQ .

Per preparare un'applicazione MTS da eseguire come applicazione client MQI WebSphere MQ , effettuare una delle seguenti operazioni per ogni componente dell'applicazione:

- Se il componente utilizza i bind in linguaggio C per MQI, seguire le istruzioni in “Preparazione di programmi C in Windows” a pagina 461 ma collegare il componente alla libreria mqicxa.lib invece di mqic.lib.

- Se il componente utilizza le classi C++ di WebSphere MQ , seguire le istruzioni riportate in “Creazione di programmi C+ + su Windows” a pagina 654 ma collegare il componente alla libreria imqx23vn.lib invece di imqc23vn.lib.
- Se il componente utilizza i bind del linguaggio Visual Basic per MQI, seguire le istruzioni in “Preparazione dei programmi Visual Basic in Windows” a pagina 465 , ma quando si definisce il progetto Visual Basic, immettere `MqType=3` nel campo **Argomenti di compilazione condizionale** .
- Se il componente utilizza WebSphere MQ Automation Classes for ActiveX (MQAX), definire una variabile di ambiente, GMQ_MQ_LIB, con il valore `mqic32xa.dll` .

  È possibile definire la variabile di ambiente dall'interno dell'applicazione oppure è possibile definirla in modo che il suo ambito sia esteso a tutto il sistema. Tuttavia, definendolo a livello di sistema, è possibile che qualsiasi applicazione MQAX esistente, che non definisce la variabile di ambiente dall'interno dell'applicazione, si comporti in modo non corretto.

Preparazione ed esecuzione delle applicazioni WebSphere MQ JMS

È possibile eseguire le applicazioni JMS WebSphere MQ in modalità client, con WebSphere Application Server come gestore transazioni. Potrebbero essere visualizzati alcuni messaggi di avvertenza.

Per preparare ed eseguire WebSphere MQ applicazioni JMS in modalit ... client, con WebSphere Application Server come gestore transazioni, seguire le istruzioni in “Utilizzo delle classi WebSphere MQ per JMS” a pagina 717.

Quando si esegue un'applicazione client JMS WebSphere MQ , potrebbero essere visualizzati i seguenti messaggi di avvertenza:

MQJE080
Unità di licenza insufficienti - eseguire setmqcap

MQJE081
Il file che contiene le informazioni sull'unità di licenza è nel formato errato - eseguire setmqcap

MQJE082
Impossibile trovare il file contenente le informazioni sull'unità di licenza - eseguire setmqcap

Uscite utente, uscite API e servizi installabili WebSphere MQ

È possibile estendere le funzioni del gestore code utilizzando le uscite utente, le uscite API o i servizi installabili. Questo argomento contiene collegamenti alle informazioni sull'utilizzo e lo sviluppo di questi programmi.

Per un'introduzione a come utilizzare le uscite utente, le uscite API e i servizi installabili per estendere le funzionalità del gestore code, vedere Estensione delle funzioni del gestore code.

Per informazioni sulla scrittura e la compilazione di uscite e servizi installabili, vedere “Scrittura e compilazione di uscite e servizi installabili” a pagina 376.

Concetti correlati
Programmi di uscita canale per canali MQI
Riferimenti correlati
Riferimento uscita API
Informazioni di riferimento per l'interfaccia dei servizi installabili

Scrittura e compilazione di uscite e servizi installabili

È possibile scrivere e compilare uscite senza collegarsi a librerie IBM WebSphere MQ in UNIX, Linux e Windows.

Informazioni su questa attività

Questo argomento si applica solo ai sistemi Windows, UNIX and Linux . Per dettagli sulla scrittura di uscite e servizi installabili per altre piattaforme, consultare gli argomenti specifici della piattaforma.

Se IBM WebSphere MQ è installato in un percorso non predefinito, è necessario scrivere e compilare le uscite senza collegarsi ad alcuna libreria IBM WebSphere MQ .

È possibile scrivere e compilare uscite sui sistemi Windows, UNIX and Linux senza collegare le seguenti librerie IBM WebSphere MQ :

- mqmzf
- mqm
- mqmvx
- mqmvxd
- mqic
- mqutl

Le uscite esistenti collegate a queste librerie continuano a funzionare, a condizione che sui UNIX and Linux sistemi IBM WebSphere MQ sia installato nell'ubicazione predefinita.

Procedura

1. Includere il file di intestazione cmqec.h .

   L'inclusione di questo file di intestazione include automaticamente i file di intestazione cmqc.h, cmqxc.h e cmqzc.h .
2. Scrivere l'uscita in modo che le chiamate MQI e DCI vengano effettuate tramite la struttura MQIEP. Per ulteriori informazioni sulla struttura MQIEP, consultare Struttura MQIEP.
   - Servizi installabili
     - Utilizzare il parametro **Hconfig** per puntare alla chiamata MQZEP.
     - È necessario verificare che i primi 4 byte di **Hconfig** corrispondano al **StrucId** della struttura MQIEP prima di utilizzare il parametro **Hconfig** .
     - Per ulteriori informazioni sulla scrittura dei componenti del servizio installabili, consultare MQIEP.
   - Uscite API
     - Utilizzare il parametro **Hconfig** per puntare alla chiamata MQXEP.
     - È necessario verificare che i primi 4 byte di **Hconfig** corrispondano al **StrucId** della struttura MQIEP prima di utilizzare il parametro **Hconfig** .
     - Per ulteriori informazioni sulla scrittura delle uscite API, consultare “Scrittura delle uscite API” a pagina 391.
   - Uscite canale
     - Utilizzare il parametro **pEntryPoints** della struttura MQCXP per puntare a chiamate MQI e DCI.
     - È necessario verificare che il numero di versione MQCXP sia alla versione 8 o superiore prima di utilizzare **pEntryPoints**.
     - Per ulteriori informazioni sulla scrittura delle uscite del canale, consultare “Scrittura di programmi di uscita canale” a pagina 401.
   - Uscite di conversione dati
     - Utilizzare il parametro **pEntryPoints** della struttura MQDXP per puntare a chiamate MQI e DCI.
     - È necessario verificare che il numero di versione di MQDXP sia alla versione 2 o superiore prima di utilizzare **pEntryPoints**.

- È possibile utilizzare il comandi **crtmqcvx** e il file di origine amqsvfc0.c per creare il codice di conversione dati che utilizza il parametro **pEntryPoints** . Consultare “Scrittura di un'uscita di conversione dati per WebSphere MQ per Windows” a pagina 423 e “Scrittura di un'uscita di conversione dati per WebSphere MQ su sistemi UNIX and Linux” a pagina 419.
- Se si dispone di uscite di conversione dati esistenti che sono state generate utilizzando il comando **crtmqcvx** , è necessario rigenerare l'uscita utilizzando il comando aggiornato.
- Per ulteriori informazioni sulla scrittura delle uscite di conversioni dati, consultare “Scrittura delle uscite di conversione dati” a pagina 417.

- Uscite di preconnessione
  - Utilizzare il parametro **pEntryPoints** della struttura MQNXP per puntare a chiamate MQI e DCI.
  - È necessario verificare che il numero di versione MQNXP sia alla versione 2 o superiore prima di utilizzare **pEntryPoints**.
  - Per ulteriori informazioni sulla scrittura di uscite di pre - connessione, consultare “Riferimento alle definizioni di connessione mediante un'uscita di pre - connessione da un repository” a pagina 425.
- Uscite di pubblicazione
  - Utilizzare il parametro **pEntryPoints** della struttura MQPSXP per puntare a chiamate MQI e DCI.
  - È necessario verificare che il numero di versione di MQPSXP sia alla versione 2 o superiore prima di utilizzare **pEntryPoints**.
  - Per ulteriori informazioni sulla scrittura delle uscite di pubblicazione, consultare “Scrittura e compilazione di uscite di pubblicazione” a pagina 427.
- Uscite carico di lavoro cluster
  - Utilizzare il parametro **pEntryPoints** della struttura MQWXP per puntare a chiamate MQXCLWLN.
  - È necessario verificare che il numero di versione MQWXP sia alla versione 4 o superiore prima di utilizzare **pEntryPoints**.
  - Per ulteriori informazioni sulla scrittura delle uscite del carico di lavoro del cluster, consultare “Scrittura e compilazione delle uscite del carico di lavoro del cluster” a pagina 429.

  Ad esempio, in un'uscita canale che richiama MQPUT:

```
pChannelExitParms -> pEntryPoints -> MQPUT_Call(pChannelExitParms -> Hconn,
                                                Hobj,
                                                &md,
                                                &pmo,
                                                messlen,
                                                buffer,
                                                &CompCode,
                                                &Reason);
```

  Ulteriori esempi possono essere visualizzati in “Programmi di WebSphere MQ di esempio” a pagina 97.

3. Compilare l'uscita:
   - Non collegarsi alle librerie IBM WebSphere MQ .
   - Non includere un RPath integrato in alcuna libreria IBM WebSphere MQ nella propria uscita.
   - Per ulteriori informazioni sulla compilazione dell'uscita, consultare uno dei seguenti argomenti:
     - Uscite API: “Compilazione delle uscite API” a pagina 393.
     - Uscite del canale, uscite di pubblicazione, uscite del carico di lavoro del cluster “Compilazione di programmi di uscita canale su Windows, sistemi UNIX and Linux” a pagina 416.
     - Uscite di conversione dati: “Scrittura delle uscite di conversione dati” a pagina 417.
4. Inserire l'uscita in uno dei seguenti posti:
   - Un percorso di propria scelta che si qualifica completamente quando si configura l'exit
   - Il percorso di uscita predefinito, in una specifica directory di installazione. Ad esempio, *MQ_DATA_PATH*/exits/installation2.

- Il percorso di uscita predefinito

  Il percorso di uscita predefinito è *MQ_DATA_PATH*/exits per le uscite a 32 bit e *MQ_DATA_PATH*/exits64 per le uscite a 64 bit. È possibile modificare questi percorsi nel file qm.ini o mqclient.ini . Per ulteriori informazioni, consultare Percorso di uscita. In ambiente Windows e Linux, è possibile utilizzare WebSphere MQ Explorer per modificare il percorso:

  a. Fare clic con il pulsante destro del mouse sul nome del gestore code
  b. Fare clic su **Proprietà ...**
  c. Fare clic su **Uscite**
  d. Nel campo del percorso predefinito delle uscite, specificare il percorso della directory che contiene il programma di uscita.

  Se un'uscita viene collocata sia in una directory di installazione specifica che nella directory del percorso predefinito, l'uscita della directory di installazione specifica viene utilizzata dall'installazione di WebSphere MQ indicata nel percorso. Ad esempio, l'uscita è collocata in /exits/installation2 e /exits, ma non in /exits/installation1. L'installazione di WebSphere MQ installation2 utilizza l'uscita da /exits/installation2. L' WebSphere MQ installazione installation1 utilizza l'uscita dalla directory /exits .

5. Se necessario, configurare l'uscita:
   - Servizi installabili “Configurazione di servizi e componenti” a pagina 387.
   - Uscite API: “Configurazione uscite API” a pagina 396.
   - Uscite canale: “Configurazione delle uscite canale” a pagina 417.
   - Uscite di pubblicazione: “Configurazione delle uscite di pubblicazione” a pagina 428.
   - Uscite di preconnessione: “Stanza PreConnect del file di configurazione client” a pagina 426.

Servizi e componenti installabili per Unix, Linux e Windows

Questa sezione introduce i servizi installabili e le funzioni e i componenti ad essi associati. L'interfaccia per queste funzioni è documentata in modo che l'utente, o i fornitori di software, possano fornire componenti.

I motivi principali per fornire i servizi installabili WebSphere MQ sono:

- Per fornire la flessibilità di scegliere se utilizzare i componenti forniti dai prodotti WebSphere MQ o sostituirli o aumentarli con altri.
- Per consentire ai fornitori di partecipare, fornendo componenti che potrebbero utilizzare nuove tecnologie, senza apportare modifiche interne ai prodotti WebSphere MQ .
- Per consentire a WebSphere MQ di sfruttare le nuove tecnologie in modo più rapido ed economico, fornendo i prodotti in modo più rapido e a prezzi inferiori.

I *servizi installabili* e i *componenti del servizio* sono parte della struttura del prodotto WebSphere MQ . Al centro di questa struttura si trova la parte del gestore code che implementa la funzione e le regole associate alla MQI (Message Queue Interface). Questa parte centrale richiede una serie di funzioni di servizio, denominate *servizi installabili*, per eseguire il suo lavoro. I servizi installabili sono:

- Servizio di autorizzazione
- Servizio di denominazione

Ogni servizio installabile è una serie correlata di funzioni implementate utilizzando uno o più *componenti del servizio*. Ogni componente viene richiamato utilizzando un'interfaccia correttamente strutturata e disponibile al pubblico. Ciò consente ai fornitori di software indipendenti e ad altre terze parti di fornire componenti installabili per convertire o sostituire quelli forniti dai prodotti WebSphere MQ . Tabella 52 a pagina 380 riepiloga i servizi e i componenti che possono essere utilizzati.

Tabella 52. Riepilogo dei componenti del servizio installabili			
servizio installabile	**Componente fornito**	**Funzione**	**Requisiti**
Servizio di autorizzazione	object authority manager (OAM)	Fornisce il controllo di autorizzazione sui comandi e sulle chiamate MQI. Gli utenti possono scrivere il proprio componente per aumentare o sostituire l'OAM. Ad esempio, per verificare che un ID utente disponga dell'autorità per aprire una coda.	(Sono presupponenti le funzioni di autorizzazione della piattaforma appropriate)
Servizio di denominazione	Nessuno	Fornisce supporto al gestore code per la ricerca del nome del gestore code proprietario di una coda specificata. • Definito dall'utente **Nota:** Le code condivise devono avere l'attributo *Scope* impostato su CELL.	• Un gestore di nomi di terze parti o scritti dall'utente

L'interfaccia dei servizi installabili è descritta in Informazioni di riferimento sull'interfaccia dei servizi installabili.

Scrittura di un componente del servizio

Questa sezione descrive la relazione tra servizi, componenti, punti di ingresso e codici di ritorno.

Funzioni e componenti

Ogni servizio è costituito da una serie di funzioni correlate. Ad esempio, il servizio dei nomi contiene la funzione per:

- Ricerca di un nome coda e restituzione del nome del gestore code in cui è definita la coda
- Inserimento di un nome coda nella directory del servizio
- Eliminazione di un nome coda dalla directory del servizio

Contiene anche funzioni di inizializzazione e di terminazione.

Un servizio installabile è fornito da uno o più componenti del servizio. Ogni componente può eseguire alcune o tutte le funzioni definite per quel servizio. Ad esempio, in WebSphere MQ per AIX, il componente del servizio di autorizzazione fornito, OAM, esegue tutte le funzioni disponibili. Per ulteriori informazioni, fare riferimento a “Interfaccia servizio di autorizzazione” a pagina 384. Il componente è anche responsabile della gestione di qualsiasi risorsa o software sottostante (ad esempio, una directory LDAP) di cui ha bisogno per implementare il servizio. I file di configurazione forniscono un modo standard di caricare il componente e determinare gli indirizzi delle routine funzionali che esso fornisce.

Figura 72 a pagina 381 mostra come i servizi e i componenti sono correlati:

- Un servizio viene definito per un gestore code dalle stanze in un file di configurazione.
- Ogni servizio è supportato dal codice fornito nel gestore code. Gli utenti non possono modificare questo codice e quindi non possono creare i propri servizi.
- Ogni servizio è implementato da uno o più componenti, che possono essere forniti con il prodotto o scritti dall'utente. È possibile richiamare più componenti per un servizio, ognuno dei quali supporta diverse funzioni all'interno del servizio.

- I punti di ingresso collegano i componenti di servizio al codice supportato nel gestore code.

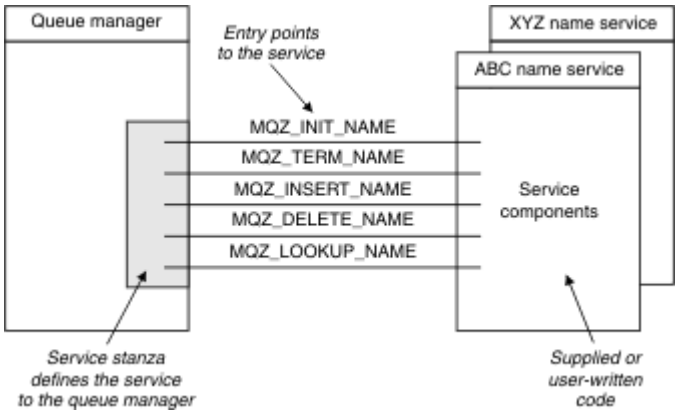


Figura 72. Informazioni su servizi, componenti e punti di ingresso

Punti di ingresso

Ciascun componente del servizio è rappresentato da un elenco di indirizzi del punto di ingresso delle routine che supportano un particolare servizio installabile. Il servizio installabile definisce la funzione che deve essere eseguita da ciascuna routine.

L'ordine dei componenti del servizio quando vengono configurati definisce l'ordine in cui i punti di ingresso vengono richiamati nel tentativo di soddisfare una richiesta per il servizio.

Nel file di intestazione fornito `cmqzc.h`, i punti di ingresso forniti per ogni servizio hanno un prefisso MQZID_.

Se i servizi sono presenti, vengono caricati in un ordine predefinito. Il seguente elenco mostra i servizi e l'ordine in cui vengono inizializzati.

1. `NameService`
2. `AuthorizationService`
3. `UserIdentifierService`

`AuthorizationService` è il unico servizio configurato per impostazione predefinita. Configurare manualmente `NameService` e `UserIdentifierService` se si desidera utilizzarli.

I servizi e i componenti del servizio hanno una mappatura uno - a - uno o uno - a - molti. È possibile definire più componenti del servizio per ciascun servizio. Sui sistemi UNIX and Linux , il valore Servizio della sezione `ServiceComponent` deve corrispondere al valore Nome della stanza del servizio nel file qm.ini . Su Windows, il valore della chiave di registro del servizio di `ServiceComponent`deve corrispondere al valore della chiave di registro del nome ed è definito come: `HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\`*qmname*`\` dove *qmname* è il nome del gestore code.

Per sistemi UNIX and Linux , i componenti del servizio vengono avviati nell'ordine in cui sono definiti nel file qm.ini . In Windows, poiché viene utilizzato il Registro di sistema di Windows, WebSphere MQ emette una chiamata **RegEnumKey** che restituisce i valori in ordine alfabetico. Pertanto, in Windows i servizi vengono richiamati in ordine alfabetico, come sono definiti nel registro.

L'ordinamento delle definizioni `ServiceComponent` è significativo. Questo ordine indica l'ordine in cui i componenti vengono eseguiti per un dato servizio. Ad esempio, `AuthorizationService` su Windows è configurato con il componente predefinito OAM denominato `MQSeries.WindowsNT.auth.service`. È possibile definire ulteriori componenti per questo servizio per sovrascrivere l'OAM predefinito. A meno che non venga specificato `MQCACF_SERVICE_COMPONENT` , il primo componente rilevato in ordine alfabetico viene utilizzato per elaborare la richiesta e viene utilizzato il relativo nome.

Codici di ritorno

I componenti del servizio forniscono codici di ritorno al gestore code per creare report su varie condizioni. Riportano l'esito positivo o negativo dell'operazione e indicano se il gestore code deve procedere al successivo componente del servizio. Un parametro *Continuazione* separato contiene questa indicazione.

Dati componente

Un singolo componente del servizio potrebbe richiedere la condivisione dei dati tra le sue varie funzioni. I servizi installabili forniscono un'area dati facoltativa da trasmettere ad ogni richiamo di un componente del servizio. Questa area dati è destinata all'uso esclusivo del componente del servizio. È condiviso da tutte le chiamate di una particolare funzione, anche se sono effettuate da diversi spazi di indirizzo o processi. È garantito che sia indirizzabile dal componente del servizio ogni volta che viene richiamato. È necessario dichiarare la dimensione di questa area nella stanza *ServiceComponent* .

Inizializzazione e terminazione dei componenti
L'utilizzo delle opzioni di inizializzazione e di chiusura del componente.

Quando viene richiamata la routine di inizializzazione del componente, deve richiamare la funzione **MQZEP** del gestore code per ogni punto di ingresso supportato dal componente. **MQZEP** definisce un punto di ingresso per il servizio. Tutti i punti di uscita non definiti vengono considerati NULL.

Un componente viene sempre richiamato una sola volta con l'opzione di inizializzazione primaria, prima di essere richiamato in qualsiasi altro modo.

Un componente può essere richiamato con l'opzione di inizializzazione secondaria su alcune piattaforme. Ad esempio, può essere richiamato una volta per ogni processo, sottoprocesso o attività del sistema operativo da cui si accede al servizio.

Se viene utilizzata l'inizializzazione secondaria:

- Il componente può essere richiamato più di una volta per l'inizializzazione secondaria. Per ciascuna di queste chiamate, viene emessa una chiamata corrispondente per la terminazione secondaria quando il servizio non è più necessario.

  Per i servizi di denominazione, questa è la chiamata MQZ_TERM_NAME.

  Per i servizi di autorizzazione questa è la chiamata MQZ_TERM_AUTHORITY.
- I punti di ingresso devono essere nuovamente specificati (richiamando MQZEP) ogni volta che il componente viene richiamato per l'inizializzazione primaria e secondaria.
- Solo una copia dei dati del componente viene utilizzata per il componente; non esiste una copia diversa per ogni inizializzazione secondaria.
- Il componente non viene richiamato per altre chiamate al servizio (dal processo del sistema operativo, dal thread o dall'attività, come appropriato) prima che sia stata eseguita l'inizializzazione secondaria.
- Il componente deve impostare il parametro *Version* sullo stesso valore per l'inizializzazione primaria e secondaria.

Il componente viene sempre richiamato con l'opzione di terminazione primaria una volta, quando non è più richiesto. Non vengono effettuate ulteriori chiamate a questo componente.

Il componente viene richiamato con l'opzione di terminazione secondaria, se è stata richiamata per l'inizializzazione secondaria.

object authority manager (OAM)
Il componente del servizio di autorizzazione fornito con i prodotti WebSphere MQ è denominato OAM (Object Authority Manager).

Per impostazione predefinita, OAM è attivo e utilizza i comandi di controllo **dspmqaut** (autorità di visualizzazione),**dmpmqaut** (autorità di dump) e **setmqaut** (autorità di impostazione o reimpostazione).

La sintassi di questi comandi e come utilizzarli sono descritti in Comandi di controllo.

OAM funziona con l' *entità* di un principal o di un gruppo.

- Su sistemi UNIX and Linux :
  - il principal è un ID utente o un ID associato ad un programma applicativo in esecuzione per conto di un utente.
  - il gruppo è una raccolta di principal definita dal sistema UNIX o Linux .
  - Le autorizzazioni possono essere concesse o revocate solo a livello di gruppo. Una richiesta di concessione o revoca dell'autorizzazione di un utente aggiorna il gruppo principale per tale utente.
- Su sistemi Windows:
  - il principal è un ID utente Windows o un ID associato ad un programma applicativo in esecuzione per conto di un utente.
  - il gruppo è un gruppo Windows.
  - Le autorizzazioni possono essere concesse o revocate a livello di principal o di gruppo.

Quando viene effettuata una richiesta MQI o viene emesso un comando, OAM verifica l'autorizzazione dell'entità associata all'operazione per verificare se è possibile:

- Eseguire l'operazione richiesta.
- Accedere alle risorse del gestore code specificate.

Il servizio di autorizzazione consente di convertire o sostituire il controllo dell'autorizzazione fornito per i gestori code scrivendo il componente del servizio di autorizzazione.

Servizio di denominazione
Il servizio dei nomi è un servizio installabile che fornisce supporto al gestore code per la ricerca del nome del gestore code che possiede una coda specificata. Nessun altro attributo della coda può essere richiamato da un servizio nomi.

Il servizio dei nomi consente a un'applicazione di aprire le code remote per l'emissione come se fossero code locali. Un servizio nomi non viene richiamato per oggetti diversi dalle code.

Nota: Le code remote ***devono*** avere il proprio attributo *Scope* impostato su CELL.

Quando un'applicazione apre una coda, cerca il nome della coda nella directory del gestore code. Se non lo trova, cerca in tutti i servizi dei nomi che sono stati configurati, fino a quando non ne trova uno che riconosca il nome della coda. Se nessuno riconosce il nome, l'apertura ha esito negativo.

Il servizio dei nomi restituisce il gestore code proprietario per tale coda. Il gestore code continua quindi con la richiesta MQOPEN come se il comando avesse specificato il nome della coda e del gestore code nella richiesta originale.

L'NSI (name service interface) fa parte del framework WebSphere MQ .

Come funziona il servizio dei nomi

Se una definizione della coda specifica l'attributo *Scope* come gestore code, ossia SCOPE (QMGR) in MQSC, la definizione della coda (insieme a tutti gli attributi della coda) viene memorizzata solo nella directory del gestore code. Non può essere sostituito da un servizio installabile.

Se una definizione della coda specifica l'attributo *Scope* come cella, cioè SCOPE (CELL) in MQSC, la definizione della coda viene nuovamente memorizzata nella directory del gestore code, insieme a tutti gli attributi della coda. Tuttavia, il nome della coda e del gestore code vengono memorizzati anche in un servizio nomi. Se non è disponibile alcun servizio che possa memorizzare queste informazioni, non è possibile definire una coda con la cella *Scope* .

La directory in cui sono memorizzate le informazioni può essere gestita dal servizio oppure il servizio può utilizzare un servizio sottostante, ad esempio una directory LDAP, per questo scopo. In entrambi i casi, le definizioni memorizzate nella directory devono persistere, anche dopo che il componente e il gestore code sono stati terminati, fino a quando non vengono esplicitamente eliminati.

Nota:

1. Per inviare un messaggio alla definizione della coda locale di un host remoto (con un ambito CELL) su un gestore code differente all'interno di una cella di directory di denominazione, è necessario definire un canale.
2. Non è possibile ottenere i messaggi direttamente dalla coda remota, anche quando ha un ambito CELL.
3. Non è richiesta alcuna definizione di coda remota durante l'invio a una coda con ambito CELL.
4. Il servizio di denominazione definisce centralmente la coda di destinazione, anche se è ancora necessaria una coda di trasmissione per il gestore code di destinazione e una coppia di definizioni di canale. Inoltre, la coda di trasmissione sul sistema locale deve avere lo stesso nome del gestore code che possiede la coda di destinazione, con l'ambito della cella, sul sistema remoto.

   Ad esempio, se il gestore code remoto ha il nome QM01, la coda di trasmissione sul sistema locale deve avere anche il nome QM01.

Interfaccia servizio di autorizzazione
Il servizio di autorizzazione fornisce punti di ingresso per l'utilizzo da parte del gestore code.

I punti di ingresso sono i seguenti:

UTENTE_AUTENTICA_MQZ
Autentica un ID utente e una parola d'ordine e può impostare campi di contesto di identità.

MQZ_CHECK_AUTHORITY
Verifica se un'entità dispone dell'autorizzazione per eseguire una o più operazioni su un oggetto specificato.

MQZ_CHECK_PRIVILEGED
Verifica se un utente specificato è un utente privilegiato.

MQZ_COPY_ALL_AUTHORITY
Copia tutte le autorizzazioni correnti esistenti per un oggetto di riferimento in un altro oggetto.

MQZ_DELETE_AUTHORITY
Elimina tutte le autorizzazioni associate ad un oggetto specificato.

MQZ_ENUMERATE_AUTHORITY_DATA
Richiama tutti i dati di autorizzazione che corrispondono ai criteri di scelta specificati.

MQZ_FREE_UTENTE
Libera le risorse assegnate associate.

AUTORA_GET_MQZ
Ottiene l'autorità di cui dispone un'entità per accedere a un oggetto specificato.

MQZ_GET_EXPLICIT_AUTHORITY
Ottiene l'autorizzazione che un gruppo denominato ha per accedere a un oggetto specificato (ma senza l'autorizzazione aggiuntiva del gruppo **nobody**) o l'autorità che il gruppo primario del principal denominato ha per accedere a un oggetto specificato.

MQZ_INIT_AUTHORITY
Inizializza il componente del servizio di autorizzazione.

INQUIRE MQZ
Interroga la funzione supportata del servizio di autorizzazione.

MQZ_REFRESH_CACHE
Aggiornare tutte le autorizzazioni.

AUTORIZZAZIONE IMPOSTAZIONE MQZ
Imposta l'autorizzazione che un'entità ha su un oggetto specificato.

AUTORIZZAZIONE TERM MQZ
Termina il componente del servizio di autorizzazione.

Inoltre, su WebSphere MQ per Windows, il servizio di autorizzazione fornisce i seguenti punti di ingresso che possono essere utilizzati dal gestore code:

- **MQZ_CHECK_AUTHORITY_2**
- **MQZ_GET_AUTHORITY_2**

- **MQZ_GET_EXPLICIT_AUTHORITY_2**
- **MQZ_SET_AUTHORITY_2**

Questi punti di ingresso supportano l'utilizzo del SID NT (Windows Security Identifier).

Questi nomi sono definiti come **typedef**, nel file di intestazione `cmqzc.h`, che può essere utilizzato per creare il prototipo delle funzioni del componente.

La funzione di inizializzazione (**MQZ_INIT_AUTHORITY**) deve essere il punto di ingresso principale per il componente. Le altre funzioni vengono invocate tramite l'indirizzo del punto di entrata che la funzione di inizializzazione ha aggiunto al vettore del punto di ingresso del componente.

Interfaccia servizio nomi
Un servizio nomi fornisce i punti di ingresso per l'utilizzo da parte del gestore code.

Vengono forniti i seguenti punti di ingresso:

NOME_INIT_MQZ
Inizializza il componente servizio nomi.

NOME MQZ_TERM
Terminare il componente servizio nomi.

MQZ_NOME_RICERCA
Ricercare il nome del gestore code per la coda specificata.

NOME MQZ_INSERT
Inserire una voce contenente il nome del gestore code proprietario per la coda specificata nella directory utilizzata dal servizio.

NOME_ELIMINAZIONE_MQZ
Eliminare la voce per la coda specificata dalla directory utilizzata dal servizio.

Se è configurato più di un servizio nomi:

- Per la ricerca, la funzione MQZ_LOOKUP_NAME viene richiamata per ogni servizio nell'elenco fino a quando il nome della coda non viene risolto (a meno che un componente non indichi che la ricerca deve essere arrestata).
- Per inserimento, la funzione MQZ_INSERT_NAME viene richiamata per il primo servizio nell'elenco che supporta questa funzione.
- Per l'eliminazione, viene richiamata la funzione MQZ_DELETE_NAME per il primo servizio nell'elenco che supporta questa funzione.

Non dispone di più di un componente che supporta le funzioni di inserimento ed eliminazione. Tuttavia, un componente che supporta solo la ricerca è fattibile e può essere utilizzato, ad esempio, come ultimo componente dell'elenco per risolvere qualsiasi nome che non sia noto a nessun altro componente del servizio nomi in un gestore code in cui è possibile definire il nome.

Nel linguaggio di programmazione C i nomi sono definiti come tipi di dati della funzione utilizzando l'istruzione typedef. Questi possono essere utilizzati per eseguire il prototipo delle funzioni di manutenzione, per garantire che i parametri siano corretti.

Il file di intestazione che contiene tutto il materiale specifico per i servizi installabili è `cmqzc.h` per il linguaggio C.

A parte la funzione di inizializzazione (MQZ_INIT_NAME), che deve essere il punto di ingresso principale del componente, le funzioni vengono richiamate dall'indirizzo del punto di ingresso aggiunto dalla funzione di inizializzazione, utilizzando la chiamata MQZEP.

Utilizzo di più componenti del servizio
È possibile installare più di un componente per un servizio. Ciò consente ai componenti di fornire solo implementazioni parziali del servizio e di fare affidamento su altri componenti per fornire le funzioni rimanenti.

Esempio di utilizzo di più componenti

Si supponga di creare due componenti dei servizi dei nomi denominati ABC_name_serv e XYZ_name_serv.

ABC_name_serv
Questo componente supporta l'inserimento o l'eliminazione di un nome dalla directory di servizio, ma non supporta la ricerca di un nome coda.

XYZ_name_serv
Questo componente supporta la ricerca di un nome coda, ma non supporta l'inserimento di un nome nella directory di servizio o l'eliminazione di un nome dalla directory di servizio.

Il componente ABC_name_serv contiene un database di nomi coda e utilizza due semplici algoritmi per inserire o eliminare un nome dalla directory di servizio.

Il componente XYZ_name_serv utilizza un algoritmo semplice che restituisce un nome gestore code fisso per qualsiasi nome coda con cui viene richiamato. Non contiene un database di nomi coda e quindi non supporta le funzioni di inserimento e di eliminazione.

I componenti vengono installati sullo stesso gestore code. Le stanze *ServiceComponent* vengono ordinate in modo che il componente ABC_name_serv venga richiamato per primo. Le chiamate per inserire o eliminare una coda in una directory del componente sono gestite dal componente ABC_name_serv; è l'unico che implementa queste funzioni. Tuttavia, una chiamata di ricerca che il componente ABC_name_serv non può risolvere viene passata al componente di sola ricerca, XYZ_name_serv. Questo componente fornisce un nome gestore code dal suo algoritmo semplice.

Omissione dei punti di ingresso quando si utilizzano più componenti

Se si decide di utilizzare più componenti per fornire un servizio, è possibile progettare un componente del servizio che non implementa determinate funzioni. Il framework dei servizi installabili non pone alcuna limitazione su cui è possibile omettere. Tuttavia, per servizi installabili specifici, l'omissione di una o più funzioni potrebbe essere logicamente incoerente con lo scopo del servizio.

Esempio di punti di ingresso utilizzati con più componenti

Tabella 53 a pagina 386 mostra un esempio del servizio nomi installabile per cui sono stati installati i due componenti. Ognuno supporta una serie diversa di funzioni associate a questo particolare servizio installabile. Per la funzione di inserimento, viene richiamato prima il punto di ingresso del componente ABC. I punti di immissione che non sono stati definiti per il servizio (utilizzando **MQZEP**) vengono considerati NULL. Nella tabella viene fornito un punto di ingresso per l'inizializzazione, ma questo non è richiesto perché l'inizializzazione viene eseguita dal punto di ingresso principale del componente.

Quando il gestore code deve utilizzare un servizio installabile, utilizza i punti di ingresso definiti per tale servizio (le colonne in Tabella 53 a pagina 386). Prendendo ogni componente a turno, il gestore code determina l'indirizzo della routine che implementa la funzione richiesta. Richiama quindi la routine, se esiste. Se l'operazione ha esito positivo, i risultati e le informazioni sullo stato vengono utilizzati dal gestore code.

Tabella 53. Esempio di punti di ingresso per un servizio installabile

Numero funzione	Componente servizio nomi ABC	Componente servizio nomi XYZ
MQZID_INIT_NAME (Inizializza)	Inizializza ABC ()	XYZ_inizializza ()
MQZID_TERM_NAME (Termina)	Terminazione ABC ()	XYZ_termina ()
MQZID_INSERT_NAME (Inserimento)	Inseriment_ABC ()	NULL
MQZID_DELETE_NAME (Elimina)	Eliminazione ABC ()	NULL
MQZID_LOOKUP_NAME (Ricerca)	NULL	Ricerca XYZ ()

Se la routine non esiste, il gestore code ripete questo processo per il componente successivo nell'elenco. Inoltre, se la routine esiste ma restituisce un codice che indica che non è stato possibile eseguire l'operazione, il tentativo continua con il successivo componente disponibile. Le routine nei componenti del servizio potrebbero restituire un codice che indica che non devono essere effettuati ulteriori tentativi di esecuzione dell'operazione.

Configurazione di servizi e componenti

Configurare i componenti del servizio utilizzando i file di configurazione del gestore code, ad eccezione dei sistemi Windows , in cui ciascun gestore code ha la propria stanza nel registro.

1. Aggiungere le stanze al file di configurazione del gestore code per definire il servizio per il gestore code e specificare l'ubicazione del modulo.

   Ogni servizio utilizzato deve avere una sezione *Service* , che definisce il servizio per il gestore code.

   Per ogni componente in un servizio, deve essere presente una stanza *ServiceComponent* . Questo identifica il nome e il percorso del modulo contenente il codice per tale componente.

   Per ulteriori informazioni, consultare “Formato stanza di servizio” a pagina 387 e “Formato stanza componente servizio” a pagina 388

   Il componente servizio di autorizzazione, noto come OAM (Object Authority Manager), viene fornito con il prodotto. Quando si crea un gestore code, il file di configurazione del gestore code (o il registro sui sistemi Windows) viene aggiornato automaticamente in modo da includere le stanze appropriate per il servizio di autorizzazione e per il componente predefinito (OAM). Per gli altri componenti, è necessario configurare il file di configurazione del gestore code manualmente.

   Il codice per ciascun componente del servizio viene caricato nel gestore code quando il gestore code viene avviato, utilizzando il bind dinamico, dove è supportato sulla piattaforma.
2. Arrestare e riavviare il gestore code per attivare il componente.

Formato stanza di servizio

La stanza Service contiene il nome del servizio e il numero di punti di ingresso definiti per il servizio.

Il formato della stanza è il seguente:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
```

dove:

<service_name>
Il nome del servizio. Questo è definito dal servizio.

<entries>
Il numero di punti di ingresso definiti per il servizio. Ciò include i punti di ingresso di inizializzazione e terminazione.

Formato stanza di servizio per i sistemi Windows

Sui sistemi Windows , la stanza *Service* include un attributo SecurityPolicy .

Il formato della stanza è:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
  SecurityPolicy=<policy>
```

dove:

<service_name>
Il nome del servizio. Questo è definito dal servizio.

<entries>
Il numero di punti di ingresso definiti per il servizio. Ciò include i punti di ingresso di inizializzazione e terminazione.

<policy>
NTSIDsRequired (l'identificativo di protezione Windows) o Default. Se non si specifica NTSIDsRequired, viene utilizzato il valore Default . Questo attributo è valido solo se Name ha il valore AuthorizationService.

Vedi anche “Configurazione delle stanze del servizio di autorizzazione: sistemi Windows” a pagina 389.

Formato stanza componente servizio

Il formato della stanza del componente del servizio è:

```
ServiceComponent:
  Service=<service_name>
  Name=<component_name>
  Module=<module_name>
  ComponentDataSize=<size>
```

dove:

<service_name>
Il nome del servizio. Deve corrispondere al Name specificato in una stanza di servizio.

<component_name>
Un nome descrittivo del componente del servizio. Deve essere univoco e contenere solo i caratteri validi per i nomi degli oggetti WebSphere MQ (ad esempio, i nomi coda). Questo nome si verifica nei messaggi dell'operatore generati dal servizio. Si consiglia di utilizzare un nome che inizia con un marchio aziendale o una stringa di distinzione simile.

<module_name>
Il nome del modulo che deve contenere il codice per questo componente.

<size>
La dimensione, in byte, dell'area dati del componente passata al componente su ogni chiamata. Specificare zero se non sono richiesti dati del componente.

Queste due stanze possono verificarsi in qualsiasi ordine e le chiavi della stanza sotto di esse possono anche verificarsi in qualsiasi ordine. Per una di queste stanze, tutte le chiavi della stanza devono essere presenti. Se una chiave di stanza è duplicata, viene utilizzata l'ultima.

All'avvio, il gestore code elabora ciascuna voce del componente del servizio nel file di configurazione a turno. Carica quindi il modulo del componente specificato, richiamando il punto di ingresso del componente (che deve essere il punto di entrata per l'inizializzazione del componente), passando un handle di configurazione.

Configurazione delle stanze del servizio di autorizzazione: sistemi UNIX and Linux
Sui sistemi UNIX and Linux , ogni gestore code ha il proprio file di configurazione del gestore code.

Ad esempio, il percorso predefinito e il nome file del file di configurazione del gestore code per il gestore code QMNAME è /var/mqm/qmgrs/QMNAME/qm.ini.

La stanza *Service* e la sezione *ServiceComponent* per il componente di autorizzazione predefinito vengono aggiunte automaticamente a qm.ini , ma possono essere sovrascritte da mqsnoaut. Qualsiasi altra stanza *ServiceComponent* deve essere aggiunta manualmente.

Ad esempio, le seguenti stanze nel file di configurazione del gestore code definiscono due componenti del servizio autorizzazione su WebSphere MQ per AIX. *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato 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
```

Figura 73. UNIX and Linux stanze del servizio di autorizzazione in qm.ini

La stanza del componente servizio (`MQSeries.UNIX.auth.service`) definisce il componente del servizio di autorizzazione predefinito, OAM. Se si rimuove questa stanza e si riavvia il gestore code, l'OAM viene disabilitato e non viene eseguito alcun controllo di autorizzazione.

Configurazione delle stanze del servizio di autorizzazione: sistemi Windows
Su WebSphere MQ per Windows , ogni gestore code ha la sua stanza nel registro.

La stanza *`Service`* e la stanza *`ServiceComponent`* per il componente di autorizzazione predefinito vengono aggiunte automaticamente al registro, ma possono essere sovrascritte utilizzando `mqsnoaut`. Qualsiasi altra stanza *`ServiceComponent`* deve essere aggiunta manualmente.

È anche possibile aggiungere l'attributo `SecurityPolicy` utilizzando i servizi WebSphere MQ . L'attributo `SsecurityPolicy` si applica solo se il servizio specificato nella stanza *`Service`* è il servizio di autorizzazione, ossia l'OAM predefinito. L'attributo `SecurityPolicy` consente di specificare la politica di sicurezza per ciascun gestore code. I valori possibili sono:

`Default`
Specificare `Default` se si desidera che la politica di sicurezza predefinita abbia effetto. Se un identificativo di sicurezza Windows (SID NT) non viene passato a OAM per un particolare ID utente, viene effettuato un tentativo di ottenere il SID appropriato ricercando i database di sicurezza pertinenti.

`NTSIDsRequired`
Richiede che un SID NT venga passato a OAM quando si eseguono i controlli di sicurezza.

Per informazioni sul formato della stanza Service, consultare “Formato stanza di servizio per i sistemi Windows” a pagina 387. Per informazioni più generali sulla sicurezza, consultare Impostazione della sicurezza su sistemi Windows, UNIX and Linux.

La stanza del componente del servizio, `MQSeries.WindowsNT.auth.service` definisce il componente del servizio di autorizzazione predefinito, OAM. Se si rimuove questa stanza e si riavvia il gestore code, l'OAM viene disabilitato e non viene eseguito alcun controllo di autorizzazione.

Configurazione delle stanze del servizio nomi: sistemi Unix e Linux
Inserire qui una breve descrizione; utilizzata per il primo paragrafo e la sintesi.

I seguenti esempi di stanze del file di configurazione UNIX and Linux per il servizio nomi specificano un componente servizio nomi fornito dalla società ABC (fittizia).

```
# 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
```

Figura 74. Stanze del servizio dei nomi in qm.ini (per sistemi UNIX and Linux)

Nota: Su sistemi Windows , le informazioni sulla sezione del servizio nomi vengono memorizzate nel registro.

Aggiornamento di OAM dopo la modifica dell'autorizzazione di un utente
In WebSphere MQ, è possibile aggiornare le informazioni sul gruppo di autorizzazione di OAM immediatamente dopo aver modificato l'appartenenza del gruppo di autorizzazione di un utente, riflettendo le modifiche apportate a livello di sistema operativo, senza dover arrestare e riavviare il gestore code. Per fare ciò, immettere il comando **REFRESH SECURITY** .

Nota: Quando si modificano le autorizzazioni con il comando `setmqaut` , OAM implementa immediatamente tali modifiche.

I gestori code memorizzano i dati di autorizzazione su una coda locale denominata SYSTEM.AUTH.DATA.QUEUE. Questi dati sono gestiti da `amqzfuma.exe` .

Riferimenti correlati
Aggiorna sicurezza

Scrittura e compilazione delle uscite API

Le uscite API consentono di scrivere il codice che modifica il comportamento delle chiamate API WebSphere MQ , come MQPUT e MQGET, e quindi inserire tale codice immediatamente prima o immediatamente dopo tali chiamate.

Nota: Non supportato su WebSphere MQ per z/OS.

Perché utilizzare le uscite API?

Ciascuna delle applicazioni ha un lavoro specifico da eseguire e il relativo codice dovrebbe eseguire tale attività nel modo più efficiente possibile. Ad un livello superiore, è possibile che si desideri applicare standard o processi di business ad un particolare gestore code per **tutte** le applicazioni che utilizzano tale gestore code. È più efficiente eseguire questa operazione al di sopra del livello delle singole applicazioni, e quindi senza dover modificare il codice di ciascuna applicazione interessata.

Di seguito sono riportati alcuni suggerimenti di aree in cui le uscite API potrebbero essere utili:

- Per la *sicurezza*, è possibile fornire l'autenticazione, controllando che le applicazioni siano autorizzate ad accedere a una coda o a un gestore code. È anche possibile controllare l'utilizzo dell'API da parte delle applicazioni, autenticando le singole chiamate API o anche i relativi parametri.
- Per *flessibilità*, è possibile rispondere a rapide modifiche nel proprio ambiente di business senza modificare le applicazioni che si basano sui dati in tale ambiente. È possibile, ad esempio, disporre di uscite API che rispondono alle variazioni dei tassi di interesse, dei tassi di cambio o del prezzo dei componenti in un ambiente di produzione.
- Per l'utilizzo *di monitoraggio* di una coda o di un gestore code, è possibile tenere traccia del flusso di applicazioni e messaggi, registrare gli errori nelle chiamate API, configurare le tracce di controllo per scopi di account o raccogliere statistiche di utilizzo per scopi di pianificazione.

Cosa accade quando viene eseguita un'uscita API?

Dopo aver scritto un programma di uscita e averlo identificato in WebSphere MQ, il gestore code richiama automaticamente il codice di uscita nei punti registrati.

Le routine API exit da eseguire sono identificate nelle stanze sui sistemi IBM i, Windows, UNIX and Linux . Questo argomento descrive le stanze nei file di configurazione mqs.ini e qm.ini.

La definizione delle routine può verificarsi in tre posizioni:

1. ApiExitCommon, nel file mqs.ini , identifica le routine, per l'intero WebSphere MQ, applicate all'avvio dei gestori code. Questi possono essere sovrascritti dalle routine definite per i singoli gestori code (vedere l'elemento “3” a pagina 391 in questo elenco).
2. Il modello ApiExit, nel file mqs.ini , identifica le routine per l'intero WebSphere MQ, copiato nel set locale ApiExit(vedere la voce “3” a pagina 391 in questo elenco) quando viene creato un nuovo gestore code.
3. ApiExitlocale, nel file qm.ini , identifica le routine che si applicano a uno specifico gestore code.

Quando viene creato un nuovo gestore code, le definizioni di modello ApiExitin mqs.ini vengono copiate in ApiExitDefinizioni locali in qm.ini per il nuovo gestore code. Quando un gestore code viene avviato, vengono utilizzate sia le definizioni ApiExitCommon che ApiExitLocal. Le definizioni locali ApiExitsostituiscono le definizioni comuni ApiExitse entrambe identificano una routine con lo stesso nome. L'attributo `Sequence` , descritto in “Configurazione uscite API” a pagina 396 determina l'ordine di esecuzione delle routine definite nelle stanze.

Utilizzo delle uscite API su più installazioni di WebSphere MQ

Assicurarsi che le uscite API scritte per la versione precedente di WebSphere MQ siano utilizzate per gestire tutte le versioni poiché le modifiche apportate alle uscite nella versione 7.1 potrebbero non funzionare con una versione precedente. Per ulteriori informazioni sulle modifiche apportate alle uscite, consultare “Scrittura e compilazione di uscite e servizi installabili” a pagina 376.

Gli esempi forniti per le uscite API amqsaem e amqsaxe riflettono le modifiche richieste durante la scrittura delle uscite. L'applicazione client deve garantire che le librerie di WebSphere MQ corrette, che corrispondono all'installazione del gestore code a cui è associata l'applicazione, siano collegate ad essa prima dell'avvio dell'applicazione.

Scrittura delle uscite API

È possibile scrivere uscite per ogni chiamata API utilizzando il linguaggio di programmazione C.

Le uscite sono disponibili per ogni chiamata API, come segue:

- MQCB, per registrare nuovamente un callback per l'handle dell'oggetto specificato e controllare l'attivazione e le modifiche al callback
- MQCTL, per eseguire azioni di controllo sugli handle di oggetto aperti per una connessione
- MQCONN/MQCONNX, per fornire un handle di connessione del gestore code da utilizzare nelle chiamate API successive
- MQDISC, per disconnettersi da un gestore code
- MQBEGIN, per iniziare un'unità di lavoro globale (UOW)
- MQBACK, per eseguire il backout di una UOW
- MQCMIT, per eseguire il commit di una UOW
- MQOPEN, per aprire una risorsa di WebSphere MQ per un accesso successivo
- MQCLOSE, per chiudere una risorsa WebSphere MQ precedentemente aperta per l'accesso
- MQGET, per richiamare un messaggio da una coda precedentemente aperta per l'accesso
- MQPUT1, per inserire un messaggio in una coda
- MQPUT, per inserire un messaggio in una coda precedentemente aperta per l'accesso

- MQINQ, per richiedere informazioni sugli attributi di una risorsa WebSphere MQ che è stata precedentemente aperta per l'accesso
- MQSET, per impostare gli attributi di una coda che è stata precedentemente aperta per l'accesso
- MQSTAT, per recuperare le informazioni sullo stato
- MQSUB, per registrare la sottoscrizione delle applicazioni a un particolare argomento
- MQSUBRQ, per effettuare una richiesta di sottoscrizione

MQ_CALLBACK_EXIT fornisce una funzione di uscita da eseguire prima e dopo l'elaborazione del callback. Per ulteriori informazioni, consultare Callback - MQ_CALLBACK_EXIT.

All'interno delle uscite API, le chiamate assumono il seguente formato generale:

```
MQ_call_EXIT (parameters, context, ApiCallParameters)
```

dove *call* è il nome della chiamata MQI senza il prefisso MQ ; ad esempio, PUT, GET. Il *parameters* controlla la funzione dell'uscita, fornendo principalmente la comunicazione tra l'uscita e i blocchi di controllo esterni MQAXP (la struttura del parametro dell'uscita API) e MQAXC (la struttura del contesto dell'uscita API). *context* descrive il contesto in cui è stata richiamata l'uscita API e *ApiCallParameters* rappresenta i parametri per la chiamata MQI.

Per aiutarti a scrivere la tua uscita API, viene fornita un'uscita di esempio, amqsaxe0.c; questa uscita genera voci di traccia in un file che specifichi. È possibile utilizzare questo esempio come punto di partenza quando si scrivono le uscite. Per ulteriori informazioni sull'utilizzo dell'uscita di esempio, consultare “Il programma di esempio di uscita API” a pagina 114.

Per ulteriori informazioni sulle chiamate API exit, sui blocchi di controllo esterni e sugli argomenti associati, vedi Guida di riferimento API exit.

Per informazioni generali su come scrivere, compilare e configurare un'uscita, vedere “Scrittura e compilazione di uscite e servizi installabili” a pagina 376.

Utilizzo degli handle dei messaggi nelle uscite API

È possibile controllare le proprietà del messaggio a cui ha accesso un'uscita API. Le proprietà sono associate a un handle ExitMsg. Le proprietà impostate in un'uscita di inserimento sono impostati sul messaggio da inserire, ma le proprietà richiamate in un'uscita di acquisizione non vengono restituite all'applicazione.

Quando si registra una funzione di uscita MQ_INIT_EXIT utilizzando la chiamata MQXEP MQI con **Function** impostato su MQXF_INIT e **ExitReason** impostato su MQXR_CONNECTION, si passa una struttura MQXEPO come parametro **ExitOpts** . La struttura MQXEPO contiene il campo ExitProperties , che specifica la serie di proprietà da rendere disponibili all'uscita. Viene specificato come una stringa di caratteri che rappresenta il prefisso delle proprietà, che corrisponde al nome di una cartella MQRFH2 .

Ogni uscita API riceve una struttura MQAXP, contenente un campo Handle ExitMsg. Questo campo è impostato su un valore generato da WebSphere MQ ed è specifico per una connessione. L'handle non viene quindi modificato tra le uscite API dello stesso tipo o di tipi differenti sulla stessa connessione.

In MQ_PUT_EXIT o MQ_PUT1_EXIT con un **ExitReason** di MQXR_BEFORE, ossia un'uscita API eseguita prima di inserire un messaggio, tutte le proprietà (diverse dalle proprietà del descrittore del messaggio) associate all'handle ExitMsgquando l'uscita viene completata vengono impostate sul messaggio da inserire. Per evitare ciò, impostare ExitMsgHandle su MQHM_NONE. È anche possibile fornire un handle del messaggio differente.

In un MQ_GET_EXIT, l'handle ExitMsgviene eliminato dalle proprietà e popolato con le proprietà specificate nel campo ExitProperties quando è stato registrato MQ_INIT_EXIT, diverse dalle proprietà del descrittore del messaggio. Queste proprietà non sono rese disponibili per l'applicazione di acquisizione. Se l'applicazione di richiamo ha specificato una gestione del messaggio nel campo MQGMO (Get message options), tutte le proprietà associate a tale gestione, incluse le proprietà del descrittore del messaggio, sono disponibili per l'uscita API. Per impedire che l'handle ExitMsgvenga popolato con le proprietà, impostarlo su MQHM_NONE.

Un programma di esempio, amqsaem0.c, viene fornito per illustrare l'utilizzo degli handle dei messaggi nelle uscite API.

Compilazione delle uscite API

Dopo aver scritto un'uscita, compilarla e collegarla nel modo seguente.

I seguenti esempi mostrano i comandi utilizzati per il programma di esempio descritto in "Il programma di esempio di uscita API" a pagina 114. Per piattaforme diverse dai sistemi Windows , è possibile trovare il codice di uscita API di esempio in *MQ_INSTALLATION_PATH*/samp e la libreria condivisa compilata e collegata in *MQ_INSTALLATION_PATH*/samp/bin. Per i sistemi Windows , è possibile trovare il codice di uscita API di esempio in *MQ_INSTALLATION_PATH*\Tools\c\Samples. *MQ_INSTALLATION_PATH* rappresenta la directory in cui è stato installato WebSphere MQ .

Nota per gli utenti:

1. La guida sulla programmazione di applicazioni a 64 bit è elencata in Standard di codifica su piattaforme a 64 bit

Con l'introduzione dei client multicast, le uscite API e le uscite di conversione dati devono essere in grado di essere eseguite sul lato client perché alcuni messaggi potrebbero non passare attraverso il gestore code. Le librerie riportate di seguito fanno ora parte dei pacchetti client e dei pacchetti server:

Tabella 54. Librerie che si trovano ora nei package client e server

Sistema operativo	**Librerie**
Finestre	32 bit & 64 bit: mqm.dll & mqm.pdb
Linux & HP-UX	32 bit & 64 bit: libmqm.so & libmqm_r.so
AIX	32 bit & 64 bit: libmqm.a & libmqm_r.a
Solaris	32 bit & 64 bit: libmqm.so

Compilazione delle uscite API su sistemi UNIX e Linux

Esempi di come compilare le uscite API su sistemi UNIX e Linux .

Su tutte le piattaforme, il punto di accesso al modulo è MQStart.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Su AIX

Compilare il codice origine dell'uscita API immettendo uno dei seguenti comandi:

Applicazioni a 32 bit

Senza thread

```
cc -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Threaded

```
xlc_r -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe_r \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Applicazioni a 64 bit

Senza thread

```
cc -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Threaded

```
xlc_r -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe_r \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Su piattaforma HP-UX Itanium

Applicazioni a 32 bit

Senza thread

Compilare il codice di origine dell'uscita API:

```
c89 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Collega il codice di origine dell'uscita API

```
ld +b: -b amqsaxe.o +ee MQStart -o /var/mqm/exits/amqsaxe
rm amqsaxe.o
```

Threaded

Compilare il codice di origine dell'uscita API:

```
c89 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Collega il codice di origine dell'uscita API

```
ld +b: -b amqsaxe.o +ee MQStart -o /var/mqm/exits/amqsaxe_r
rm amqsaxe.o
```

Applicazioni a 64 bit

Senza thread

Compilare il codice di origine dell'uscita API:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Collega il codice di origine dell'uscita API

```
ld -b amqsaxe.o +ee MQStart -o  /var/mqm/exits64/amqsaxe
rm amqsaxe.o
```

Threaded

Compilare il codice di origine dell'uscita API:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Collega il codice di origine dell'uscita API

```
ld -b amqsaxe.o +ee MQStart -o  /var/mqm/exits64/amqsaxe_r
rm amqsaxe.o
```

SULinux

Compilare il codice origine dell'uscita API immettendo uno dei seguenti comandi:

Applicazioni a 31 bit

Senza thread

```
 gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Threaded

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -IMQ_INSTALLATION_PATH/inc
```

Applicazioni a 32 bit

Senza thread

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Threaded

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -IMQ_INSTALLATION_PATH/inc
```

Applicazioni a 64 bit

Senza thread

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Threaded

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe_r amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Su Solaris

Compilare il codice origine dell'uscita API immettendo uno dei seguenti comandi:

Applicazioni a 32 bit

Piattaforma 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
```

Piattaforma x86-64

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/32 -lsocket -lnsl -ldl
```

Applicazioni a 64 bit

Piattaforma 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
```

Piattaforma x86-64

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/64 -lsocket -lnsl -ldl
```

Su sistemi Windows

Compilare e collegare il programma di uscita API di esempio, `amqsaxe0.c`, su Windows

Un file manifest è un documento XML facoltativo contenente la versione o qualsiasi altra informazione che può essere incorporata in un'applicazione compilata o DLL.

Se non hai tale documento, ometti il parametro -manifest *manifest.file* nel comando **mt** .

Adattare i comandi negli esempi in Figura 75 a pagina 396 o Figura 76 a pagina 396 per compilare e collegare `amqsaxe0.c` su Windows . I comandi funzionano con Microsoft Visual Studio 2005, 2008 o 2010. Gli esempi presuppongono che la directory WebSphere MQ `C:\Program Files\IBM\WebSphere MQ\tools\c\samples` sia la directory corrente.

32 bit

```
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
```

Figura 75. Compila e collega amqsaxe0.c su Windows a 32 bit

64 bit

```
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
```

Figura 76. Compilare e collegare amqsaxe0.c su Windows a 64 bit

Concetti correlati

"Il programma di esempio di uscita API" a pagina 114
L'uscita API di esempio genera una traccia MQI per un file specificato dall'utente con un prefisso definito nella variabile di ambiente MQAPI_TRACE.

Configurazione uscite API

Configurare IBM WebSphere MQ per abilitare le uscite API modificando le informazioni di configurazione.

Per modificare le informazioni di configurazione, è necessario modificare le stanze che definiscono le routine di uscita e la sequenza in cui vengono eseguite. Queste informazioni possono essere modificate nei seguenti modi:

- Utilizzo di IBM WebSphere MQ Explorer (su Windows e Linux (piattaformex86 e x86-64))
- Utilizzo del comando **amqmdain** (Su Windows)
- Utilizzando direttamente i file mqs.ini e qm.ini (su Windows, UNIX and Linux).

  Il file mqs.ini contiene informazioni relative a tutti i gestori code su un determinato nodo. È possibile trovarlo nella directory `/var/mqm` su UNIX and Linux e in WorkPath specificato nella chiave `HKLM\SOFTWARE\IBM\WebSphere MQ` sui sistemi Windows .

  Il file qm.ini contiene informazioni relative a un determinato gestore code. È presente un file di configurazione del gestore code per ciascun gestore code, contenuto nella root della struttura di directory occupata dal gestore code. Ad esempio, il percorso e il nome di un file di configurazione per un gestore code denominato QMNAME è:

  Su sistemi UNIX and Linux :

  ```
  /var/mqm/qmgrs/QMNAME/qm.ini
  ```

  Su sistemi Windows :

  ```
  C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\qm.ini
  ```

Prima di modificare un file di configurazione, eseguirne il backup in modo da disporre di una copia a cui è possibile tornare se necessario.

È possibile modificare i file di configurazione:

– Automaticamente, utilizzando i comandi che modificano la configurazione dei gestori code sul nodo

– Manualmente, utilizzando un editor di testo standard

Se si imposta un valore non corretto su un attributo del file di configurazione, il valore viene ignorato e viene emesso un messaggio operatore per indicare il problema. (L'effetto è lo stesso di perdere completamente l'attributo.)

Stanze da configurare

Le stanze che devono essere modificate sono le seguenti:

ApiExitCommon

Definito in mqs.ini e in IBM WebSphere MQ Explorer nella pagina delle proprietà di IBM WebSphere MQ , in Uscite.

Quando un gestore code viene avviato, gli attributi in questa stanza vengono letti e quindi sovrascritti dalle uscite API definite in qm.ini.

ApiExitTemplate

Definito in mqs.ini e in IBM WebSphere MQ Explorer nella pagina delle proprietà di IBM WebSphere MQ , in Uscite.

Quando viene creato un qualsiasi gestore code, gli attributi in questa stanza vengono copiati nel file qm.ini appena creato nella stanza ApiExitlocale.

ApiExitLocal

Definito in qm.ini e in IBM WebSphere MQ Explorer nella pagina Proprietà del gestore code, in Uscite.

Quando il gestore code viene avviato, le uscite API qui definite sovrascrivono le impostazioni predefinite definite in mqs.ini.

Attributi per le stanze

- Denominare l'uscita API utilizzando il seguente attributo:

  Nome=ApiExit_name
  Il nome descrittivo dell'uscita API inoltrato nel campo Nome ExitInfodella struttura MQAXP.

  Questo nome deve essere univoco, non deve superare i 48 caratteri e contenere solo caratteri validi per i nomi degli oggetti IBM WebSphere MQ (ad esempio, nomi coda).

- Identificare il modulo e il punto di ingresso del codice di uscita API da eseguire utilizzando i seguenti attributi:

  Funzione=nome_funzione
  Il nome del punto di ingresso della funzione nel modulo contenente il codice di uscita API. Questo punto di ingresso è la funzione MQ_INIT_EXIT.

  La lunghezza di questo campo è limitata a MQ_EXIT_NAME_LENGTH.

  Modulo=nome_modulo
  Il modulo contenente il codice di uscita API.

  Se questo campo contiene il percorso completo del modulo, questo verrà visualizzato così come è.

  Se questo campo contiene solo il nome del modulo, il modulo si trova utilizzando l'attributo `ExitsDefaultPath` in `ExitPath` in qm.ini.

  Su piattaforme che supportano librerie separate, è necessario fornire sia una versione non con thread che una versione con thread del modulo di uscita API. La versione con thread deve avere

un suffisso `_r` . La versione con thread dello stub dell'applicazione IBM WebSphere MQ accoda implicitamente `_r` al nome modulo fornito prima che venga caricato.

La lunghezza di questo campo è limitata alla lunghezza massima del percorso supportata dalla piattaforma.

- Facoltativamente, passare i dati con l'exit utilizzando il seguente attributo:

  Dati=nome_dati
  I dati da passare all'uscita API nel campo ExitData della struttura MQAXP.

  Se si include questo attributo, vengono rimossi gli spazi iniziali e finali, la stringa rimanente viene troncata a 32 caratteri e il risultato viene passato all'uscita. Se si omette questo attributo, il valore predefinito di 32 spazi viene passato all'uscita.

  La lunghezza massima di questo campo è 32 caratteri.

- Identificare la sequenza di questa uscita in relazione ad altre uscite utilizzando il seguente attributo:

  Sequenza=numero_sequenza
  La sequenza in cui questa uscita API viene richiamata rispetto ad altre uscite API. Un'uscita con un numero di sequenza basso viene richiamata prima di un'uscita con un numero di sequenza più alto. Non è necessario che la numerazione di sequenza delle uscite sia contigua. Una sequenza di 1, 2, 3 ha lo stesso risultato di una sequenza di 7, 42, 1096. Se due uscite hanno lo stesso numero di sequenza, il gestore code decide quale richiamare per primo. È possibile determinare quale elemento è stato richiamato dopo l'evento inserendo l'ora o un indicatore nell'area ExitChainindicata da ExitChainAreaPtr in MQAXP oppure scrivendo il proprio file di log.

  Questo attributo è un valore numerico senza segno.

Stanze di esempio

Il file mqs.ini di esempio contiene le seguenti stanze:

ApiExitTemplate
Questa stanza definisce un'uscita con il nome descrittivo `OurPayrollQueueAuditor`, il nome modulo `auditore` il numero di sequenza 2. Un valore dati di 123 viene passato all'uscita.

ApiExitCommon
Questa stanza definisce un'uscita con il nome descrittivo `MQPoliceman`, il nome modulo `tmqpe` il numero di sequenza 1. I dati passati sono un'istruzione (`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
```

Il seguente file qm.ini di esempio contiene una definizione locale ApiExitdi un'uscita con nome descrittivo `ClientApplicationAPIchecker`, nome modulo `ClientAppCheckere` numero di sequenza 3.

```
qm.ini

   ApiExitLocal:
      Name=ClientApplicationAPIchecker
      Sequence=3
      Function=EntryPoint
      Module=/usr/Dev/ClientAppChecker
      Data=9.20.176.20
```

Programmi di uscita canale per canali di messaggistica

Questa raccolta di argomenti contiene informazioni sui programmi channel - exit WebSphere MQ per canali di messaggistica.

Gli MCA (Message Channel Agent) possono anche richiamare le uscite di conversione dati. Per ulteriori informazioni sulla scrittura delle uscite di conversione dati, consultare “Scrittura delle uscite di conversione dati” a pagina 417.

Alcune di queste informazioni si applicano anche alle uscite sui canali MQI, che collegano i client WebSphere MQ MQI ai gestori code. Per ulteriori informazioni, consultare Channel - exit programs for MQI channels.

I programmi di uscita canale vengono richiamati in punti definiti nell'elaborazione eseguita dai programmi MCA.

Alcuni di questi programmi di uscita utente funzionano in coppie complementari. Ad esempio, se un programma di uscita utente viene richiamato dall'MCA mittente per codificare i messaggi per la trasmissione, il processo complementare deve funzionare all'estremità ricevente per invertire il processo.

La Tabella 55 a pagina 399 mostra i tipi di uscita canale disponibili per ogni tipo di canale.

Tabella 55. Uscite canale disponibili per ciascun tipo di canale

Tipo canale	Uscita messaggi	Uscita nuovo tentativo messaggio	Uscita ricezione	Uscita di sicurezza	Uscita invio	Uscita definizione automatica
Canale di trasmissione	Sì		Sì	Sì	Sì	
Canale server	Sì		Sì	Sì	Sì	
Canale mittente cluster	Sì		Sì	Sì	Sì	Sì
Canale di ricezione	Sì	Sì	Sì	Sì	Sì	Sì
Canale richiedente	Sì	Sì	Sì	Sì	Sì	
Canale ricevente cluster	Sì	Sì	Sì	Sì	Sì	Sì
Canale di connessione client			Sì	Sì	Sì	
Canale di connessione server			Sì	Sì	Sì	Sì

Se si stanno per eseguire uscite di canale su un client, non è possibile utilizzare la variabile di ambiente MQSERVER. Invece, creare e fare riferimento a una tabella di definizione del canale client (CCDT) come descritto in Tabella di definizione del canale client .

Panoramica sull'elaborazione

Una panoramica di come gli MCA utilizzano i programmi channel - exit.

All'avvio, gli MCA scambiano una finestra di dialogo di avvio per sincronizzare l'elaborazione. Quindi passano a uno scambio di dati che include le uscite di sicurezza. Queste uscite devono terminare correttamente per completare la fase di avvio e consentire il trasferimento dei messaggi.

La fase di verifica della protezione è un loop, come mostrato in Figura 77 a pagina 400.

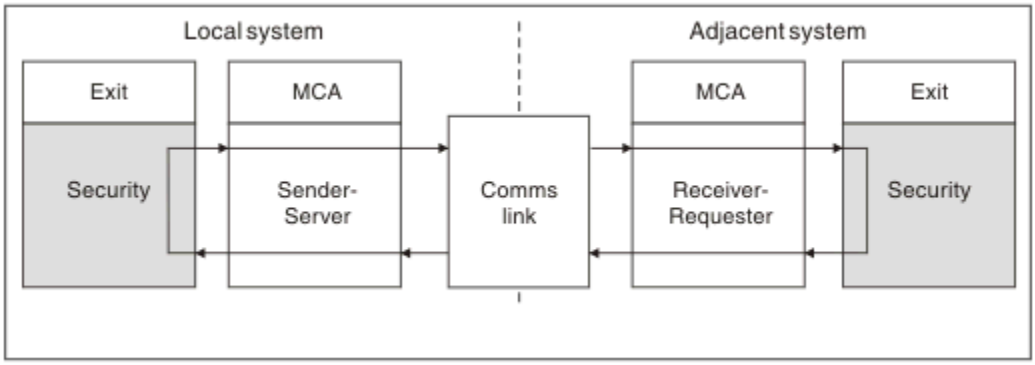


Figura 77. Loop uscita di sicurezza

Durante la fase di trasferimento del messaggio, l'MCA mittente riceve i messaggi da una coda di trasmissione, richiama l'uscita del messaggio, richiama l'uscita di invio e quindi invia il messaggio all'MCA ricevente, come mostrato in Figura 78 a pagina 400.

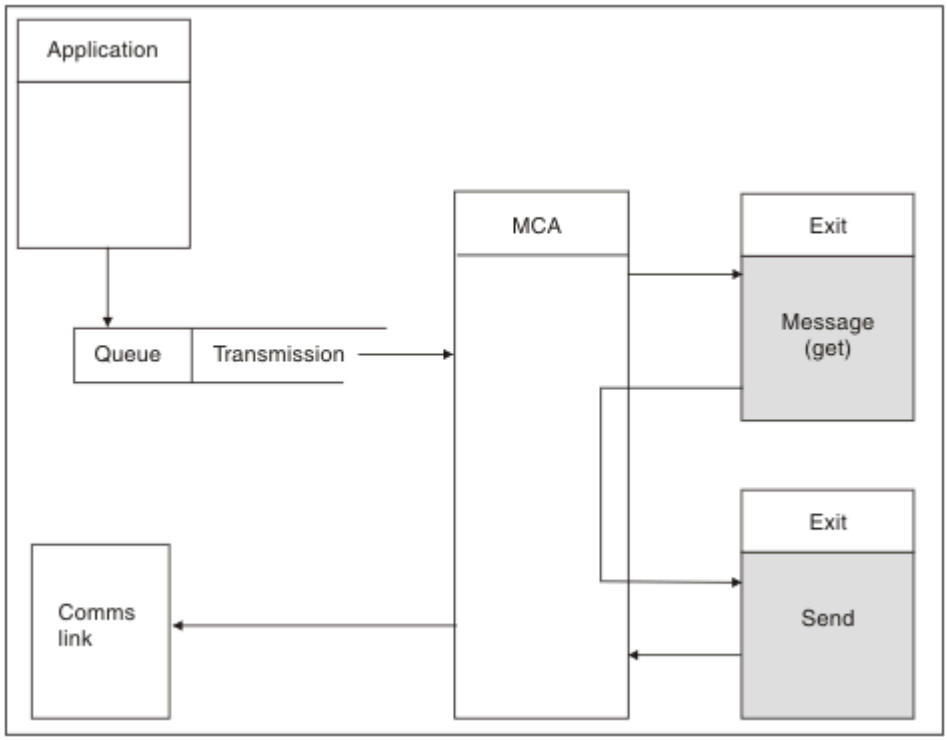


Figura 78. Esempio di uscita di invio all'estremità mittente del canale di messaggi

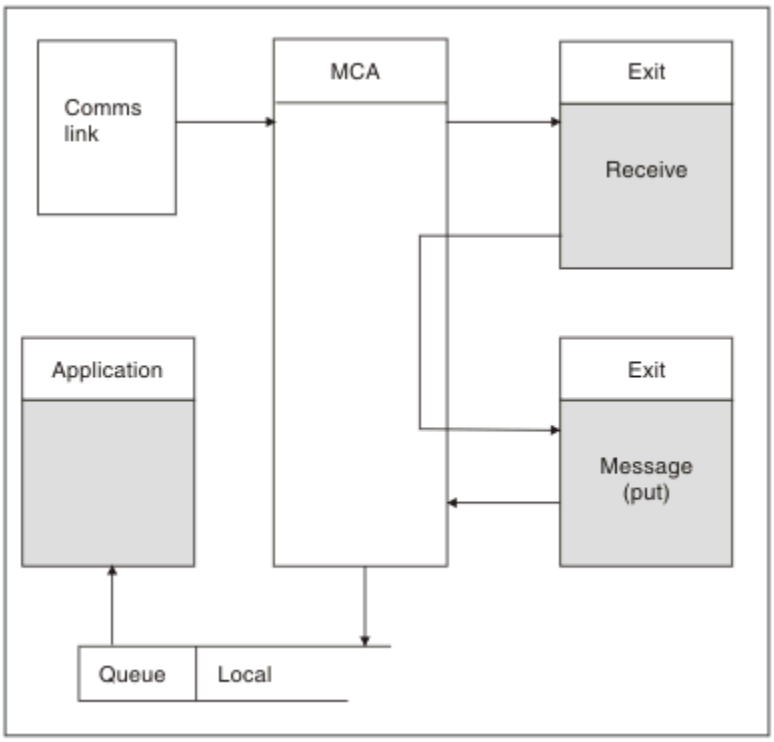


Figura 79. Esempio di un'uscita di ricezione all'estremità del destinatario del canale di messaggi

L'MCA ricevente riceve un messaggio dal collegamento di comunicazione, richiama l'uscita di ricezione, richiama l'uscita di messaggio e inserisce il messaggio sulla coda locale, come mostrato in Figura 79 a pagina 401. (L'uscita di ricezione può essere richiamata più di una volta prima che venga richiamata l'uscita del messaggio.)

Scrittura di programmi di uscita canale

È possibile utilizzare le seguenti informazioni per scrivere programmi di uscita canale.

Le uscite utente e i programmi di uscita canale possono utilizzare tutte le chiamate MQI, tranne come indicato nelle sezioni che seguono. Per MQ V7 e versioni successive, la struttura MQCXP versione 7 e successive contiene l'handle di connessione hConn, che può essere utilizzato invece di emettere MQCONN. Per le versioni precedenti, per ottenere l'handle di connessione, è necessario emettere un MQCONN, anche se viene restituita un'avvertenza MQRC_ALREADY_CONNECTED perché il canale stesso è connesso al gestore code.

Notare che l'uscita del canale deve essere thread - safe.

Per le uscite sui canali di connessione client, il gestore code a cui l'exit tenta di connettersi dipende dal modo in cui l'exit è stata collegata. Se l'uscita è stata collegata a MQM.LIB e non si specifica un nome del gestore code nella chiamata MQCONN, l'uscita tenta di connettersi al gestore code predefinito sul sistema. Se l'uscita è stata collegata a MQM.LIB e si specifica il nome del gestore code che è stato passato all'uscita tramite il campo QMgrName di MQCD, l'uscita tenta di connettersi a tale gestore code. Se l'uscita è stata collegata a MQIC.LIB o qualsiasi altra libreria, la chiamata MQCONN ha esito negativo se si specifica o meno un nome gestore code.

Si consiglia di evitare di modificare lo stato della transazione associata al hConn inoltrato in un'uscita del canale; non è necessario utilizzare i verbi MQCMIT, MQBACK o MQDISC con il canale hConne non è possibile utilizzare il verbo MQBEGIN specificando il canale hConn.

Se si utilizza MQCONNX specificando MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK per creare una nuova connessione IBM WebSphere MQ , è responsabilità dell'utente assicurarsi che la connessione sia gestita correttamente e che si disconnetta correttamente dal gestore code. Ad esempio, un'uscita del canale che crea una nuova connessione al

gestore code ad ogni chiamata senza disconnettersi, determina la creazione di handle di connessione e un aumento del numero di thread dell'agent.

Un'uscita viene eseguita nello stesso thread dell'MCA stesso e utilizza lo stesso handle di connessione. Quindi, viene eseguito all'interno della stessa UOW dell'MCA e tutte le chiamate effettuate nel punto di sincronizzazione vengono sottoposte a commit o a backout dal canale alla fine del batch.

Pertanto, un'uscita messaggio del canale potrebbe inviare messaggi di notifica di cui è stato eseguito il commit solo su quella coda quando è stato eseguito il commit del batch contenente il messaggio originale. Quindi, è possibile emettere le chiamate MQI del punto di sincronizzazione da un'uscita del messaggio del canale.

Un'uscita canale può cambiare i campi in MQCD. Tuttavia, tali modifiche non vengono applicate, ad eccezione delle circostanze elencate. Se un programma di uscita del canale modifica un campo nella struttura dati MQCD, il nuovo valore viene ignorato dal processo del canale IBM WebSphere MQ . Tuttavia, il nuovo valore rimane nel MQCD e viene passato a tutte le uscite rimanenti in una catena di uscita e a tutte le conversazioni che condividono l'istanza del canale. Per ulteriori informazioni, consultare Modifica dei campi MQCD in un'uscita canale

Inoltre, per i programmi scritti in C, la funzione libreria C non rientrante non deve essere utilizzata in un programma di uscita canale.

Se si utilizzano più librerie di uscita del canale contemporaneamente, possono verificarsi problemi su alcune piattaforme UNIX and Linux se il codice per due diverse uscite contiene funzioni con lo stesso nome. Quando viene caricata un'uscita canale, il programma di caricamento dinamico risolve i nomi delle funzioni nella libreria di uscita negli indirizzi in cui viene caricata la libreria. Se due librerie di uscita definiscono funzioni separate che hanno nomi identici, questo processo di risoluzione potrebbe risolvere in modo non corretto i nomi funzione di una libreria per utilizzare le funzioni di un'altra. Se si verifica questo problema, specificare al linker che deve solo esportare le funzioni di uscita e MQStart richieste, poiché queste funzioni non sono interessate. Altre funzioni devono avere una visibilità locale in modo che non vengano utilizzate da funzioni esterne alla propria libreria di uscita. Consultare la documentazione per il linker per ulteriori informazioni.

Tutte le uscite sono richiamate con una struttura di parametri di uscita del canale (MQCXP), una struttura di definizione del canale (MQCD), un buffer di dati preparato, un parametro di lunghezza dati e un parametro di lunghezza del buffer. La lunghezza del buffer non deve essere superata:

- Per le uscite dei messaggi, è necessario consentire l'invio del messaggio più grande richiesto attraverso il canale, più la lunghezza della struttura MQXQH.
- Per le uscite di invio e ricezione, il buffer più grande che è necessario consentire è il seguente:

  LU 6.2
  32 KB

  TCP:
  32 KB

  Nota: La lunghezza massima utilizzabile potrebbe essere di 2 byte inferiore a questa lunghezza. Verificare il valore restituito in MaxSegmentLength per i dettagli. Per ulteriori informazioni sulla lunghezza MaxSegment, consultare MaxSegmentLength.

  NetBIOS:
  64 KB

  SPX:
  64 KB

  Nota: Le uscite di ricezione sui canali mittenti e le uscite mittente sui canali riceventi utilizzano buffer di 2 KB per TCP.
- Per le uscite di sicurezza, la funzione di accodamento distribuito assegna un buffer di 4000 byte.

È consentito che l'uscita restituisca un buffer alternativo, insieme ai parametri pertinenti. Consultare “Programmi di uscita canale per canali di messaggistica” a pagina 399 per i dettagli della chiamata.

Scrittura di programmi di uscita del canale su sistemi Windows, UNIX and Linux
È possibile utilizzare le seguenti informazioni per scrivere programmi di uscita del canale per sistemi Windows, UNIX and Linux .

Seguire le istruzioni descritte in "Scrittura e compilazione di uscite e servizi installabili" a pagina 376. Utilizzare le seguenti informazioni specifiche sull'uscita del canale, dove appropriato:

L'uscita deve essere scritta in C ed è una DLL in Windows.

Definire una routine MQStart () fittizia nell'uscita e specificare `MQStart` come punto di ingresso nella libreria. Figura 80 a pagina 403 mostra come impostare una voce per il programma:

```
#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
}
```

Figura 80. Codice sorgente di esempio per un'uscita canale

Quando si scrivono uscite di canale per Windows utilizzando Visual C + +, è necessario scrivere il file DEF . Un esempio di come viene mostrato in Figura 81 a pagina 403. Per ulteriori informazioni sulla scrittura dei programmi di uscita del canale, consultare "Scrittura di programmi di uscita canale" a pagina 401.

```
EXPORTS
ChannelExit
```

Figura 81. File DEF di esempio per Windows

Programmi di uscita di sicurezza del canale
È possibile utilizzare i programmi di uscita di sicurezza per verificare che il partner all'altra estremità di un canale sia autentico. Questa operazione è nota come autenticazione. Per specificare che un canale deve utilizzare un'uscita sicurezza, specificare il nome dell'uscita nel campo SCYEXIT della definizione del canale.

Nota: L'autenticazione può essere ottenuta anche con i record di autenticazione di canale. I record di autenticazione di canale forniscono una grande flessibilità nella prevenzione dell'accesso ai gestori code da parte di determinati utenti e canali e nell'associazione di utenti remoti agli identificatori utente IBM WebSphere MQ . Il supporto SSL e TLS viene fornito anche da IBM WebSphere MQ per autenticare gli utenti e per fornire la crittografia e i controlli di integrità dei dati per i tuoi dati. Per ulteriori informazioni su SSL e TLS, consultare WebSphere MQ support for SSL and TLS. Tuttavia, se si richiedono ancora forme più sofisticate (o diverse) di elaborazione della sicurezza e altri tipi di verifiche e di creazione di un contesto di sicurezza, considerare la scrittura di uscite di sicurezza.

Per le uscite di sicurezza scritte prima di IBM WebSphere MQ Version 7.1 , vale la pena notare che le versioni precedenti di IBM WebSphere MQ hanno eseguito una query del provider di socket sicuri sottostante (ad esempio, GSKit) per determinare il certificato del partner remoto SSLPEER (Subject Distinguished Name) e SSLCERTI (Issuer Distinguished Name). In IBM WebSphere MQ Version 7.1 è stato aggiunto il supporto per una serie di nuovi attributi di sicurezza. Per accedere a questi attributi, IBM WebSphere MQ Version 7.1 ottiene la codifica DER del certificato e la utilizza per determinare il DN soggetto e emittente. Gli attributi Oggetto e DN emittente vengono visualizzati nei seguenti attributi di stato del canale:

- SSLPEER (selettore PCF MQCACH_SSL_SHORT_PEER_NAME)
- SSLCERTI (selettore PCF MQCACH_SSL_CERT_ISSUER_NAME)

Questi valori vengono restituiti dai comandi di stato del canale e dai dati passati alle uscite di sicurezza del canale elencati, come mostrato:

- Ptr MQCD SSLPeerName
- Ptr MQCXP SSLRemCertIssName

In IBM WebSphere MQ Version 7.1, un attributo SERIALNUMBER è incluso anche nel DN soggetto e contiene il numero di serie per il certificato partner remoto. Inoltre, alcuni attributi DN vengono restituiti in una sequenza diversa rispetto ai rilasci precedenti. Di conseguenza, la composizione dei campi SSLPEER e SSLCERTI viene alterata in Version 7.1 rispetto alle release precedenti e si consiglia pertanto di esaminare e aggiornare tutte le uscite di sicurezza o le applicazioni dipendenti da tali campi.

I filtri del nome peer WebSphere MQ esistenti specificati tramite il campo SSLPEER di una definizione di canale non sono interessati e continueranno a funzionare nello stesso modo delle release precedenti. Questo perché l'algoritmo di corrispondenza del nome peer WebSphere MQ è stato aggiornato per elaborare i filtri SSLPEER esistenti senza alcuna necessità di modificare le definizioni di canale. È più probabile che questa modifica influisca sulle uscite di sicurezza e sulle applicazioni che dipendono dai valori DN soggetto e DN emittente restituiti dall'interfaccia di programmazione PCF.

Un'uscita di sicurezza può essere scritta in C o Java.

I programmi di uscita di sicurezza del canale vengono richiamati nelle seguenti posizioni nel ciclo di elaborazione di un MCA:

- All'inizio e alla fine di MCA.
- Immediatamente dopo che la negoziazione dei dati iniziale è terminata all'avvio del canale. Il destinatario o l'estremità del server del canale può avviare uno scambio di messaggi di sicurezza con l'estremità remota fornendo un messaggio da consegnare all'uscita di sicurezza all'estremità remota. Potrebbe anche rifiutarsi di farlo. Il programma di uscita viene riavviato per elaborare qualsiasi messaggio di sicurezza ricevuto dall'estremità remota.
- Immediatamente dopo che la negoziazione dei dati iniziale è terminata all'avvio del canale. L'estremità mittente o richiedente del canale elabora un messaggio di sicurezza ricevuto dall'estremità remota oppure avvia uno scambio di sicurezza quando l'estremità remota non può. Il programma di uscita viene avviato di nuovo per elaborare tutti i messaggi di sicurezza successivi che potrebbero essere ricevuti.

Un canale richiedente non viene mai richiamato con MQXR_INIT_SEC. Il canale notifica al server di avere un programma di uscita di sicurezza e il server ha quindi la possibilità di avviare un'uscita di sicurezza. Se non ne ha uno, informa il richiedente e viene restituito un flusso di lunghezza zero al programma di uscita.

Nota: Evitare di inviare messaggi di sicurezza a lunghezza zero.

Esempi di dati scambiati dai programmi di uscita di sicurezza sono illustrati nelle figure Figura 82 a pagina 405 tramite Figura 85 a pagina 407. Questi esempi mostrano la sequenza di eventi che si verificano che coinvolgono l'uscita di sicurezza del ricevente e l'uscita di sicurezza del mittente. Le righe successive nelle figure rappresentano il passare del tempo. In alcuni casi, gli eventi al destinatario e al mittente non sono correlati e quindi possono verificarsi contemporaneamente o in momenti diversi. In altri casi, un evento in un programma di uscita risulta in un evento complementare che si verifica successivamente nell'altro programma di uscita. Ad esempio, in Figura 82 a pagina 405:

1. Il destinatario e il mittente vengono richiamati ciascuno con MQXR_INIT, ma questi richiami non sono correlati e possono quindi verificarsi contemporaneamente o in momenti diversi.
2. Il destinatario viene successivamente richiamato con MQXR_INIT_SEC, ma restituisce MQXCC_OK che non richiede alcun evento complementare all'uscita del mittente.
3. Il mittente viene quindi richiamato con MQXR_INIT_SEC. Ciò non è correlato al richiamo del destinatario con MQXR_INIT_SEC. Il mittente restituisce MQXCC_SEND_SEC_MSG, che causa un evento complementare all'uscita del destinatario.
4. Il destinatario viene quindi richiamato con MQXR_SEC_MSG e restituisce MQXCC_SEND_SEC_MSG, che causa un evento complementare all'uscita del mittente.
5. Il mittente viene quindi richiamato con MQXR_SEC_MSG e restituisce MQXCC_OK che non richiede alcun evento complementare all'uscita del destinatario.

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	

Figura 82. Scambio avviato dal mittente con accordo

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

Figura 83. Scambio avviato dal mittente senza accordo

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

Figura 84. Scambio avviato dal destinatario con accordo

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	

Figura 85. Scambio avviato dal destinatario senza accordo

Al programma di uscita di sicurezza del canale viene passato un buffer dell'agent contenente i dati di sicurezza, escluse le intestazioni di trasmissione, generati dall'uscita di sicurezza. Questi dati possono essere dati adatti in modo che entrambe le estremità del canale siano in grado di eseguire la convalida di sicurezza.

Il programma di uscita di sicurezza, sia all'estremità di invio che a quella di ricezione del canale messaggi, può restituire uno dei due codici di risposta a qualsiasi chiamata:

- Scambio di sicurezza terminato senza errori
- Sopprimere il canale e chiudere

Nota:

1. Le uscite di sicurezza del canale generalmente funzionano a coppie. Quando si definiscono i canali appropriati, verificare che i programmi di uscita compatibili siano denominati per entrambe le estremità del canale.
2. In IBM i, i programmi di uscita di sicurezza che sono stati compilati con "Utilizza autorizzazione adottata" (USEADPAUT = *YES) possono adottare l'autorizzazione QMQM o QMQMADM. Fare attenzione che l'uscita non utilizzi questa funzione per rappresentare un rischio per la sicurezza del sistema.
3. Su un canale SSL su cui l'altra estremità del canale fornisce un certificato, l'uscita di sicurezza riceve il DN (Distinguished Name) dell'oggetto di questo certificato nel campo MQCD a cui accede SSLPeerNamePtr e il DN (Distinguished Name) dell'emittente nel campo MQCXP a cui accede SSLRemCertIssNamePtr. Gli utilizzi in cui è possibile inserire questo nome sono:
   - Per limitare l'accesso sul canale SSL.
   - Per modificare MQCD.MCAUserIdentifier basato sul nome.

Concetti correlati

Record di autenticazione di canale

Concetti SSL (Secure Sockets Layer) e TLS (Transport Layer Security)

Scrittura di un'uscita di sicurezza

È possibile scrivere un'uscita di sicurezza utilizzando il codice di base dell'uscita di sicurezza.

Figura 86 a pagina 408 illustra come scrivere un'uscita di sicurezza.

```
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 */

}
```

Figura 86. Codice di base uscita di sicurezza

Lo standard WebSphere MQ Entry Point MQStart deve esistere, ma non è richiesto per eseguire alcuna funzione. Il nome della funzione (EntryPoint in questo esempio) può essere modificato, ma la funzione deve essere esportata quando la libreria viene compilata e collegata. Come nell'esempio precedente, i puntatori pChannelExitParms devono essere passati a PMQCXP e la definizione pChannela PMQCD. Per informazioni generali sulla chiamata delle uscite del canale e sull'utilizzo dei parametri, vedi MQ_CHANNEL_EXIT. Questi parametri vengono utilizzati in un'uscita di sicurezza come segue:

PMQVOID pChannelExitParms
Input/output

Puntatore alla struttura MQCXP - cast a PMQCXP per accedere ai campi. Questa struttura viene utilizzata per comunicare tra Exit e MCA. I seguenti campi in MQCXP sono particolarmente interessanti per le uscite di sicurezza:

ExitReason
Indica all'uscita di sicurezza lo stato corrente nello scambio di sicurezza e viene utilizzato quando si decide quale azione intraprendere.

ExitResponse
La risposta all'MCA che determina la fase successiva nello scambio di sicurezza.

ExitResponse2
Indicatori di controllo supplementari per gestire il modo in cui l'MCA interpreta la risposta dell'uscita di sicurezza.

Area ExitUser
16 byte (massimo) di memoria che possono essere utilizzati dall'uscita di sicurezza per mantenere lo stato tra le chiamate.

ExitData
Contiene i dati specificati nel campo SCYDATA della definizione del canale (32 byte riempiti a destra con spazi).

Definizione pChannelPMQVOID
Input/output

Puntatore alla struttura MQCD - cast a PMQCD per accedere ai campi. Questo parametro contiene la definizione del canale. I seguenti campi in MQCD sono di particolare interesse per le uscite di sicurezza:

ChannelName
Il nome del canale (20 byte riempiti a destra con spazi).

ChannelType
Un codice che definisce il tipo di canale.

Identificativo utente MCA
Questo gruppo di tre campi viene inizializzato sul valore del campo MCAUSER specificato nella definizione del canale. Qualsiasi identificativo utente specificato dall'uscita di sicurezza in questi campi viene utilizzato per il controllo accessi (non applicabile ai canali SDR, SVR, CLNTCONN o CLUSSDR).

MCAUserIdentifier
I primi 12 byte dell'identificativo sono stati riempiti a destra con spazi.

LongMCAUserIdPtr
Il puntatore a un buffer che contiene l'identificativo di lunghezza completa (con terminazione null non garantita) ha la priorità su MCAUserIdentifier.

LongMCAUserIdLength
Lunghezza della stringa indicata da LongMCAUserIdPtr - deve essere impostata se è impostato LongMCAUserIdPtr .

Identificativo utente remoto
Si applica solo alle coppie di canali CLNTCONN/SVRCONN. Se non è definita alcuna uscita di sicurezza CLNTCONN, questi tre campi vengono inizializzati dall'MCA client, quindi potrebbero contenere un identificativo utente dall'ambiente del client che può essere utilizzato da un'uscita di sicurezza SVRCONN per l'autenticazione e quando si specifica l'identificativo utente MCA. Se è definita un'uscita di sicurezza CLNTCONN, questi campi non vengono inizializzati e possono essere impostati dall'uscita di protezione CLNTCONN oppure è possibile utilizzare i messaggi di sicurezza per passare un identificativo utente dal client al server.

Identificativo RemoteUser
I primi 12 byte dell'identificativo sono stati riempiti a destra con spazi.

LongRemoteUserIdPtr
Il puntatore a un buffer contenente l'identificativo di lunghezza completa (con terminazione null non garantita) ha la priorità sull'identificativo RemoteUser.

LongRemoteUserIdLunghezza
Lunghezza della stringa indicata da LongRemoteUserIdPtr - deve essere impostata se è impostato LongRemoteUserIdPtr.

Lunghezza pDataPMQLONG
Input/output

Puntatore a MQLONG. Contiene la lunghezza di qualsiasi uscita di sicurezza contenuta in AgentBuffer al richiamo dell'uscita di sicurezza. Deve essere impostato da un'uscita di sicurezza sulla lunghezza di qualsiasi messaggio inviato in AgentBuffer o ExitBuffer.

PMQLONG pAgentBufferLength
input

Puntatore a MQLONG. La lunghezza dei dati contenuti nel AgentBuffer al richiamo dell'uscita di sicurezza.

Buffer pAgentPMQVOID
Input/output

Al richiamo dell'uscita di sicurezza, questo fa riferimento a qualsiasi messaggio inviato dall'uscita partner. Se ExitResponse2 nella struttura di MQCXP ha l'indicatore MQXR2_USE_AGENT_BUFFER impostato (predefinito), un'uscita di sicurezza deve impostare questo parametro per puntare a tutti i dati del messaggio inviati.

PMQLONG pExitBufferLength
Input/output

Puntatore a MQLONG. Questo parametro viene inizializzato a 0 al primo richiamo di un'uscita di sicurezza e il valore restituito viene mantenuto tra le chiamate all'uscita di sicurezza durante uno scambio di sicurezza.

PMQPTR pExitBufferAddr
Input/output

Questo parametro viene inizializzato su un puntatore null al primo richiamo di un'uscita di sicurezza e il valore restituito viene mantenuto tra le chiamate all'uscita di sicurezza durante uno scambio di sicurezza. Se l'indicatore MQXR2_USE_EXIT_BUFFER è impostato in ExitResponse2 nella struttura MQCXP, un'uscita di sicurezza deve impostare questo parametro per puntare a qualsiasi dato del messaggio inviato.

Differenze nel comportamento tra le uscite di sicurezza definite sulle coppie di canali CLNTCONN/SVRCONN e altre coppie di canali
Le uscite di sicurezza possono essere definite su tutti i tipi di canale. Tuttavia, il comportamento delle uscite di sicurezza definite sulle coppie di canali CLNTCONN/SVRCONN è leggermente differente dalle uscite di sicurezza definite su altre coppie di canali.

Un'uscita di sicurezza su un canale CLNTCONN può impostare l'identificativo utente remoto nella definizione del canale per l'elaborazione da parte di un'uscita SVRCONN partner o per l'autorizzazione OAM se non è definita alcuna uscita di sicurezza SVRCONN e il campo MCAUSER di SVRCONN non è impostato.

Se non viene definita alcuna uscita di sicurezza CLNTCONN, l'identificativo utente remoto nella definizione del canale viene impostato su un identificativo utente dall'ambiente client (che può essere vuoto) dall'MCA del client.

Uno scambio di sicurezza tra le uscite di sicurezza definite su una coppia di canali CLNTCONN e SVRCONN viene completato correttamente quando l'uscita di sicurezza SVRCONN restituisce una ExitResponse di MQXCC_OK. Uno scambio di sicurezza tra altre coppie di canali viene completato correttamente quando l'uscita di protezione che ha avviato lo scambio restituisce una ExitResponse di MQXCC_OK.

Tuttavia, il codice MQXCC_SEND_AND_REQUEST_SEC_MSG ExitResponse può essere utilizzato per forzare il proseguimento dello scambio di sicurezza: se una ExitResponse di MQXCC_SEND_AND_REQUEST_SEC_MSG viene restituita da un'uscita di sicurezza CLNTCONN o SVRCONN, l'uscita partner deve rispondere inviando un messaggio di sicurezza (non MQXCC_OK o una risposta null) o il canale viene terminato. Per le uscite di sicurezza definite su altri tipi di canale, una ExitResponse di MQXCC_OK restituita in risposta a MQXCC_SEND_AND_REQUEST_SEC_MSG dall'uscita di sicurezza del partner risulta nella continuazione dello scambio di sicurezza come se fosse stata restituita una risposta null e non in chiusura del canale.

Uscita di sicurezza SSPI
WebSphere MQ per Windows fornisce un'uscita di sicurezza che fornisce autenticazione per canali WebSphere MQ utilizzando SSPI (Security Services Programming Interface). SSPi fornisce le funzioni di sicurezza integrate di Windows.

Questa uscita di sicurezza è sia per il client WebSphere MQ che per il server WebSphere MQ .

I pacchetti di sicurezza vengono caricati da security.dll o secur32.dll. Queste DLL vengono fornite con il sistema operativo.

L'autenticazione unidirezionale viene fornita su Windows, utilizzando i servizi di autenticazione NTLM. L'autenticazione bidirezionale viene fornita su Windows 2000, utilizzando servizi di autenticazione Kerberos .

Il programma di uscita di sicurezza viene fornito in formato origine e oggetto. È possibile utilizzare il codice oggetto così com' è oppure è possibile utilizzare il codice origine come punto di partenza per creare i propri programmi di uscita utente. Per ulteriori informazioni sull'utilizzo dell'oggetto o del codice sorgente dell'uscita di sicurezza SSPI, consultare “Utilizzo dell'uscita di sicurezza SSPI su sistemi Windows” a pagina 170

Programmi di uscita di invio e ricezione del canale
È possibile utilizzare le uscite di invio e ricezione per eseguire attività quali la compressione e la decompressione dei dati. È possibile specificare un elenco di programmi di uscita di invio e ricezione da eseguire in successione.

I programmi di uscita di invio e ricezione del canale vengono richiamate nelle seguenti posizioni nel ciclo di elaborazione di un MCA:

- I programmi di uscita di invio e ricezione vengono richiamati per l'inizializzazione all'avvio MCA e per la terminazione alla terminazione MCA.
- Il programma di uscita di invio viene richiamato all'una o all'altra estremità del canale, a seconda della fine in cui viene inviata una trasmissione per un trasferimento di messaggi, immediatamente prima che venga inviata una trasmissione sul collegamento. La nota 4 spiega perché le uscite sono disponibili in entrambe le direzioni anche se i canali di messaggi inviano messaggi in una sola direzione.
- Il programma di uscita di ricezione viene richiamato all'una o all'altra estremità del canale, a seconda della fine in cui viene ricevuta una trasmissione per un trasferimento di messaggi, immediatamente dopo che una trasmissione è stata presa dal collegamento. La nota 4 spiega perché le uscite sono disponibili in entrambe le direzioni anche se i canali di messaggi inviano messaggi in una sola direzione.

È possibile che vi siano molte trasmissioni per un trasferimento di messaggi e che vi siano molte iterazioni dei programmi di uscita di invio e ricezione prima che un messaggio raggiunga l'uscita di messaggio all'estremità di ricezione.

Ai programmi di uscita di invio e ricezione del canale viene passato un buffer dell'agent che contiene i dati di trasmissione come inviati o ricevuti dal collegamento delle comunicazioni. Per i programmi di uscita di invio, i primi 8 byte del buffer sono riservati all'utilizzo da parte di MCA e non devono essere modificati. Se il programma restituisce un buffer differente, questi primi 8 byte devono esistere nel nuovo buffer. Il formato dei dati presentati ai programmi di uscita non è definito.

Un buon codice di risposta deve essere restituito dai programmi di uscita di invio e ricezione. Qualsiasi altra risposta causa una fine anomala MCA (fine anomala).

Nota: Non emettere una chiamata MQGET, MQPUT o MQPUT1 all'interno di un punto di sincronizzazione da un'uscita di invio o ricezione.

Nota:

1. Le uscite di invio e ricezione generalmente funzionano in coppie. Ad esempio, un'uscita di invio potrebbe comprimere i dati e un'uscita di ricezione potrebbe decomprimerli oppure un'uscita di invio potrebbe codificare i dati e un'uscita di ricezione potrebbe decodificarli. Quando si definiscono i canali appropriati, verificare che i programmi di uscita compatibili siano denominati per entrambe le estremità del canale.
2. Se la compressione è attivata per il canale, alle uscite vengono trasmessi dati compressi.
3. Le uscite di invio e ricezione del canale potrebbero essere richiamate per i segmenti di messaggio diversi dai dati dell'applicazione, ad esempio i messaggi di stato. Non vengono richiamati durante la finestra di avvio, né durante la fase di verifica della sicurezza.
4. Anche se i canali dei messaggi inviano i messaggi solo in una direzione, i dati di controllo del canale, come i battiti cardiaci e la fine dell'elaborazione batch, fluiscono in entrambe le direzioni e queste uscite sono disponibili anche in entrambe le direzioni. Tuttavia, alcuni dei flussi di dati di avvio del canale iniziale sono esenti dall'elaborazione da parte di una qualsiasi delle uscite.
5. Ci sono circostanze in cui le uscite di invio e ricezione possono essere richiamate fuori sequenza; ad esempio, se si sta eseguendo una serie di programmi di uscita o se si stanno eseguendo anche uscite di sicurezza. Quindi, quando l'uscita di ricezione viene richiamata per la prima volta per elaborare i dati, potrebbe ricevere i dati che non sono passati attraverso l'uscita di invio corrispondente. Se l'uscita di ricezione ha appena eseguito l'operazione, ad esempio la decompressione, senza prima verificare che sia stata richiesta, i risultati sarebbero imprevisti.

   È necessario codificare le uscite di invio e ricezione in modo che l'uscita di ricezione possa controllare che i dati ricevuti siano stati elaborati dalla corrispondente uscita di invio. Il modo consigliato per farlo è quello di codificare i programmi di uscita in modo che:

   - L'uscita di invio imposta il valore del nono byte di dati su 0 e sposta tutti i dati su 1 byte, prima di eseguire l'operazione. (I primi 8 byte sono riservati per l'utilizzo da parte di MCA.)
   - Se l'uscita di ricezione riceve dati che hanno uno 0 in byte 9, sa che i dati provengono dall'uscita di invio. Rimuove lo 0, esegue l'operazione complementare e sposta i dati risultanti indietro di 1 byte.
   - Se l'uscita di ricezione riceve dati che hanno un valore diverso da 0 nel byte 9, assume che l'uscita di invio non sia stata eseguita e restituisce i dati al chiamante senza modificarli.

   Quando si utilizzano le uscite di sicurezza, se il canale viene terminato dall'uscita di sicurezza, è possibile che un'uscita di invio venga richiamata senza la corrispondente uscita di ricezione. Un modo per evitare questo problema consiste nel codificare l'uscita di sicurezza per impostare un indicatore, in MQCD.SecurityUserData o MQCD.SendUserData, ad esempio, quando l'uscita decide di terminare il canale. Quindi, l'uscita di invio deve controllare questo campo ed elaborare i dati solo se l'indicatore non è impostato. Questo controllo impedisce all'uscita di invio di modificare inutilmente i dati e quindi evita eventuali errori di conversione che potrebbero verificarsi se l'uscita di sicurezza ha ricevuto dati modificati.

Programmi di uscita di invio del canale - prenotazione di spazio
È possibile utilizzare uscite di invio e ricezione per trasformare i dati prima della trasmissione. I programmi di uscita invio canale possono aggiungere i propri dati sulla trasformazione riservando spazio nel buffer di trasmissione.

Questi dati vengono elaborati dal programma di uscita di ricezione e quindi rimossi dal buffer. Ad esempio, potresti voler crittografare i dati e aggiungere una chiave di sicurezza per la decrittografia.

Come riservare spazio e utilizzarlo

Quando il programma di uscita di invio viene richiamato per l'inizializzazione, impostare il campo *ExitSpace* di MQXCP sul numero di byte da riservare. Consultare MQCXP per i dettagli. *ExitSpace* può essere impostato solo durante l'inizializzazione, ovvero quando *ExitReason* ha valore MQXR_INIT.

Quando l'uscita di invio viene richiamata immediatamente prima della trasmissione, con *ExitReason* impostato su MQXR_XMIT, *ExitSpace* byte sono riservati nel buffer di trasmissione. *ExitSpace* non è supportato su z/OS.

L'uscita di invio non deve necessariamente utilizzare tutto lo spazio riservato. Può utilizzare meno di *ExitSpace* byte o, se il buffer di trasmissione non è pieno, l'uscita può utilizzare più della quantità riservata. Quando si imposta il valore *ExitSpace*, è necessario lasciare almeno 1 KB per i dati del messaggio nel buffer di trasmissione. Le prestazioni del canale possono essere influenzate se viene utilizzato spazio riservato per grandi quantità di dati.

Cosa succede all'estremità ricevente del canale

I programmi di uscita di ricezione del canale devono essere impostati per essere compatibili con le corrispondenti uscite di invio. Le uscite di ricezione devono conoscere il numero di byte nello spazio riservato e devono rimuovere i dati in tale spazio.

Uscite di invio multiple

È possibile specificare un elenco di programmi di uscita di invio e ricezione da eseguire in successione. WebSphere MQ conserva un totale per lo spazio riservato da tutte le uscite di invio. Questo spazio totale deve lasciare almeno 1 KB per i dati del messaggio nel buffer di trasmissione.

Il seguente esempio mostra come viene assegnato lo spazio per tre uscite di invio, chiamate in successione:

1. Quando viene richiamata l'inizializzazione:
   - L'uscita di invio A riserva 1 KB.
   - L'uscita di invio B riserva 2 KB.
   - Uscita di invio C riserva 3 KB.
2. La dimensione massima di trasmissione è 32 KB e i dati utente sono lunghi 5 KB.
3. L'uscita A viene richiamata con 5 KB di dati; sono disponibili fino a 27 KB, poiché 5 KB sono riservati alle uscite B e C. L'uscita A aggiunge 1 KB, la quantità riservata.
4. L'uscita B viene richiamata con 6 KB di dati; sono disponibili fino a 29 KB, poiché 3 KB sono riservati all'uscita C. L'uscita B aggiunge 1 KB, meno dei 2 KB riservati.
5. L'uscita C viene richiamata con 7 KB di dati; sono disponibili fino a 32 KB. L'uscita C aggiunge 10K, più dei 3 KB che ha riservato. Questa quantità è valida, poiché la quantità totale di dati, 17 KB, è inferiore al massimo di 32 KB.

Programmi di uscita messaggi canale
È possibile utilizzare l'uscita del messaggio del canale per eseguire attività quali la crittografia sul collegamento, la convalida o la sostituzione di ID utente in entrata, la conversione dei dati del messaggio, la registrazione su giornale e la gestione dei messaggi di riferimento. È possibile specificare un elenco di programmi di uscita messaggi da eseguire in successione.

I programmi di uscita dei messaggi del canale vengono richiamati nelle seguenti posizioni nel ciclo di elaborazione dell'MCA:

- All'inizio e alla fine di MCA
- Immediatamente dopo che un MCA di invio ha emesso una chiamata MQGET
- Prima che l'MCA ricevente emani una chiamata MQPUT

All'uscita del messaggio viene passato un buffer dell'agent contenente l'intestazione della coda di trasmissione, MQXQH, e il testo del messaggio dell'applicazione come richiamato dalla coda. (Il formato di MQXQH viene fornito in MQXQH.) Se si utilizzano i messaggi di riferimento, ossia i messaggi che contengono solo un'intestazione che punta a qualche altro oggetto da inviare, l'uscita del messaggio riconosce l'intestazione, MQRMH. Identifica l'oggetto, lo richiama in qualsiasi modo appropriato lo

aggiunge all'intestazione e lo passa all'MCA per la trasmissione all'MCA ricevente. All'MCA di ricezione, un'altra uscita del messaggio riconosce che questo messaggio è un messaggio di riferimento, estrae l'oggetto e trasmette l'intestazione alla coda di destinazione. Consultare “Messaggi di riferimento” a pagina 267 e “Esecuzione degli esempi del messaggio di riferimento” a pagina 142 per ulteriori informazioni sui messaggi di riferimento e su alcune uscite di messaggi di esempio che li gestiscono.

Le uscite messaggio possono restituire le seguenti risposte:

- Inviare il messaggio (GET exit). Il messaggio potrebbe essere stato modificato dall'uscita. (Questo restituisce MQXCC_OK.)
- Inserire il messaggio nella coda (uscita PUT). Il messaggio potrebbe essere stato modificato dall'uscita. (Questo restituisce MQXCC_OK.)
- Non elaborare il messaggio. Il messaggio viene inserito nella coda di messaggi non recapitabili (coda di messaggi non recapitati) dall'MCA.
- Chiudere il canale.
- Codice di ritorno errato, che causa la fine anomala dell'MCA.

Nota:

1. Le uscite messaggio vengono richiamate una volta per ogni messaggio completo trasferito, anche quando il messaggio è suddiviso in parti.
2. Nei sistemi UNIX , se si fornisce un'uscita messaggio per qualsiasi motivo, la conversione automatica degli ID utente in caratteri minuscoli non funziona. Vedere Sicurezza degli oggetti sui sistemi UNIX and Linux.
3. Un'uscita viene eseguita nello stesso thread dell'MCA stesso. Viene eseguito anche all'interno della stessa unità di lavoro (UOW) dell'MCA poiché utilizza lo stesso handle di connessione. Pertanto, tutte le chiamate effettuate nel punto di sincronizzazione vengono sottoposte a commit o a backout dal canale alla fine del batch. Ad esempio, un programma di uscita del messaggio del canale può inviare messaggi di notifica a un altro e questi messaggi vengono sottoposti a commit solo sulla coda quando viene eseguito il commit del batch contenente il messaggio originale.

   Pertanto, è possibile emettere chiamate MQI del punto di sincronizzazione da un programma di uscita del messaggio del canale.

Conversione del messaggio al di fuori dell'uscita del messaggio
Prima di richiamare l'uscita messaggio, l'MCA ricevente esegue alcune conversioni sul messaggio. Questo argomento descrive gli algoritmi utilizzati per eseguire la conversione.

Quali intestazioni vengono elaborate

Una routine di conversione viene eseguita nell'MCA del destinatario prima che venga richiamata l'uscita del messaggio. La routine di conversione inizia con l'intestazione MQXQH all'inizio del messaggio. La routine di conversione viene quindi eseguita tramite le intestazioni concatenate che seguono MQXQH, eseguendo la conversione dove necessario. Le intestazioni concatenate possono estendersi oltre l'offset contenuto nel parametro HeaderLength dei dati MQCXP passati all'uscita del messaggio del destinatario. Le seguenti intestazioni vengono convertite in loco:

- MQXQH (nome formato "MQXMIT ")
- MQMD (questa intestazione fa parte di MQXQH e non ha un nome formato)
- MQMDE (nome formato "MQHMDE ")
- MQDH (nome formato "MQHDIST ")
- MQWIH (nome formato "MQHWIH ")

Le seguenti intestazioni non vengono convertite, ma vengono superate mentre l'MCA continua a elaborare le intestazioni concatenate:

- MQDLH (nome formato "MQDEAD ")

- tutte le intestazioni con nomi formato che iniziano con i tre caratteri 'MQH' (ad esempio "MQHRF ") che non sono altrimenti menzionati

Modalità di elaborazione delle intestazioni

Il parametro Format di ogni intestazione WebSphere MQ viene letto da MCA. Il parametro Formato è di 8 byte all'interno dell'intestazione, che sono 8 caratteri a byte singolo contenenti un nome.

L'MCA interpreta quindi i dati che seguono ciascuna intestazione come se fossero del tipo denominato. Se il formato è il nome di un tipo di intestazione idoneo per la conversione dei dati di WebSphere MQ , viene convertito. Se è un altro nome che indica dati nonMQ (ad esempio MQFMT_NONE o MQFMT_STRING), l'MCA arresta l'elaborazione delle intestazioni.

Cos' è MQCXP HeaderLength?

Il parametro HeaderLength nei dati MQCXP forniti a un'uscita messaggio è la lunghezza totale delle intestazioni MQXQH (che include MQMD), MQMDE e MQDH all'inizio del messaggio. Queste intestazioni sono concatenate utilizzando i nomi e le lunghezze 'Formato'.

MQWIH

Le intestazioni concatenate possono estendersi oltre la HeaderLength nell'area dati utente. L'intestazione MQWIH, se presente, è una di quelle intestazioni che vengono visualizzate oltre HeaderLength.

Se è presente un'intestazione MQWIH nelle intestazioni concatenate, viene convertita in posizione prima che venga richiamata l'uscita del messaggio del destinatario.

Programma di uscita nuovo tentativo messaggio canale
L'uscita di nuovo tentativo del messaggio del canale viene richiamata quando un tentativo di aprire la coda di destinazione non riesce. È possibile utilizzare l'uscita per determinare in quali circostanze ritentare, quante volte ritentare e con quale frequenza.

Questa uscita viene richiamata anche all'estremità ricevente del canale all'avvio e alla chiusura MCA.

L'uscita del nuovo tentativo del messaggio del canale viene passata a un buffer dell'agent contenente l'intestazione della coda di trasmissione, MQXQH, e il testo del messaggio dell'applicazione come richiamato dalla coda. Il formato di MQXQH viene fornito in Panoramica per MQXQH.

L'uscita viene richiamata per tutti i codici di errore; l'uscita determina per quali codici di errore desidera che l'MCA esegua un nuovo tentativo, per quante volte e a quali intervalli. (Il valore del conteggio dei tentativi dei messaggi impostato quando il canale è stato definito viene passato all'exit in MQCD, ma l'exit può ignorare questo valore.)

Il campo MsgRetryCount in MQCXP viene incrementato dall'MCA ogni volta che viene richiamata l'uscita e l'uscita restituisce MQXCC_OK con il tempo di attesa contenuto nel campo Intervallo MsgRetrydi MQCXP o MQXCC_SUPPRESS_FUNCTION. I tentativi continuano indefinitamente fino a quando l'uscita non restituisce MQXCC_SUPPRESS_FUNCTION nel campo ExitResponse di MQCXP. Consultare MQCXP per informazioni sull'azione eseguita da MCA per questi codici di completamento.

Se tutti i tentativi hanno esito negativo, il messaggio viene scritto nella coda di messaggi non recapitabili. Se non è disponibile una coda di messaggi non instradabili, il canale si arresta.

Se non si definisce un'uscita di nuovo tentativo del messaggio per un canale e si verifica un errore che è probabile sia temporaneo, ad esempio MQRC_Q_FULL, l'MCA utilizza il conteggio di nuovi tentativi del messaggio e gli intervalli di nuovi tentativi del messaggio impostati quando il canale è stato definito. Se l'errore è di natura più permanente e non è stato definito un programma di uscita per gestirlo, il messaggio viene scritto nella coda di messaggi non recapitabili.

Programma di uscita di definizione automatica del canale
L'uscita di definizione automatica del canale può essere utilizzata quando viene ricevuta una richiesta di avviare un canale ricevente o di connessione server, ma non esiste alcuna definizione per tale canale (non

per WebSphere MQ per z/OS). Può anche essere richiamato su tutte le piattaforme per i canali mittente e ricevente del cluster per consentire la modifica della definizione per un'istanza del canale.

L'uscita di definizione automatica del canale può essere richiamata su tutte le piattaforme ad eccezione di z/OS quando viene ricevuta una richiesta di avvio di un canale ricevente o di connessione server ma non esiste alcuna definizione di canale. È possibile utilizzarla per modificare la definizione predefinita fornita per un canale ricevente o di connessione server definito automaticamente, SYSTEM.AUTO.RECEIVERo SYSTEM.AUTO.SVRCON. Consultare Preparazione dei canali per una descrizione del modo in cui le definizioni dei canali possono essere create automaticamente.

L'uscita di definizione automatica del canale può essere richiamata anche quando viene ricevuta una richiesta di avvio di un canale mittente del cluster. Può essere richiamato per i canali mittente cluster e ricevente cluster per consentire la modifica della definizione per questa istanza del canale. In questo caso, l'exit si applica anche a WebSphere MQ per z/OS. Un uso comune dell'uscita di definizione automatica del canale consiste nel modificare i nomi delle uscite dei messaggi (MSGEXIT, RCVEXIT, SCYEXIT e SENDEXIT) perché i nomi delle uscite hanno formati differenti su piattaforme differenti. Se non viene specificata alcuna uscita di definizione automatica del canale, il comportamento predefinito su z/OS consiste nell'esaminare un nome di uscita distribuito nel formato *[path]/libraryname(function)* e utilizzare fino a otto caratteri di funzione, se presenti, o nome libreria. Su z/OS, un programma di uscita di definizione automatica del canale deve modificare i campi indicati da MsgExitPtr, MsgUserDataPtr, SendExitPtr, SendUserDataPtr, ReceiveExitPtr, e ReceiveUserDataPtr, piuttosto che MsgExit, MsgUserData, SendExit, SendUserDati, campi ReceiveExit e ReceiveUserDati stessi.

Per ulteriori informazioni, consultare Definizione automatica dei canali.

Come per altre uscite di canale, l'elenco dei parametri è:

```
MQ_CHANNEL_AUTO_DEF_EXIT (ChannelExitParms, ChannelDefinition)
```

`ChannelExitParms` sono descritti in MQCXP. `ChannelDefinition` è descritto in MQCD.

MQCD contiene i valori utilizzati nella definizione di canale predefinita se non vengono modificati dall'exit. L'uscita può modificare solo un sottoinsieme di campi; consultare MQ_CHANNEL_AUTO_DEF_EXIT. Tuttavia, il tentativo di modificare altri campi non causa un errore.

L'uscita di definizione automatica del canale restituisce una risposta MQXCC_OK o MQXCC_SUPPRESS_FUNCTION. Se nessuna di queste risposte viene restituita, l'MCA continua l'elaborazione come se fosse stata restituita MQXCC_SUPPRESS_FUNCTION. In altre parole, la definizione automatica viene abbandonata, non viene creata alcuna nuova definizione di canale e il canale non può essere avviato.

Compilazione di programmi di uscita canale su Windows, sistemi UNIX and Linux

Utilizzare i seguenti esempi per facilitare la compilazione di programmi di uscita canale per sistemi Windows, UNIX and Linux .

Finestre

Windows

Il comando compilatore e linker per i programmi channel - exit su Windows:

```
cl.exe /Ic:\mqm\tools\c\include /nologo /c myexit.c
link.exe /nologo /dll myexit.obj /def:myexit.def /out:myexit.dll
```

Sistemi UNIX e Linux

Linux UNIX

In questi esempi `exit` è il nome della libreria e `ChannelExit` è il nome della funzione. Su AIX il file di esportazione è denominato `exit.exp`. Questi nomi vengono utilizzati dalla definizione di canale per fare riferimento al programma di uscita utilizzando il formato descritto in Definizione canale MQCD. Consultare anche il parametro MSGEXIT del comando DEFINE CHANNEL .

Esempi di comandi del compilatore e del linker per le uscite dei canali su AIX:

```
$ xlc_r -q64 -e MQStart -bE:exit.exp -bM:SRE -o /var/mqm/exits64/exit
exit.c -I/usr/mqm/inc
```

Esempi di comandi del compilatore e del linker per uscite di canale su 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
```

Esempi di comandi del compilatore e del linker per le uscite del canale sulle piattaforme Linux in cui il gestore code è a 32 bit:

```
$ gcc -shared -fPIC -o /var/mqm/exits/exit exit.c -I/opt/mqm/inc
```

Esempi di comandi del compilatore e del linker per le uscite del canale su piattaforme Linux in cui il gestore code è a 64 bit:

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
```

Esempi di comandi del compilatore e del linker per uscite di canale su Solaris:

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
-R/usr/lib/64 -lsocket -lnsl -ldl
```

Sul client, è possibile utilizzare un'uscita a 32 bit o a 64 bit. Questa uscita deve essere collegata a mqic_r.

Su AIX, tutte le funzioni richiamate da IBM WebSphere MQ devono essere esportate. Un file di esportazione di esempio per questo file make:

```
#!
channelExit
MQStart
```

Configurazione delle uscite canale

Per richiamare l'uscita del canale, è necessario denominarlo nella definizione del canale.

Le uscite canale devono essere denominate nella definizione del canale. È possibile eseguire questa denominazione quando si definiscono per la prima volta i canali oppure è possibile aggiungere le informazioni in un secondo momento utilizzando, ad esempio, il comando MQSC ALTER CHANNEL. È anche possibile fornire i nomi di uscita del canale nella struttura dati del canale MQCD. Il formato del nome di uscita dipende dalla piattaforma IBM WebSphere MQ ; per informazioni, consultare Comandi MQCD o Script (MQSC) .

Se la definizione del canale non contiene un nome di programma di uscita utente, l'uscita utente non viene richiamata.

L'uscita di definizione automatica del canale è la proprietà del gestore code, non il singolo canale. Affinché questa uscita possa essere richiamata, deve essere denominata nella definizione del gestore code. Per modificare una definizione del gestore code, utilizzare il comando MQSC ALTER QMGR.

Scrittura delle uscite di conversione dati

Questa raccolta di argomenti contiene informazioni su come scrivere le uscite di conversione dati.

Nota: Non supportata in MQSeries per VSE/ESA.

Quando si esegue un MQPUT, l'applicazione crea il descrittore del messaggio (MQMD) del messaggio. Poiché WebSphere MQ deve essere in grado di comprendere il contenuto di MQMD indipendentemente dalla piattaforma su cui viene creato, viene convertito automaticamente dal sistema.

I dati dell'applicazione, tuttavia, non vengono convertiti automaticamente. Se i dati carattere vengono scambiati tra le piattaforme in cui i campi *CodedCharSetId* e *Encoding* differiscono, ad esempio, tra ASCII e EBCDIC, l'applicazione deve organizzare la conversione del messaggio. La conversione dei dati dell'applicazione può essere eseguita dal gestore code stesso o da un programma di uscita utente, denominato *uscita di conversione dati*. Il gestore code può eseguire la conversione dei dati autonomamente, utilizzando una delle routine di conversione integrate, se i dati dell'applicazione si trovano in uno dei formati integrati (come MQFMT_STRING). Questo argomento contiene informazioni sulla funzione di uscita di conversione dati fornita da WebSphere MQ quando i dati dell'applicazione non sono in un formato integrato.

Il controllo può essere passato all'uscita di conversione dati durante una chiamata MQGET. Ciò evita la conversione su diverse piattaforme prima di raggiungere la destinazione finale. Tuttavia, se la destinazione finale è una piattaforma che non supporta la conversione dei dati su MQGET, è necessario specificare CONVERT (YES) sul canale mittente che invia i dati alla destinazione finale. Ciò garantisce che WebSphere MQ converta i dati durante la trasmissione. In questo caso, l'uscita di conversione dati deve risiedere sul sistema in cui è definito il canale mittente.

La chiamata MQGET viene emessa direttamente dall'applicazione. Impostare i campi *CodedCharSetId* e *Encoding* in MQMD sulla serie di caratteri e sulla codifica richieste. Se l'applicazione utilizza la stessa serie di caratteri e la stessa codifica del gestore code, impostare *CodedCharSetId* su MQCCSI_Q_MGR e *Encoding* su MQENC_NATIVE. Una volta completata la chiamata MQGET, questi campi hanno i valori appropriati per i dati del messaggio restituiti. Potrebbero essere diversi dai valori richiesti se la conversione non ha avuto esito positivo. L'applicazione deve reimpostare questi campi sui valori richiesti prima di ogni chiamata MQGET.

Le condizioni richieste per l'uscita di conversione dati da chiamare sono definite per la chiamata MQGET in MQGET.

Per una descrizione dei parametri passati all'uscita di conversione dati e per le note di utilizzo dettagliate, consultare Conversione dati per la chiamata MQ_DATA_CONV_EXIT e la struttura MQDXP.

I programmi che convertono i dati dell'applicazione tra diverse codifiche macchina e CCSID devono essere conformi a DCI (Data Conversion Interface) WebSphere MQ .

Con l'introduzione dei client multicast, le uscite API e le uscite di conversione dati devono essere in grado di essere eseguite sul lato client perché alcuni messaggi potrebbero non passare attraverso il gestore code. Le librerie riportate di seguito fanno ora parte dei pacchetti client e dei pacchetti server:

Tabella 56. Librerie che si trovano ora nei package client e server

Sistema operativo	**Librerie**
Finestre	32 bit & 64 bit: mqm.dll & mqm.pdb
Linux & HP-UX	32 bit & 64 bit: libmqm.so & libmqm_r.so
AIX	32 bit & 64 bit: libmqm.a & libmqm_r.a
Solaris	32 bit & 64 bit: libmqm.so

Richiamo dell'uscita di conversione dati

Un'uscita di conversione dati è un'uscita scritta dall'utente che riceve il controllo durante l'elaborazione di una chiamata MQGET.

L'uscita viene richiamata se si verificano le seguenti condizioni:

- L'opzione MQGMO_CONVERT è specificata nella chiamata MQGET.
- Alcuni o tutti i dati del messaggio non si trovano nella serie di caratteri o nella codifica richiesta.
- Il campo *Format* nella struttura MQMD associata al messaggio non è MQFMT_NONE.
- Il valore *BufferLength* specificato nella chiamata MQGET non è zero.
- La lunghezza dei dati del messaggio è diversa da zero.

- Il messaggio contiene dati con un formato definito dall'utente. Il formato definito dall'utente può occupare l'intero messaggio o essere preceduto da uno o più formati incorporati. Ad esempio, il formato definito dall'utente potrebbe essere preceduto da un formato MQFMT_DEAD_LETTER_HEADER. L'uscita viene richiamata per convertire solo il formato definito dall'utente; il gestore code converte tutti i formati integrati che precedono il formato definito dall'utente.

  Un'uscita scritta dall'utente può essere richiamata anche per convertire un formato integrato, ma ciò si verifica solo se le routine di conversione integrate non possono convertire il formato incorporato con esito positivo.

Ci sono altre condizioni, descritte completamente nelle note di utilizzo della chiamata MQ_DATA_CONV_EXIT in MQ_DATA_CONV_EXIT.

Consultare MQGET per i dettagli della chiamata MQGET. Le uscite di conversione dati non possono utilizzare chiamate MQI diverse da MQXCNVC.

Una nuova copia dell'uscita viene caricata quando un'applicazione tenta di recuperare il primo messaggio che utilizza *Format* da quando l'applicazione si è connessa al gestore code. Una nuova copia potrebbe essere caricata anche in altre occasioni se il gestore code ha eliminato una copia precedentemente caricata.

L'uscita conversione dati viene eseguita in un ambiente simile a quello del programma che ha emesso la chiamata MQGET. Oltre alle applicazioni utente, il programma può essere un MCA (message channel agent) che invia messaggi a un gestore code di destinazione che non supporta la conversione dei messaggi. L'ambiente include lo spazio di indirizzo e il profilo utente, dove applicabile. L'uscita non può compromettere l'integrità del gestore code, perché non viene eseguita nell'ambiente del gestore code.

Scrittura di un'uscita di conversione dati per WebSphere MQ su sistemi UNIX and Linux

Informazioni sui passi da considerare durante la scrittura dei programmi di uscita conversione dati per WebSphere MQ su sistemi UNIX and Linux .

Eseguire queste operazioni:

1. Denominare il formato del messaggio. Il nome deve rientrare nel campo *Format* di MQMD e deve essere in maiuscolo, ad esempio MYFORMAT. Il nome *Format* non deve contenere spazi iniziali. Gli spazi finali vengono ignorati. Il nome dell'oggetto non deve contenere più di otto caratteri non vuoti, poiché *Format* è lungo solo otto caratteri. Ricordarsi di utilizzare questo nome ogni volta che si invia un messaggio.

   Se l'uscita di conversione dati viene utilizzata in un ambiente con thread, l'oggetto caricabile deve essere seguito da _r per indicare che si tratta di una versione con thread.
2. Creare una struttura per rappresentare il proprio messaggio. Vedere Sintassi valida per un esempio.
3. Eseguire questa struttura tramite il comando `crtmqcvx` per creare un frammento di codice per l'uscita di conversione dati.

   Le funzioni generate dal comando `crtmqcvx` utilizzano macro che presuppongono che tutte le strutture siano impacchettate; in caso contrario, modificarle.
4. Copiare il file di origine della struttura fornito, ridenominandolo nel nome del formato del messaggio impostato nel passo “1” a pagina 419. Il file di origine della struttura e la copia sono di sola lettura.

   Il file di origine della struttura è denominato amqsvfc0.c.
5. In WebSphere MQ per AIX, viene fornito anche un file di esportazione della struttura denominato amqsvfc.exp . Copiare questo file, ridenominandolo MYFORMAT.EXP.
6. La struttura include un file di intestazione di esempio, amqsvmha.h, nella directory *MQ_INSTALLATION_PATH*/inc, dove *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ . Verificare che il percorso di inclusione punti a questa directory per selezionare questo file.

   Il file amqsvmha.h contiene le macro utilizzate dal codice generato dal comando `crtmqcvx` . Se la struttura da convertire contiene dati carattere, queste macro richiamano MQXCNVC.
7. Trovare le seguenti caselle di commento nel file di origine e inserire il codice come descritto:

a. Verso la fine del file di origine, una casella di commento inizia con:

```
/* Insert the functions produced by the data-conversion exit */
```

In questo punto, inserire il frammento di codice generato nel passo “3” a pagina 419.

b. Vicino al centro del file di origine, una casella di commento inizia con:

```
/* Insert calls to the code fragments to convert the format's */
```

Questo è seguito da una chiamata di commento alla funzione `ConverttagSTRUCT`.

Modificare il nome della funzione nel nome della funzione aggiunta nel passaggio “7.a” a pagina 420. Eliminare i caratteri di commento per attivare la funzione. Se ci sono diverse funzioni, creare chiamate per ognuna di esse.

c. All'inizio del file di origine, una casella di commento inizia con:

```
/* Insert the function prototypes for the functions produced by */
```

Qui, inserire le istruzioni del prototipo della funzione per le funzioni aggiunte nel passo “3” a pagina 419 precedente.

8. Compilare l'uscita come una libreria condivisa, utilizzando MQStart come punto di ingresso. Per fare ciò, consultare “Compilazione delle uscite di conversione dati su sistemi UNIX and Linux” a pagina 420.
9. Inserire l'emissione nella directory di uscita. La directory di uscita predefinita è `/var/mqm/exits` per i sistemi a 32 bit e `/var/mqm/exits64`per i sistemi a 64 bit. È possibile modificare tali directory nel file qm.ini o mqclient.ini . Questo percorso può essere impostato per ogni gestore code e l'uscita viene ricercata solo in tale percorso o percorsi.

Nota:

1. Se `crtmqcvx` utilizza strutture compresse, tutte le applicazioni WebSphere MQ devono essere compilate in questo modo.
2. I programmi di uscita conversione dati devono essere rientranti.
3. MQXCNVC è la *sola* chiamata MQI che può essere emessa da un'uscita di conversione dati.

Compilazione delle uscite di conversione dati su sistemi UNIX and Linux
Esempi di come compilare un'uscita di conversione dati su sistemi UNIX and Linux .

Su tutte le piattaforme, il punto di accesso al modulo è MQStart.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

AIX

Compilare il codice di origine di uscita immettendo uno dei comandi seguenti:

Applicazioni a 32 bit

Senza thread

```
cc -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Threaded

```
xlc_r -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT_r \
     MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Applicazioni a 64 bit

Senza thread

```
cc -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Threaded

```
xlc_r -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT_r \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Piattaforma HP-UX Itanium

Compilare e collegare il codice di origine di uscita immettendo una delle seguenti serie di comandi:

Applicazioni a 32 bit

Senza thread

Compilare il codice sorgente di uscita:

```
c89 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Collegare l'oggetto di uscita:

```
ld +b: -b MYFORMAT.o +ee MQStart -o  \
    /var/mqm/exits/MYFORMAT -L/usr/lib/hpux32
rm MYFORMAT.o
```

Threaded

Compilare il codice sorgente di uscita:

```
c89 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Collegare l'oggetto di uscita:

```
ld +b: -b MYFORMAT.o +ee MQStart -o  \
    /var/mqm/exits/MYFORMAT_r -L/usr/lib/hpux32 \
    -lpthread
rm MYFORMAT.o
```

Applicazioni a 64 bit

Senza thread

Compilare il codice sorgente di uscita:

```
 c89 +DD64 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Collegare l'oggetto di uscita:

```
ld -b MYFORMAT.o +ee MQStart \
    -o  /var/mqm/exits64/MYFORMAT \
    -L/usr/lib/hpux64
rm MYFORMAT.o
```

Threaded

Compilare il codice sorgente di uscita:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Collegare l'oggetto di uscita:

```
ld -b MYFORMAT.o +ee MQStart  \
    -o  /var/mqm/exits64/MYFORMAT_r  \
    -L/usr/lib/hpux64 -lpthread
rm MYFORMAT.o
```

Linux

Compilare il codice di origine di uscita immettendo uno dei comandi seguenti:

Applicazioni a 31 bit

Senza thread

```
 gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c \
      -IMQ_INSTALLATION_PATH/inc
```

Threaded

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

Applicazioni a 32 bit

Senza thread

```
 gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c
      -IMQ_INSTALLATION_PATH/inc
```

Threaded

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

Applicazioni a 64 bit

Senza thread

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT MYFORMAT.c
      -IMQ_INSTALLATION_PATH/inc
```

Threaded

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT_r MYFORMAT.c
      -IMQ_INSTALLATION_PATH/inc
```

Solaris

Compilare il codice di origine di uscita immettendo uno dei comandi seguenti:

Applicazioni a 32 bit

Piattaforma 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
```

Piattaforma x86-64

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

Applicazioni a 64 bit

Piattaforma 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
```

Piattaforma x86-64

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/64  -lsocket -lnsl -ldl
```

Scrittura di un'uscita di conversione dati per WebSphere MQ per Windows

Informazioni sulle fasi da considerare durante la scrittura di programmi di uscita di conversione dati per WebSphere MQ per Windows.

Eseguire queste operazioni:

1. Denominare il formato del messaggio. Il nome deve rientrare nel campo *Format* di MQMD. Il nome *Format* non deve contenere spazi iniziali. Gli spazi finali vengono ignorati. Il nome dell'oggetto non deve contenere più di otto caratteri non vuoti, poiché *Format* è lungo solo otto caratteri.

   Nella directory degli esempi viene fornito anche un file .DEF denominato amqsvfcn.def , *MQ_INSTALLATION_PATH*\Tools\C\Samples. *MQ_INSTALLATION_PATH* è la directory in cui è installato WebSphere MQ . Prendere una copia di questo file e ridenominarlo, ad esempio, in MYFORMAT.DEF. Assicurarsi che il nome della DLL da creare e il nome specificato in MYFORMAT.DEF sono le stesse. Sovrascrivere il nome FORMAT1 in MYFORMAT.DEF con il nome del nuovo formato.

   Ricordarsi di utilizzare questo nome ogni volta che si invia un messaggio.
2. Creare una struttura per rappresentare il proprio messaggio. Vedere Sintassi valida per un esempio.
3. Eseguire questa struttura tramite il comando `crtmqcvx` per creare un frammento di codice per l'uscita di conversione dati.

   Le funzioni generate dal comando CRTMQCVX utilizzano macro scritte presupponendo che tutte le strutture siano impacchettate; in caso contrario, modificarle.
4. Copiare il file di origine della struttura fornito, amqsvfc0.c, ridenominandolo con il nome del formato del messaggio impostato nel passo “1” a pagina 423.

   amqsvfc0.c si trova in *MQ_INSTALLATION_PATH*\Tools\C\Samples dove *MQ_INSTALLATION_PATH* è la directory in cui è installato WebSphere MQ . La directory di installazione predefinita è `C:\Program Files\IBM\WebSphere MQ`.

   La struttura include un file di intestazione di esempio amqsvmha.h nella directory *MQ_INSTALLATION_PATH*\Tools\C\include . Verificare che il percorso di inclusione punti a questa directory per selezionare questo file.

   Il file amqsvmha.h contiene le macro utilizzate dal codice generato dal comando CRTMQCVX. Se la struttura da convertire contiene dati carattere, queste macro richiamano MQXCNVC.
5. Trovare le seguenti caselle di commento nel file di origine e inserire il codice come descritto:

   a. Verso la fine del file di origine, una casella di commento inizia con:

   ```
   /* Insert the functions produced by the data-conversion exit */
   ```

   In questo punto, inserire il frammento di codice generato nel passo “3” a pagina 423.

   b. Vicino al centro del file di origine, una casella di commento inizia con:

   ```
   /* Insert calls to the code fragments to convert the format's */
   ```

   Questo è seguito da una chiamata di commento alla funzione `ConverttagSTRUCT`.

Modificare il nome della funzione nel nome della funzione aggiunta nel passaggio “5.a” a pagina 423. Eliminare i caratteri di commento per attivare la funzione. Se ci sono diverse funzioni, creare chiamate per ognuna di esse.

c. All'inizio del file di origine, una casella di commento inizia con:

```
/* Insert the function prototypes for the functions produced by */
```

Qui, inserire le istruzioni del prototipo di funzione per le funzioni aggiunte nel passo “3” a pagina 423.

6. Creare il seguente file di comandi:

```
cl -I MQ_INSTALLATION_PATH\Tools\C\Include -Tp   \
MYFORMAT.C

MYFORMAT.DEF
```

dove *MQ_INSTALLATION_PATH* è la directory in cui è installato WebSphere MQ .

7. Immettere il file dei comandi per compilare l'uscita come un file DLL.
8. Inserire l'output nella sottodirectory di uscita al di sotto della directory di dati WebSphere MQ . La directory predefinita per l'installazione delle uscite su sistemi a 32 bit è *MQ_DATA_PATH*\Exits e per sistemi a 64 bit è *MQ_DATA_PATH*\Exits64

Il percorso utilizzato per ricercare le uscite di conversione dati viene fornito nel registro. La cartella del registro è:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\ClientExitPa
th\
```

e la chiave di registro è ExitsDefaultPath. Questo percorso può essere impostato per ogni gestore code e l'uscita viene ricercata solo in tale percorso o percorsi.

Nota:

1. Se CRTMQCVX utilizza strutture compresse, tutte le applicazioni WebSphere MQ devono essere compilate in questo modo.
2. I programmi di uscita conversione dati devono essere rientranti.
3. MQXCNVC è la *sola* chiamata MQI che può essere emessa da un'uscita di conversione dati.

Uscire e passare ai file di caricamento sui sistemi operativi Windows

I processi del gestore code IBM WebSphere MQ for Windows Version 7.5 sono a 32 bit. Di conseguenza, quando si utilizzano applicazioni a 64 bit, alcuni tipi di file di uscita e di file di caricamento degli switch XA devono avere anche una versione a 32 bit disponibile per l'utilizzo da parte del gestore code. Se la versione a 32 bit del file di caricamento dell'uscita o dello switch XA è richiesta e non è disponibile, la chiamata o il comando API pertinente non riesce.

Sono supportati due attributi in qm.ini file per *ExitPath*. Questi sono ExitsDefaultPath=*MQ_INSTALLATION_PATH*\exits e ExitsDefaultPath64=*MQ_INSTALLATION_PATH*\exits64 *MQ_INSTALLATION_PATH* . rappresenta la directory di alto livello in cui è installato WebSphere MQ . L'utilizzo di queste informazioni garantisce che sia possibile trovare la libreria appropriata. Se in un cluster WebSphere MQ viene utilizzata un'uscita, ciò garantisce anche che sia possibile trovare la libreria appropriata su un sistema remoto.

La seguente tabella elenca i diversi tipi di file di caricamento Exit e Switch e indica se sono richieste versioni a 32 bit o a 64 bit o entrambe, in base all'utilizzo di applicazioni a 32 bit o a 64 bit:

Tipi di file	**Applicazioni a 32 bit**	**Applicazioni a 64 bit**
uscita incrociata API	32 bit	32 bit e 64 bit

Tipi di file	Applicazioni a 32 bit	Applicazioni a 64 bit
Uscita conversione dati	32 bit	64 bit
Uscite canale server (tutti i tipi)	32 bit	32 bit
Uscite canale client (tutti i tipi)	32 bit	64 bit
Uscita di servizio installabile	32 bit	32 bit
Modulo di traccia del servizio	32 bit	32 bit e 64 bit
Uscita WLM cluster	32 bit	32 bit
Uscita instradamento pubblicazione / sottoscrizione	32 bit	32 bit
File di caricamento switch database	32 bit	32 bit e 64 bit
Librerie AX di External Transaction Manager	32 bit	64 bit

Riferimento alle definizioni di connessione mediante un'uscita di pre - connessione da un repository

WebSphere MQ I client MQI possono essere configurati per ricercare un repository per ottenere definizioni di connessione utilizzando una libreria di uscita di pre - connessione.

Introduzione

Un'applicazione client può connettersi a un gestore code utilizzando CCDT (client channel definition tables). In genere, il file CCDT si trova su un server di file di rete centrale e dispone di client che vi fanno riferimento. Poiché è difficile gestire e gestire varie applicazioni client che fanno riferimento al file CCDT, un approccio flessibile consiste nel memorizzare le definizioni client in un repository globale come una directory LDAP, un WebSphere Registry and Repository o qualsiasi altro repository. La memorizzazione delle definizioni di connessione client in un repository rende più semplice la gestione delle definizioni di connessione client e le applicazioni possono accedere alle definizioni di connessione client corrette e più aggiornate.

Durante l'esecuzione della chiamata MQCONN/X, IBM WebSphere MQ MQI client carica una libreria di uscita pre - connessione specificata dall'applicazione e richiama una funzione di uscita per richiamare le definizioni di connessione. Le definizioni di connessione richiamate vengono quindi utilizzate per stabilire una connessione a un gestore code. I dettagli della libreria di uscita e della funzione da richiamare sono specificati nel file di configurazione mqclient.ini .

Sintassi

`void MQ_PRECONNECT_EXIT` (*pExitParms*, *pQMgrName*, *ppConnectOpts*, *pCompCode*, *pReason*);

Parametri

pExitParms

Tipo: input / output PMQNXP

La struttura del parametro di uscita **`PreConnection`** .

La struttura è assegnata e gestita dal chiamante dell'uscita.

pQMgrNome

Tipo: input/output PMQCHAR

Il nome del gestore code.

In fase di input, questo parametro è la stringa di filtro fornita alla chiamata API MQCONN tramite il parametro **QMgrName** . Questo campo potrebbe essere vuoto, esplicito o contenere determinati caratteri jolly. Il campo viene modificato dall'uscita. Il parametro è NULL quando l'uscita viene chiamata con MQXR_TERM.

ppConnectOpzioni
Tipo: ppConnectOpts input/output

Opzioni che controllano l'azione di MQCONNX.

Questo è un puntatore a una struttura di opzioni di connessione MQCNO che controlla l'azione della chiamata API MQCONN. Il parametro è NULL quando l'uscita viene chiamata con MQXR_TERM. Il client MQI fornisce sempre una struttura MQCNO all'uscita, anche se non è stata originariamente fornita dall'applicazione. Se un'applicazione fornisce una struttura MQCNO, il client effettua un duplicato per inoltrarlo all'uscita in cui viene modificato. Il client conserva la proprietà di MQCNO.

Un MQCD a cui si fa riferimento tramite MQCNO ha la precedenza su qualsiasi definizione di connessione fornita tramite l'array. Il client utilizza la struttura MQCNO per connettersi al gestore code e gli altri vengono ignorati.

Codice pComp
Tipo: input / output PMQLONG

Codice di completamento.

Puntatore a un MQLONG che riceve il codice di completamento delle uscite. Deve essere uno dei seguenti valori:

- MQCC_OK - Completamento riuscito
- MQCC_WARNING - Avviso (completamento parziale)
- MQCC_FAILED - Chiamata non riuscita

pReason
Tipo: input / output PMQLONG

Codice di qualificazione motivo pComp.

Puntatore ad un MQLONG che riceve il codice motivo di uscita. Se il codice di completamento è MQCC_OK, l'unico valore valido è:

- MQRC_NONE - (0, x '000') Nessun motivo per la notifica.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo 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*/
```

Stanza PreConnect del file di configurazione client

Utilizzare la stanza PreConnect per configurare l'uscita PreConnect nel file `mqclient.ini` .

I seguenti attributi possono essere inclusi nella stanza PreConnect :

Data=*< URL>*
URL del repository in cui sono memorizzate le definizioni di connessione. Ad esempio, quando si usa un server LDAP:

Dati = ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com

Function=*<myFunc>*
Nome del punto di immissione funzionale nella libreria che contiene il codice di uscita PreConnect .

La definizione della funzione aderisce al prototipo di uscita PreConnect MQ_PRECONNECT_EXIT.

La lunghezza massima di questo campo è MQ_EXIT_NAME_LENGTH.

Module=*< amqldapi>*
Il nome del modulo contenente il codice di uscita API.

Se questo campo contiene il nome percorso completo del modulo, viene utilizzato così com' è.

Sequence=*< numero_sequenza>*
La sequenza in cui questa uscita viene chiamata rispetto ad altre uscite. Un'uscita con un numero di sequenza basso viene richiamata prima di un'uscita con un numero di sequenza più alto. Non c'è bisogno che la numerazione di sequenza delle uscite sia continua; una sequenza di 1, 2, 3 ha lo stesso risultato di una sequenza di 7, 42, 1096. Questo attributo è un valore numerico senza segno.

Più stanze PreConnect possono essere definite all'interno del file `mqclient.ini` . L'ordine di elaborazione di ciascuna uscita è determinato dall'attributo Sequenza della stanza.

Scrittura e compilazione di uscite di pubblicazione

È possibile configurare un'uscita di pubblicazione sul gestore code per modificare i contenuti di un messaggio pubblicato prima che venga ricevuto dai sottoscrittori. È anche possibile modificare l'intestazione del messaggio o non consegnare il messaggio a una sottoscrizione.

Le uscite di pubblicazione non sono supportate in z/OS.

È possibile utilizzare l'uscita di pubblicazione per esaminare e modificare i messaggi consegnati ai sottoscrittori:

- Esaminare il contenuto di un messaggio pubblicato per ciascun sottoscrittore
- Modificare il contenuto di un messaggio pubblicato per ciascun sottoscrittore
- Modificare la coda in cui viene inserito un messaggio
- Arrestare la consegna di un messaggio a un sottoscrittore

Scrittura di un'uscita di pubblicazione

Utilizzare la procedura riportata in “Scrittura e compilazione di uscite e servizi installabili” a pagina 376per scrivere e compilare l'uscita.

Il provider dell'uscita di pubblicazione definisce le operazioni dell'uscita. L'uscita, tuttavia, deve essere conforme alle regole definite in MQPSXP.

WebSphere MQ non fornisce un'implementazione del punto di ingresso MQ_PUBLISH_EXIT. Fornisce una dichiarazione typedef di linguaggio C. Utilizzare typedef per dichiarare correttamente i parametri in un'uscita scritta dall'utente. Il seguente esempio illustra come utilizzare la dichiarazione 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 */
}
```

L'uscita di pubblicazione viene eseguita nel processo del gestore code, come risultato delle seguenti operazioni:

- Un'operazione di pubblicazione in cui un messaggio viene consegnato a uno o più sottoscrittori

- Un'operazione di sottoscrizione in cui vengono consegnati uno o più messaggi conservati
- Un'operazione di richiesta di sottoscrizione in cui vengono consegnati uno o più messaggi conservati

Se l'uscita di pubblicazione viene richiamata per una connessione, la prima volta che viene richiamata viene impostato un codice *ExitReason* di MQXR_INIT . Prima che la connessione si disconnetta dopo aver utilizzato un'uscita di pubblicazione, l'uscita viene richiamata con un codice *ExitReason* di MQXR_TERM.

Se l'uscita di pubblicazione è configurata, ma non può essere caricata quando il gestore code viene avviato, le operazioni di pubblicazione / sottoscrizione dei messaggi sono inibite per il gestore code. È necessario risolvere il problema o riavviare il gestore code prima di riabilitare la messaggistica di pubblicazione / sottoscrizione.

Ogni connessione WebSphere MQ che richiede l'uscita di pubblicazione potrebbe non riuscire a caricare o inizializzare l'uscita. Se l'uscita non viene caricata o inizializzata, le operazioni di pubblicazione / sottoscrizione che richiedono l'uscita di pubblicazione vengono disabilitate per tale connessione. Le operazioni non riescono con il codice di errore WebSphere MQ MQRC_PUBLISH_EXIT_ERROR.

Il contesto in cui viene richiamata l'uscita di pubblicazione è la connessione di un'applicazione al gestore code. Un'area dati utente viene gestita dal gestore code per ogni connessione che sta eseguendo operazioni di pubblicazione. L'uscita può conservare le informazioni nell'area dati utente per ogni connessione.

Un'uscita di pubblicazione può utilizzare alcune chiamate MQI. Può utilizzare solo quelle chiamate MQI che gestiscono le proprietà del messaggio. Le chiamate sono:

- MQBUFMH
- MQCRTMH
- MQDLTMH
- MQDLTMP
- MQMHBUF
- MQINQMP
- MQSETMP

Se l'uscita di pubblicazione modifica il gestore code di destinazione o il nome della coda, non viene eseguito alcun nuovo controllo di autorizzazione.

Compilazione di un'uscita di pubblicazione

L'uscita di pubblicazione è una libreria caricata dinamicamente; può essere considerata come un'uscita canale. Per informazioni sulla compilazione delle uscite, consultare “Scrittura e compilazione di uscite e servizi installabili” a pagina 376.

Uscita di pubblicazione di esempio

Il programma di uscita di esempio è denominato amqspse0.c. Scrive un messaggio diverso in un file di log a seconda che l'exit sia stata richiamata per le operazioni di inizializzazione, pubblicazione o terminazione. Dimostra inoltre l'utilizzo del campo dell'area utente di uscita per allocare e liberare la memoria in modo appropriato.

Configurazione delle uscite di pubblicazione

È necessario definire alcuni attributi per configurare un'uscita di pubblicazione.

Su Windows e Linux è possibile utilizzare WebSphere MQ Explorer per definire gli attributi. Gli attributi sono definiti nella pagina delle proprietà del gestore code, in Pubblicazione / Sottoscrizione.

Per configurare l'uscita di pubblicazione nel file qm.ini su sistemi UNIX e Linux , creare una sezione denominata PublishSubscribe. La stanza PublishSubscribe ha i seguenti attributi:

PublishExitPath=[path]|*module_name*
Il nome e il percorso del modulo contenente il codice di uscita di pubblicazione. La lunghezza massima di questo campo è MQ_EXIT_NAME_LENGTH. L'impostazione predefinita è nessuna uscita di pubblicazione.

PublishExitFunction=*function_name*
Il nome del punto di ingresso della funzione nel modulo che contiene il codice di uscita pubblicazione. La lunghezza massima di questo campo è MQ_EXIT_NAME_LENGTH.

PublishExitData=*string*
Se il gestore code sta richiamando un'uscita di pubblicazione, passa una struttura MQPSXP come input. I dati specificati utilizzando l'attributo *PublishExitData* vengono forniti nel campo *ExitData* della struttura. La stringa può avere una lunghezza massima di MQ_EXIT_DATA_LENGTH caratteri. Il valore predefinito è 32 caratteri vuoti.

Scrittura e compilazione delle uscite del carico di lavoro del cluster

Scrivere un programma di uscita del carico di lavoro del cluster per personalizzare la gestione del carico di lavoro dei cluster. È possibile prendere in considerazione il costo dell'utilizzo di un canale in diverse ore del giorno o il contenuto del messaggio quando si instradano i messaggi. Questi sono fattori che non vengono considerati dall'algoritmo di gestione del workload standard.

Nella maggior parte dei casi, l'algoritmo di gestione del carico di lavoro è sufficiente per le proprie esigenze. Tuttavia, in modo da poter fornire il proprio programma di uscita utente per personalizzare la gestione del carico di lavoro, WebSphere MQ include un'uscita utente, l'uscita del carico di lavoro del cluster.

Potresti avere alcune informazioni particolari sulla tua rete o sui messaggi che potresti utilizzare per influenzare il bilanciamento del carico di lavoro. Potresti sapere quali sono i canali ad alta capacità o gli instradamenti di rete economici oppure potresti voler instradare i messaggi in base al loro contenuto. È possibile decidere di scrivere un programma di uscita del carico di lavoro del cluster o utilizzarne uno fornito da una terza parte.

L'uscita del carico di lavoro del cluster viene richiamata quando si accede a una coda cluster. Viene richiamato da MQOPEN, MQPUT1 e MQPUT.

Il gestore code di destinazione selezionato all'ora MQOPEN è fisso se è specificato MQOO_BIND_ON_OPEN . In questo caso l'uscita viene eseguita solo una volta.

Se il gestore code di destinazione non è corretto all'ora MQOPEN , il gestore code di destinazione viene scelto al momento della chiamata MQPUT . Se il gestore code di destinazione non è disponibile o ha esito negativo mentre il messaggio è ancora sulla coda di trasmissione, l'uscita viene richiamata di nuovo. Viene selezionato un nuovo gestore code di destinazione. Se il canale dei messaggi ha esito negativo mentre il messaggio è in fase di trasferimento e il messaggio viene ripristinato, viene selezionato un nuovo gestore code di destinazione.

Su piattaforme diverse da z/OS, il gestore code caricherà la nuova uscita del carico di lavoro del cluster al successivo avvio del gestore code.

Se la definizione del gestore code non contiene un nome del programma di uscita del carico di lavoro del cluster, l'uscita del carico di lavoro del cluster non viene richiamata.

Vari dati vengono passati a un'exit del carico di lavoro del cluster nella struttura del parametro di uscita, MQWXP:

- La struttura di definizione dei messaggi, MQMD.
- Il parametro di lunghezza del messaggio.
- Una copia del messaggio o parte di esso.

Su piattaforme nonz/OS , se si utilizza CLWLMode=FAST, ogni processo del sistema operativo carica la propria copia dell'uscita. Connessioni differenti al gestore code possono causare il richiamo di copie differenti dell'exit. Se l'uscita viene eseguita in modalità sicura predefinita, CLWLMode=SAFE, una singola copia dell'uscita viene eseguita nel proprio processo separato.

Scrittura delle uscite del carico di lavoro del cluster

Per piattaforme diverse da z/OS, le uscite del carico di lavoro del cluster non devono utilizzare chiamate MQI. Per altri aspetti, le norme per la scrittura e la compilazione dei programmi di uscita del workload del cluster sono simili alle regole che si applicano ai programmi di uscita del canale. Seguire i passi riportati in “Scrittura e compilazione di uscite e servizi installabili” a pagina 376e utilizzare il programma di esempio “Uscita del carico di lavoro del cluster di esempio” a pagina 430 per scrivere e compilare l'uscita.

Per ulteriori informazioni sulle uscite dei canali, consultare “Scrittura di programmi di uscita canale” a pagina 401.

Configurazione delle uscite del carico di lavoro del cluster

È possibile denominare le uscite del carico di lavoro del cluster nella definizione del gestore code specificando l'attributo di uscita del carico di lavoro cluster nel comando `ALTER QMGR` . Ad esempio:

```
ALTER QMGR CLWLEXIT(myexit)
```

Uscita del carico di lavoro del cluster di esempio

WebSphere MQ include un esempio di programma di uscita del workload del cluster. È possibile copiare l'esempio e utilizzarlo come base per i propri programmi.

Su piattaforme diverse da z/OS

Il programma di uscita del carico di lavoro del cluster di esempio viene fornito in C e viene denominato `amqswlm0.c`. Può essere trovato in:

Tabella 57. Ubicazione del programma di uscita del carico di lavoro del cluster di esempio (non z/OS)

Piattaforma	**Percorso file**
AIX, HP-UX, Sun Solaris	*MQ_INSTALLATION_PATH*`/samp`
Windows	*MQ_INSTALLATION_PATH*`\Tools\c\Samples`

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Questa uscita di esempio instrada tutti i messaggi a un particolare gestore code, a meno che tale gestore code non diventi non disponibile. Reagisce all'errore del gestore code instradando i messaggi a un altro gestore code.

Indicare a quale gestore code si desidera inviare i messaggi. Specificare il nome del canale ricevente del cluster nell'attributo `CLWLDATA` nella definizione del gestore code. Ad esempio:

```
ALTER QMGR CLWLDATA('my-cluster-name.my-queue-manager')
```

Per abilitare l'exit, fornire il nome e il percorso completo nell'attributo `CLWLEXIT` :

Su sistemi UNIX and Linux :

```
ALTER QMGR CLWLEXIT('path/amqswlm(clwlFunction)')
```

Su Windows:

```
ALTER QMGR CLWLEXIT('path\amqswlm(clwlFunction)')
```

Ora, invece di utilizzare l'algoritmo di gestione del carico di lavoro fornito, WebSphere MQ richiama questa exit per instradare tutti i messaggi al gestore code scelto.

Creazione di un'applicazione IBM WebSphere MQ

Utilizzare queste informazioni per informazioni sulla generazione di un'applicazione IBM WebSphere MQ su piattaforme differenti.

Creazione della tua applicazione su AIX

Le pubblicazioni AIX descrivono come creare applicazioni eseguibili dai programmi scritti.

Questo argomento descrive le attività aggiuntive e le modifiche alle attività standard che è necessario eseguire quando si creano applicazioni WebSphere MQ per AIX da eseguire in AIX. Sono supportati C, C + + e COBOL. Per informazioni sulla preparazione dei programmi C++, consultare Utilizzo di C++.

Le attività che è necessario eseguire per creare un'applicazione eseguibile utilizzando WebSphere MQ per AIX variano in base al linguaggio di programmazione in cui è scritto il codice sorgente. Oltre a codificare le chiamate MQI nel tuo codice sorgente, devi aggiungere le istruzioni di lingua appropriate per includere i file di inclusione WebSphere MQ per AIX per la lingua che stai utilizzando. Acquisire familiarità con il contenuto di questi file. Consultare “File di definizione dati IBM WebSphere MQ” a pagina 81 per una descrizione completa.

Quando si eseguono applicazioni server con thread o client con thread, impostare la variabile di ambiente AIXTHREAD_SCOPE = S.

Preparazione dei programmi C in AIX

Questo argomento contiene informazioni sul collegamento delle librerie necessarie per preparare i programmi C su AIX.

I programmi C precompilati vengono forniti nella directory *MQ_INSTALLATION_PATH*/samp/bin . Utilizzare il compilatore ANSI ed eseguire i seguenti comandi. Per ulteriori informazioni sulla programmazione di applicazioni a 64 bit, consultare Coding standards on 64 - bit platforms.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Per applicazioni a 32 bit:

```
  $ xlc_r -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm
```

dove `amqsput0` è un programma di esempio.

Per le applicazioni a 64 bit:

```
  $ xlc_r -q64 -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64
-lmqm
```

dove `amqsput0` è un programma di esempio.

Se si sta utilizzando il compilatore VisualAge C/C++ per i programmi C + +, è necessario includere l'opzione `-q namemangling=v5` per ottenere tutti i simboli WebSphere MQ risolti durante il collegamento delle librerie.

Se si desidera utilizzare i programmi su una macchina su cui è installato solo il client WebSphere MQ per AIX , ricompilare i programmi per collegarli alla libreria del client (`-lmqic`).

Collegamento di librerie

Sono necessarie le seguenti librerie:

- Collegare i programmi con la libreria appropriata fornita da WebSphere WebSphere MQ.

  In un ambiente senza thread, collegarsi a una delle seguenti librerie:

File di libreria	Tipo di programma / uscita
libmqm.a	Server per C
libmqic.a & libmqm.a	Client per C

In un ambiente con thread, collegarsi a una delle seguenti librerie:

File di libreria	Tipo di programma / uscita
libmqm_r.a	Server per C
libmqic_r.a & libmqm_r.a	Client per C

Ad esempio, per creare un'applicazione WebSphere MQ con thread semplice da una sola unità di compilazione, eseguire questi comandi.

Per applicazioni a 32 bit:

```
$ xlc_r -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm_r
```

dove `amqsput0` è un programma di esempio.

Per le applicazioni a 64 bit:

```
$ xlc_r -q64 -o amqsputc_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64
-lmqm_r
```

dove `amqsput0` è un programma di esempio.

Se si desidera utilizzare i programmi su una macchina su cui è installato solo il client WebSphere MQ per AIX , ricompilare i programmi per collegarli alla libreria del client (`-lmqic`).

Nota:

1. Se si sta scrivendo un servizio installabile (per ulteriori informazioni, consultare Amministrazione), è necessario collegarsi alla libreria `libmqmzf.a` in un'applicazione senza thread e alla libreria `libmqmzf_r.a` in un'applicazione con thread.
2. Se si sta producendo un'applicazione per il coordinamento esterno da un gestore transazioni conforme a XA, come IBM TXSeries, Encinao BEA Tuxedo, è necessario collegarsi a `libmqmxa.a` (o `libmqmxa64.a` se il gestore transazioni considera il tipo 'long' come 64 bit) e alle librerie `libmqz.a` in un'applicazione non con thread e alle librerie `libmqmxa_r.a` (o `libmqmxa64_r.a`) e `libmqz_r.a` in un'applicazione con thread.
3. È necessario collegare le applicazioni attendibili alle librerie WebSphere MQ con thread. Tuttavia, solo un thread alla volta in un'applicazione attendibile su WebSphere MQ su sistemi UNIX and Linux può essere connesso.
4. È necessario collegare le librerie WebSphere MQ prima di qualsiasi altra libreria del prodotto.

Preparazione dei programmi COBOL in AIX

Utilizzare queste informazioni durante la preparazione dei programmi COBOL in AIX utilizzando IBM COBOL Set e Micro Focus COBOL.

MQ_INSTALLATION_PATH rappresenta la directory di livello superiore in cui è installato IBM WebSphere MQ .

- I copy book COBOL a 32 bit sono installati nella seguente directory:

  ```
  MQ_INSTALLATION_PATH/inc/cobcpy32
  ```

  e i collegamenti simbolici vengono creati in:

```
MQ_INSTALLATION_PATH/inc
```

- I copy book COBOL a 64 bit sono installati nella seguente directory:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

Nei seguenti esempi impostare la variabile di ambiente **COBCPY** su:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

per applicazioni a 32 bit e:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

per applicazioni a 64 bit.

È necessario collegare il programma a uno dei seguenti file di libreria:

File di libreria	Tipo di programma / uscita
libmqmcb.a	Server per COBOL (applicazione senza thread)
libmqmcb_r.a	Server per COBOL (applicazione con thread)
libmqicb.a	Client per COBOL (applicazione senza thread)
libmqicb_r.a	Client per COBOL (applicazione con thread)

È possibile utilizzare il compilatore IBM COBOL Set o Micro Focus COBOL a seconda del programma:

- I programmi che iniziano con `amqm` sono adatti per il compilatore Micro Focus COBOL e
- I programmi che iniziano con `amq0` sono adatti per il compilatore.

Preparazione dei programmi COBOL utilizzando IBM COBOL Set for AIX

I programmi COBOL di esempio vengono forniti con IBM WebSphere MQ. Per compilare tale programma, immettere il comando appropriato dal seguente elenco:

Applicazione server senza thread a 32 bit

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb -qLIB \
-I<COBCPY>
```

Applicazione client senza thread a 32 bit

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb -qLIB \
-I<COBCPY>
```

Applicazione server con thread a 32 bit

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -I<COBCPY>
```

Applicazione client con thread a 32 bit

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -I<COBCPY>
```

Applicazione server senza thread a 64 bit

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqmcb \
-qLIB -I<COBCPY>
```

Applicazione client senza thread a 64 bit

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqicb \
-qLIB -I<COBCPY>
```

Applicazione server con thread a 64 bit

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -I<COBCPY>
```

Applicazione client con thread a 64 bit

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -I<COBCPY>
```

Preparazione dei programmi COBOL utilizzando Micro Focus COBOL

Impostare le variabili di ambiente prima di compilare il proprio programma come segue:

```
export COBCPY=<COBCPY>
export LIBPATH=MQ_INSTALLATION_PATH/lib:$LIBPATH
```

Per compilare un programma COBOL a 32 bit utilizzando Micro Focus COBOL, immettere:

- Server per COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb
```

- Client per COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb
```

- Server con thread per COBOL

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r
```

- Client con thread per COBOL

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r
```

Per compilare un programma COBOL a 64 bit utilizzando Micro Focus COBOL, immettere:

- Server per COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb
```

- Client per COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb
```

- Server con thread per COBOL

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r
```

- Client con thread per COBOL

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r
```

dove amqminqx è un programma di esempio

Consultare la documentazione Micro Focus COBOL per una descrizione delle variabili di ambiente che è necessario impostare.

Preparazione dei programmi applicativi CICS in AIX

Utilizzare queste informazioni durante la preparazione dei programmi CICS in AIX.

I moduli switch XA vengono forniti per consentire il collegamento di CICS con IBM WebSphere MQ:

Tabella 58. Codice essenziale per i programmi di applicazione di CICS su AIX: routine di inizializzazione XA

Descrizione	**C (origine)**	**C (exec) - aggiungi al tuo XAD.Stanza**
Routine di inizializzazione XA	amqzscix.c	*amqzsc* - CICS per AIX

Utilizzare la versione preintegrata del IBM WebSphere MQ file di caricamento switch *amqzsc* , fornito con il prodotto.

Collegare sempre le transazioni C con la libreria IBM WebSphere MQ threadsafe *libmqm_r.a.*, e le transazioni COBOL con la libreria COBOL *libmqmcb_r.a*.

È possibile trovare ulteriori informazioni sul supporto delle transazioni CICS in Amministrazione.

Supporto TXSeries CICS

IBM WebSphere MQ su AIX supporta TXSeries CICS utilizzando l'interfaccia XA. Assicurarsi che le applicazioni CICS siano collegate alla versione con thread delle librerie IBM WebSphere MQ .

È possibile eseguire i programmi CICS utilizzando IBM COBOL Set for AIX o Micro Focus COBOL. Le seguenti sezioni descrivono la differenza tra l'esecuzione dei programmi CICS su IBM COBOL Set for AIX e Micro Focus COBOL.

Scrivere i programmi WebSphere MQ caricati nella stessa regione CICS in C o COBOL. Non è possibile effettuare una combinazione di chiamate C e COBOL MQI nella stessa regione CICS . La maggior parte delle chiamate MQI nella seconda lingua utilizzata ha esito negativo con un codice motivo MQRC_HOBJ_ERROR.

Preparazione dei programmi CICS COBOL utilizzando IBM COBOL Set for AIX

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato IBM WebSphere MQ .

Per utilizzare IBM COBOL, attenersi alla seguente procedura:

1. Esportare la seguente variabile di ambiente:

```
export LDFLAGS="-qLIB -bI:/usr/lpp/cics/lib/cicsprIBMCOB.exp  \
                -IMQ_INSTALLATION_PATH/inc -I/usr/lpp/cics/include  \
                -e _iwz_cobol_main  \
```

   dove LIB è una direttiva del compilatore.
2. Convertire, compilare e collegare il programma immettendo:

```
cicstcl -l IBMCOB <yourprog>.ccp
```

Preparazione di programmi COBOL CICS utilizzando Micro Focus COBOL

MQ_INSTALLATION_PATH rappresenta la directory di livello superiore in cui è installato IBM WebSphere MQ .

Per utilizzare Micro Focus COBOL, attenersi alla seguente procedura:

1. Aggiungere il modulo della libreria di runtime IBM WebSphere MQ COBOL alla libreria di runtime utilizzando il seguente comando:

```
cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
            MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
```

   Nota: Con `cicsmkcobol`, IBM WebSphere MQ non consente di effettuare chiamate MQI nel linguaggio di programmazione C dall'applicazione COBOL.

   Se le applicazioni esistenti dispongono di tali chiamate, si consiglia di spostare queste funzioni dalle applicazioni COBOL alla propria libreria, ad esempio, `myMQ.so`. Dopo lo spostamento delle funzioni, non includere la IBM WebSphere MQ libreria `libmqmcbrt.o` quando si crea l'applicazione COBOL per CICS.

   Inoltre, se l'applicazione COBOL non effettua alcuna chiamata COBOL MQI, non collegare `libmqmz_r` con `cicsmkcobol`.

   Ciò crea il file del metodo del linguaggio Micro Focus COBOL e abilita la libreria COBOL di runtime CICS a richiamare IBM WebSphere MQ su sistemi UNIX and Linux .

   Nota: Eseguire `cicsmkcobol` solo quando si installa uno dei seguenti prodotti:

   - Nuova versione o release di Micro Focus COBOL
   - Nuova versione o release di CICS per AIX
   - Nuova versione o release di qualsiasi prodotto database supportato (solo per transazioni COBOL)
   - Nuova versione o release di IBM WebSphere MQ
2. Esportare la seguente variabile di ambiente:

```
COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
```

3. Convertire, compilare e collegare il programma immettendo:

```
cicstcl -l COBOL -e <yourprog>.ccp
```

Preparazione dei programmi C CICS

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato IBM WebSphere MQ .

Creare programmi CICS C utilizzando le funzionalità standard di CICS :

1. Esportare ***una*** delle seguenti variabili di ambiente:
   - LDFLAGS= "-L/*MQ_INSTALLATION_PATH*lib -lmqm_r" export LDFLAGS
   - USERLIB = "-L*MQ_INSTALLATION_PATH*lib -lmqm_r" export USERLIB
2. Convertire, compilare e collegare il programma immettendo:

```
cicstcl -l C amqscic0.ccs
```

Transazione di esempio CICS C

L'origine C di esempio per una transazione AIX IBM WebSphere MQ è fornita da AMQSCIC0.CCS. La transazione legge i messaggi dalla coda di trasmissione SYSTEM.SAMPLE.CICS.WORKQUEUE sul gestore code predefinito e le inserisce nella coda locale con un nome coda contenuto nell'intestazione di trasmissione del messaggio. Eventuali errori vengono inviati alla coda SYSTEM.SAMPLE.CICS.DLQ. Utilizzare lo script MQSC di esempio AMQSCIC0.TST per creare queste code e le code di input di esempio.

Creazione della tua applicazione su HP Integrity NonStop Server

Queste informazioni descrivono le attività aggiuntive e le modifiche alle attività standard che è necessario eseguire quando si crea un client IBM WebSphere MQ per le applicazioni HP Integrity NonStop Server da eseguire in HP Integrity NonStop Server.

Sono supportati C, COBOL e pTAL .

Intestazioni OSS e Guardian e biblioteche pubbliche

Fornisce elenchi di intestazioni OSS e Guardian e librerie pubbliche. Sono elencate le intestazioni OSS, l'eseguibile pubblico OSS e le librerie di importazione pubbliche, le intestazioni Guardian e le librerie di importazione pubbliche ed eseguibili pubbliche Guardian.

“Intestazioni OSS” a pagina 437

“Librerie pubbliche di importazione ed eseguibili OSS” a pagina 438

“Intestazioni del guardiano” a pagina 439

“Librerie di importazione pubbliche ed eseguibili pubbliche Guardian” a pagina 439

Intestazioni OSS

Tabella 59. Intestazioni OSS

Oggetto	Ubicazione	Descrizione
cmqbc.h	<mqinstall>/inc	IBM WebSphere MQ OSS (C - language header)
cmqc.h	<mqinstall>/inc	IBM WebSphere MQ OSS (C - language header)
cmqfc.h	<mqinstall>/inc	IBM WebSphere MQ OSS (C - language header)
cmqec.h	<mqinstall>/inc	IBM WebSphere MQ OSS (C - language header)

Tabella 59. Intestazioni OSS (Continua)		
Oggetto	**Ubicazione**	**Descrizione**
cmqpsc.h	<mqinstall>/inc	IBM WebSphere MQ OSS (C - language header)
cmqxc.h	<mqinstall>/inc	IBM WebSphere MQ OSS (C - language header)
cmqzc.h	<mqinstall>/inc	IBM WebSphere MQ OSS (C - language header)
cmqcobol.cpy	<mqinstall>/inc	IBM WebSphere MQ Copybook COBOL (OSS)
cmqbt.tal	<mqinstall>/inc	Intestazione IBM WebSphere MQ pTAL (OSS)
cmqcft.tal	<mqinstall>/inc	Intestazione IBM WebSphere MQ pTAL (OSS)
cmqpst.tal	<mqinstall>/inc	Intestazione IBM WebSphere MQ pTAL (OSS)
cmqt.tal	<mqinstall>/inc	Intestazione IBM WebSphere MQ pTAL (OSS)
cmqxt.tal	<mqinstall>/inc	Intestazione IBM WebSphere MQ pTAL (OSS)

Librerie pubbliche di importazione ed eseguibili OSS

Tabella 60. Librerie pubbliche di importazione ed eseguibili OSS		
Oggetto	**Ubicazione**	**Descrizione**
libmqic.so	<mqinstall>/bin	Libreria eseguibile pubblica IBM WebSphere MQ (senza thread OSS)
libmqic_r.so	<mqinstall>/bin	Libreria eseguibile pubblica IBM WebSphere MQ (multithread OSS)
libmqic.so	<mqinstall>/lib	Libreria di importazione pubblica IBM WebSphere MQ (senza thread OSS)
libmqic_r.so	<mqinstall>/lib	Libreria di importazione pubblica IBM WebSphere MQ (multithread OSS)
mqicb	<mqinstall>/lib	Libreria di importazione pubblica IBM WebSphere MQ per COBOL (OSS)

Intestazioni del guardiano

Tabella 61. Intestazioni del guardiano

Oggetto	Ubicazione	Descrizione
cmqbch	<mqinstall>/inc/G	IBM WebSphere MQ Intestazione in linguaggio C (Guardian)
cmqch	<mqinstall>/inc/G	IBM WebSphere MQ Intestazione in linguaggio C (Guardian)
cmqfch	<mqinstall>/inc/G	IBM WebSphere MQ Intestazione in linguaggio C (Guardian)
cmqech	<mqinstall>/inc/G	IBM WebSphere MQ Intestazione in linguaggio C (Guardian)
cmqpsch	<mqinstall>/inc/G	IBM WebSphere MQ Intestazione in linguaggio C (Guardian)
cmqxch	<mqinstall>/inc/G	IBM WebSphere MQ Intestazione in linguaggio C (Guardian)
cmqzch	<mqinstall>/inc/G	IBM WebSphere MQ Intestazione in linguaggio C (Guardian)
cmqcobol	<mqinstall>/inc/G	Copybook IBM WebSphere MQ COBOL (Guardian)
cmqbt	<mqinstall>/inc/G	Intestazione IBM WebSphere MQ pTAL (Guardian)
cmqcft	<mqinstall>/inc/G	Intestazione IBM WebSphere MQ pTAL (Guardian)
cmqpst	<mqinstall>/inc/G	Intestazione IBM WebSphere MQ pTAL (Guardian)
cmqt	<mqinstall>/inc/G	Intestazione IBM WebSphere MQ pTAL (Guardian)
cmqxt	<mqinstall>/inc/G	Intestazione IBM WebSphere MQ pTAL (Guardian)

Librerie di importazione pubbliche ed eseguibili pubbliche Guardian

Tabella 62. Librerie di importazione pubbliche ed eseguibili pubbliche Guardian

Oggetto	Ubicazione	Descrizione
mqic	<mqinstall>/bin/G	Libreria eseguibile pubblica IBM WebSphere MQ (Guardian)
mqicb	<mqinstall>/lib/G	Libreria di importazione pubblica IBM WebSphere MQ per COBOL (Guardian)

Preparazione dei programmi C in HP Integrity NonStop Server

Questo argomento contiene informazioni da considerare quando si preparano i programmi C in HP Integrity NonStop Server insieme ad esempi dei comandi che si utilizzano quando si creano le applicazioni quando si utilizza il compilatore OSS C e quando si utilizza il compilatore Guardian C.

I programmi C precompilati vengono forniti nella directory MQ_INSTALLATION_PATH/opt/mqm/samp/bin . Per creare un esempio dal codice sorgente, utilizzare il compilatore c89 .

È necessario collegare i programmi con la libreria appropriata fornita da IBM WebSphere MQ. La seguente tabella elenca le librerie a cui è necessario collegarsi quando si stanno preparando i programmi C su HP Integrity NonStop Server.

Tabella 63. . Librerie di collegamento HP Integrity NonStop Server

Libreria	**Descrizione**
libmqic.so	OSS non sottoposto a thread
libmqic_r.so	OSS multi - thread
mqic	Unità di controllo

Le applicazioni IBM WebSphere MQ native a più thread devono utilizzare la funzione PUT (Posix User Threads). Non esiste alcun supporto per SPT (Standard Posix Threads) in questo prodotto.

Creazione di applicazioni utilizzando il compilatore OSS C

Questa sezione contiene esempi dei comandi utilizzati per creare programmi destinati a OSS o Guardian quando si utilizza il compilatore OSS.

MQ_INSTALLATION_PATH rappresenta la directory di livello superiore in cui è installato IBM WebSphere MQ .

Il seguente esempio crea un'applicazione OSS client C non sottoposta a thread:

```
c89 -Wsystype=oss -o amqsputc amqsput0.c -IMQ_INSTALLATION_PATH/opt/mqm/inc
-LMQ_INSTALLATION_PATH/opt/mqm/lib -lmqic
```

Il seguente esempio crea un'applicazione OSS client C a più thread:

```
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
```

Il seguente esempio crea un'applicazione client 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
```

Creazione di applicazioni utilizzando il compilatore Guardian C

Questa sezione contiene esempi di comandi utilizzati per creare programmi destinati a Guardian quando si utilizza il compilatore Guardian.

MQ_INSTALLATION_PATH rappresenta il volume e il volume secondario Guardian in cui è installato IBM WebSphere MQ .

Il seguente esempio crea un'applicazione client Guardian C:

```
CCOMP /in AMQSPUT0/ AMQSPUTC;&
       runnable,systype guardian,nolist,&
       ssv0 "$system.system",&
       ssv1 "MQINSTALLATION_SUBVOL",&
       ld(-LMQINSTALLATION_SUBVOL -lmqic)
```

Preparazione dei programmi COBOL

Questo argomento contiene informazioni da considerare quando si preparano programmi C per il client IBM WebSphere MQ per HP Integrity NonStop Server. Contiene esempi dei comandi che si utilizzano quando si creano applicazioni quando si utilizza il compilatore OSS ECOBOL e quando si utilizza il compilatore Guardian ECOBOL.

Per creare un esempio COBOL dal codice sorgente, utilizzare il compilatore ECOBOL.

La seguente tabella elenca le librerie necessarie durante la preparazione dei programmi COBOL su HP Integrity NonStop Server. È necessario collegare i programmi con la libreria appropriata fornita da IBM WebSphere MQ.

Tabella 64. . Librerie di collegamento HP Integrity NonStop Server	
Libreria	**Descrizione**
libmqic.so	OSS non sottoposto a thread
mqic	Unità di controllo

Quando si esegue un'applicazione COBOL che si connette a un gestore code, è innanzitutto necessario impostare la variabile *SAVE - ENVIRONMENT* su ON. Per impostare la variabile *SAVE - ENVIRONMENT* su ON:

- Per OSS, immettere il comando seguente:

```
export SAVE-ENVIRONMENT=ON
```

- Per Guardian, immettere il seguente comando:

```
param SAVE-ENVIRONMENT ON
```

Se non si imposta la variabile *SAVE - ENVIRONMENT* su ON, quando l'applicazione tenta di connettersi a un gestore code, l'operazione ha esito negativo con il codice motivo 2058 (080A) (RC2058): MQRC_Q_MGR_NAME_ERROR.

Creazione di applicazioni utilizzando il compilatore OSS ECOBOL

Questa sezione contiene esempi dei comandi utilizzati per creare programmi destinati a OSS o Guardian quando si utilizza il compilatore OSS ECOBOL.

MQ_INSTALLATION_PATH rappresenta la directory di livello superiore in cui è installato IBM WebSphere MQ .

Il seguente esempio crea un'applicazione OSS client 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
```

Il seguente esempio crea un'applicazione Guardian del client 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
```

Creazione di applicazioni utilizzando il compilatore Guardian ECOBOL

Questa sezione contiene esempi di comandi utilizzati per creare programmi destinati a Guardian quando si utilizza il compilatore ECOBOL Guardian.

MQ_INSTALLATION_SUBVOL rappresenta il volume e il volume secondario Guardian in cui è installato IBM WebSphere MQ .

Il seguente esempio crea un'applicazione Guardian del client COBOL:

```
ECOBOL /in MQSPUTL/ MQSPUT,MQINSTALLATION_SUBVOL.cmqcobol;
       call-shared;ansi;port;save all;nolist;runnable;
       consult MQINSTALLATION_SUBVOL.mqicb;
       eld(-LMQINSTALLATION_SUBVOL -lmqic)
```

Preparazione dei programmi pTAL

Imparare a creare programmi pTAL per il client IBM WebSphere MQ sulla piattaforma HP Integrity NonStop Server .

Per creare un esempio pTAL dal codice sorgente, utilizzare il compilatore EPTAL.

Nota:

- Le applicazioni pTAL IBM WebSphere MQ devono utilizzare una routine principale scritta in linguaggio C o COBOL.
- Le applicazioni pTAL possono essere create solo in Guardian.

La seguente tabella elenca la libreria necessaria quando si preparano programmi pTAL su HP Integrity NonStop Server. È necessario collegare i programmi con la libreria appropriata fornita da IBM WebSphere MQ.

Tabella 65. . Libreria di link HP Integrity NonStop Server	
Libreria	**Descrizione**
mqic	Unità di controllo

Creazione di applicazioni utilizzando il compilatore Guardian EPTAL

Questa sezione contiene esempi di comandi utilizzati per creare programmi destinati a Guardian quando si utilizza il compilatore Guardian EPTAL.

MQINSTALLATION_SUBVOL rappresenta il volume e il volume secondario Guardian in cui è installato IBM WebSphere MQ .

Le applicazioni pTAL IBM WebSphere MQ devono utilizzare una routine principale scritta in linguaggio C o COBOL.

Il seguente esempio crea un'applicazione Guardian client 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)
```

Creazione dell'applicazione su HP-UX

Queste informazioni descrivono le attività aggiuntive e le modifiche alle attività standard da eseguire quando si creano applicazioni WebSphere MQ per le applicazioni HP-UX da eseguire in HP-UX.

Sono supportati C, C + + e COBOL. Per informazioni sulla preparazione dei programmi C++, consultare Utilizzo di C++.

Le attività da eseguire per creare un'applicazione eseguibile utilizzando WebSphere MQ per HP-UX variano in base al linguaggio di programmazione in cui è scritto il codice sorgente. Oltre a codificare le chiamate MQI nel codice sorgente, è necessario aggiungere le istruzioni di lingua appropriate per includere i file di inclusione WebSphere MQ per HP-UX per la lingua che si sta utilizzando. Acquisire

familiarità con il contenuto di questi file. Consultare “File di definizione dati IBM WebSphere MQ” a pagina 81 per una descrizione completa.

In questo argomento, viene utilizzato il carattere barra retroversa (\) per suddividere i comandi lunghi su più di una riga. Non immettere questo carattere; immettere ogni comando come riga singola.

Preparazione dei programmi C in HP-UX

Questo argomento contiene informazioni da considerare durante la preparazione dei programmi C in HP-UX; con esempi per la piattaforma IA64 (IPF).

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Lavorare nell'ambiente normale. I programmi C precompilati vengono forniti nella directory *MQ_INSTALLATION_PATH*/samp/bin .

Per ulteriori informazioni sulla programmazione di applicazioni a 64 bit, consultare Standard di codifica su piattaforme a 64 bit.

Per utilizzare SSL, i client WebSphere MQ MQI su HP-UX devono essere creati utilizzando i thread POSIX .

Alcuni esempi da considerare sono:

- “Piattaforma IA64 (IPF)” a pagina 443
- “Collegamento di librerie” a pagina 445

Piattaforma IA64 (IPF)

Crea esempi di amqsput0, cliexit e srvexit sulla piattaforma IA64(IPF).

Il seguente esempio crea il programma di esempio amqsput0 come un'applicazione client in un ambiente a 32 bit non con thread:

```
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
```

Il seguente esempio crea il programma di esempio amqsput0 come applicazione client in un ambiente a 32 bit con thread:

```
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
```

Il seguente esempio crea il programma di esempio amqsput0 come applicazione client in un ambiente a 64 bit senza thread:

```
c89 +DD64 +e -D_HPUX_SOURCE -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic
```

Il seguente esempio crea il programma di esempio amqsput0 come applicazione client in un ambiente a 64 bit con thread:

```
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
```

Il seguente esempio crea il programma di esempio amqsput0 come applicazione server in un ambiente a 32 bit senza thread:

```
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
```

Il seguente esempio crea il programma di esempio, amqsput0 , come applicazione server in un ambiente a 32 bit con thread:

```
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
```

Il seguente esempio crea il programma di esempio amqsput0 come applicazione server in un ambiente a 64 bit non thread:

```
c89 +DD64 +e -D_HPUX_SOURCE -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm
```

Il seguente esempio crea il programma di esempio amqsput0 come applicazione server in un ambiente a 64 bit con thread:

```
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
```

Il seguente esempio crea un'uscita client cliexit in un ambiente a 32 bit non thread:

```
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
```

Il seguente esempio crea un'uscita client cliexit in un ambiente a 32 bit con thread:

```
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
```

Il seguente esempio crea un'uscita client cliexit in un ambiente a 64 bit non thread:

```
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
```

Il seguente esempio crea un'uscita client cliexit in un ambiente a 64 bit con thread:

```
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
```

Il seguente esempio crea un'uscita server srvexit in un ambiente a 32 bit non thread:

```
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
```

Il seguente esempio crea un'uscita server srvexit in un ambiente a 32 bit con thread:

```
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
```

Il seguente esempio crea un'uscita server srvexit in un ambiente a 64 bit non thread:

```
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
```

Il seguente esempio crea un'uscita server srvexit in un ambiente a 64 bit con thread:

```
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
```

Collegamento di librerie

È necessario collegare i programmi con la libreria appropriata fornita da WebSphere MQ.

La seguente tabella mostra quale libreria utilizzare in ambienti diversi

Piattaforma hardware	Ambiente con o senza thread	Tipo di programma / uscita	File di libreria
IA64 (IPF)	Threaded	Server & client per C	libmqm_r.so
IA64 (IPF)	Threaded	Client per C	libmqic_r.so
IA64 (IPF)	Senza thread	Server & client per C	libmqm.so
IA64 (IPF)	Senza thread	Client per C	libmqic.so

Nota:

1. Se si sta scrivendo un servizio installabile (per ulteriori informazioni, consultare Amministrazione), è necessario collegarsi alla libreria `libmqmzf.sl` .
2. Se si sta producendo un'applicazione per il coordinamento esterno da parte di un gestore transazioni compatibile con XA, come IBM TXSeries Encinao BEA Tuxedo, è necessario collegarsi a `libmqmxa.sl` (o `libmqmxa64.sl` se il gestore transazioni considera il tipo 'long' come 64 bit) e alle librerie `libmqz.sl` in un'applicazione non con thread e alle librerie `libmqmxa_r.sl` (o `libmqmxa64_r.sl`) e `libmqz_r.sl` in un'applicazione con thread.
3. È necessario collegare le librerie WebSphere MQ prima di qualsiasi altra libreria del prodotto.

Preparazione di programmi COBOL in HP-UX

Informazioni sulla preparazione dei programmi COBOL in HP-UX, sull'utilizzo di Micro Focus Server Express con WebSphere MQ sulla piattaforma IA64 (IPF) e sull'esecuzione di programmi nell'ambiente client WebSphere MQ MQI.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Note per gli utenti

1. I copy book COBOL a 32 bit sono installati nella seguente directory:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   e i collegamenti simbolici vengono creati in:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. I copy book COBOL a 64 bit sono installati nella seguente directory:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. Nei seguenti esempi impostare COBCPY su:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

per applicazioni a 32 bit e:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

per applicazioni a 64 bit.

Compilare i programmi utilizzando il compilatore Micro Focus. I file di copia che dichiarano le strutture sono in *MQ_INSTALLATION_PATH*/inc:

```
$ export LIB=MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="<COBCPY>"
```

Compilazione di programmi a 32 bit:

```
$ 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
```

Compilazione di programmi a 64 bit:

```
$ 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
```

dove `amqsput` è un programma di esempio

Assicurarsi di aver specificato dimensioni di stack di runtime adeguate; 16 KB è il minimo consigliato.

È necessario collegare i programmi con la libreria appropriata fornita da WebSphere MQ. La seguente tabella mostra quale libreria utilizzare in ambienti diversi

Piattaforma hardware	Tipo di programma / uscita	File di libreria
IA64 (IPF)	Server per COBOL	libmqmcb.so
IA64 (IPF)	Client per COBOL	libmqicb.so
IA64 (IPF)	Applicazioni con thread	libmqmcb_r.so

Utilizzo di Micro Focus Server Express con WebSphere MQ sulla piattaforma IA64 (IPF)

Consultare “Modelli di spazio di indirizzo supportati da WebSphere MQ per HP-UX on IA64 (IPF)” a pagina 448 per i dettagli sull'utilizzo di Micro Focus Server Express in combinazione con WebSphere MQ sulla piattaforma HP/IPF.

Programmi da eseguire nell'ambiente client MQI WebSphere MQ

Se si utilizza LU 6.2 per connettere il client MQI a un server, collegare l'applicazione a libsna.a, parte del prodotto SNAplusAPI . Utilizzare le opzioni `-lV3` e `-lstr` nel comando di compilazione e collegamento.

- L'opzione `-lV3` fornisce l'accesso del programma alla libreria di segnalazione AT & T (SNAplusAPI usa segnali AT & T)
- L'opzione `-lstr` collega il programma al componente Streams

Preparazione dei programmi di CICS in HP-UX

Imparare a creare programmi di transazione CICS in HP-UX.

Per creare la transazione CICS di esempio, amqscic0.ccs, eseguire il seguente comando:

```
$ export USERLIB="-lmqm_r"
$ cicstcl -l C amqscic0.ccs
```

Viene fornito un modulo switch XA per consentire il collegamento di CICS con WebSphere MQ:

Tabella 66. Codice essenziale per applicazioni CICS (HP-UX)

Descrizione	**C (origine)**	**C (exec)**
Routine di inizializzazione XA	amqzscix.c	amqzsc

È possibile trovare ulteriori informazioni sul supporto delle transazioni CICS in Amministrazione.

Supporto TXSeries CICS

WebSphere MQ su HP-UX supporta TXSeries CICS utilizzando l'interfaccia XA. Accertarsi che le applicazioni CICS siano collegate alla versione con thread delle librerie MQ .

Scrivere i programmi WebSphere MQ caricati nella stessa regione CICS in C o COBOL. Non è possibile effettuare una combinazione di chiamate C e COBOL MQI nella stessa regione CICS . La maggior parte delle chiamate MQI nella seconda lingua utilizzata ha esito negativo con un codice motivo `MQRC_HOBJ_ERROR`.

Transazione di esempio CICS C

L'origine C di esempio per una transazione CICS WebSphere MQ è fornita da AMQSCIC0.CCS. La transazione legge i messaggi dalla coda di trasmissione SYSTEM.SAMPLE.CICS.WORKQUEUE sul gestore code predefinito e le inserisce nella coda locale con il nome della coda contenuto nell'intestazione di trasmissione del messaggio. Eventuali errori vengono inviati alla coda SYSTEM.SAMPLE.CICS.DLQ. Utilizzare lo script MQSC di esempio AMQSCIC0.TST per creare queste code e le code di input di esempio.

Preparazione dei programmi COBOL CICS utilizzando Micro Focus COBOL

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Per utilizzare Micro Focus COBOL, attenersi alla seguente procedura:

1. Aggiungere WebSphere MQ modulo della libreria di runtime COBOL alla libreria di runtime utilizzando il seguente comando:

   ```
   cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
               MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
   ```

   Nota: Con `cicsmkcobol`, WebSphere MQ non consente di effettuare chiamate MQI nel linguaggio di programmazione C dall'applicazione COBOL.

   Se le applicazioni esistenti dispongono di tali chiamate, si consiglia di spostare queste funzioni dalle applicazioni COBOL alla propria libreria, ad esempio, `myMQ.so`. Dopo lo spostamento, queste funzioni non includono la libreria WebSphere MQ `libmqmcbrt.o` durante la creazione dell'applicazione COBOL per CICS.

   Inoltre, se l'applicazione COBOL non effettua alcuna chiamata COBOL MQI, non collegare `libmqmz_r` con `cicsmkcobol`.

   Ciò crea il file del metodo del linguaggio Micro Focus COBOL e consente alla libreria COBOL di runtime CICS di chiamare WebSphere MQ su sistemi UNIX and Linux .

Nota: Eseguire `cicsmkcobol` solo quando si installa uno dei seguenti prodotti:

- Nuova versione o release di Micro Focus COBOL
- Nuova versione o release di CICS per HP-UX
- Nuova versione o release di qualsiasi prodotto database supportato (solo per transazioni COBOL)
- Nuova versione o release di WebSphere MQ

2. Esportare la seguente variabile di ambiente:

```
COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
```

3. Convertire, compilare e collegare il programma immettendo:

```
cicstcl -l COBOL -e <yourprog>.ccp
```

Modelli di spazio di indirizzo supportati da WebSphere MQ per HP-UX on IA64 (IPF)

HP-UX fornisce diversi modelli di spazio di indirizzo che possono essere sfruttati dalle applicazioni WebSphere MQ .

HP-UX supporta due modelli di spazio di indirizzo:

- MGAS - Spazio di indirizzo globale (questo è il valore predefinito e viene utilizzato da WebSphere MQ)
- MPAS - Spazio di indirizzo privato

Le applicazioni che si connettono a WebSphere MQ possono utilizzare i modelli di spazio di indirizzi MGAS o MPAS. Le applicazioni create utilizzando il modello MPAS che si collegano a WebSphere MQ utilizzando la memoria condivisa potrebbero comportare un costo delle prestazioni minore a causa dell'inefficienza nell'associazione delle pagine di memoria condivisa utilizzate da WebSphere MQ nello spazio di indirizzo virtuale del programma MPAS.

Le applicazioni COBOL create utilizzando Micro Focus Server Express utilizzano il modello MPAS per impostazione predefinita.

È possibile utilizzare il programma **chatr** per controllare e modificare il modello di indirizzamento utilizzato da un programma.

Se si verificano problemi durante la connessione a WebSphere MQ da programmi MPAS a 32 bit, considerare l'utilizzo del modello di indirizzamento MGAS o la creazione dell'applicazione come applicazione MPAS a 64 bit piuttosto che come applicazione MPAS a 32 bit.

Ulteriori dettagli sui modelli di spazio di indirizzi MGAS e MPAS sono disponibili nella documentazione HP-UX .

Creazione della tua applicazione su Linux

Queste informazioni descrivono le attività aggiuntive e le modifiche alle attività standard da eseguire durante la creazione di WebSphere MQ per l'esecuzione delle applicazioni Linux .

C e C++ sono supportati. Per informazioni sulla preparazione dei programmi C++, consultare Utilizzo di C++.

Preparazione dei programmi C in Linux

I programmi C precompilati vengono forniti nella directory *MQ_INSTALLATION_PATH*/samp/bin . Per creare un esempio dal codice sorgente, utilizzare il compilatore `gcc` .

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Lavorare nell'ambiente normale. Per ulteriori informazioni sulla programmazione di applicazioni a 64 bit, consultare Coding standards on 64 - bit platforms.

Collegamento di librerie

La seguente tabella elenca le librerie necessarie durante la preparazione dei programmi C su Linux.

- È necessario collegare i programmi con la libreria appropriata fornita da WebSphere MQ.

  In un ambiente senza thread, collegarsi a una delle seguenti librerie:

File di libreria	Tipo di programma / uscita
libmqm.so	Server per C
libmqic.so & libmqm.so	Client per C

  In un ambiente con thread, collegarsi a una delle seguenti librerie:

File di libreria	Tipo di programma / uscita
libmqm_r.so	Server per C
libmqic_r.so & libmqm_r.so	Client per C

  Nota:

  1. Se si sta scrivendo un servizio installabile (per ulteriori informazioni, consultare Amministrazione), è necessario collegarsi alla libreria `libmqmzf.so` .
  2. Se si sta producendo un'applicazione per il coordinamento esterno da parte di un gestore transazioni compatibile con XA, come IBM TXSeries Encinao BEA Tuxedo, è necessario collegarsi a `libmqmxa.so` (o `libmqmxa64.so` se il gestore transazioni considera il tipo 'long' come 64 bit) e alle librerie `libmqz.so` in un'applicazione non con thread e alle librerie `libmqmxa_r.so` (o `libmqmxa64_r.so`) e `libmqz_r.so` in un'applicazione con thread.
  3. È necessario collegare le librerie WebSphere MQ prima di qualsiasi altra libreria del prodotto.

Creazione di applicazioni a 31 bit

Questo argomento contiene esempi di comandi utilizzati per creare programmi a 31 bit in vari ambienti.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Applicazione client C, a 31 bit, senza thread

```
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
```

Applicazione client C, a 31 bit, con thread

```
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
```

Applicazione server C, a 31 bit, senza thread

```
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
```

Applicazione server C, a 31 bit, con thread

```
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
```

Applicazione client C++, a 31 bit, senza thread

```
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
```

Applicazione client C++, a 31 bit, con thread

```
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
```

Applicazione server C++, a 31 bit, senza thread

```
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
```

Applicazione server C++, a 31 bit, con thread

```
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
```

Uscita client C, a 31 bit, senza thread

```
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
```

Uscita client C, a 31 bit, con thread

```
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
```

Uscita server C, a 31 bit, senza thread

```
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
```

Uscita server C, a 31 bit, con thread

```
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
```

Creazione di applicazioni a 32 bit

Questo argomento contiene esempi dei comandi utilizzati per creare programmi a 32 - bit in vari ambienti.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Applicazione client C, a 32 bit, senza thread

```
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
```

Applicazione client C, a 32 bit, con thread

```
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
```

Applicazione server C, a 32 bit, senza thread

```
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
```

Applicazione server C, 32 bit, con thread

```
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
```

Applicazione client C + +, a 32 bit, senza thread

```
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
```

Applicazione client C++, a 32 bit, con thread

```
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
```

Applicazione server C + +, 32 bit, senza thread

```
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
```

Applicazione server C + +, a 32 bit, con thread

```
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
```

Uscita client C, a 32 bit, senza thread

```
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
```

Uscita client C, 32 bit, con thread

```
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
```

Uscita server C, 32 bit, senza thread

```
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
```

Uscita server C, a 32 bit, con thread

```
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
```

Creazione di applicazioni a 64 bit

Questo argomento contiene esempi di comandi utilizzati per creare programmi a 64 bit in vari ambienti.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Applicazione client C, a 64 bit, senza thread

```
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
```

Applicazione client C, a 64 bit, con thread

```
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
```

Applicazione server C, 64 bit, senza thread

```
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
```

Applicazione server C, a 64 bit, con thread

```
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
```

Applicazione client C++, a 64 bit, senza thread

```
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
```

Applicazione client C + +, 64 bit, con thread

```
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
```

Applicazione server C++, a 64 bit, senza thread

```
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
```

Applicazione server C++, a 64 bit, con thread

```
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
```

Uscita client C, a 64 bit, senza thread

```
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
```

Uscita client C, a 64 bit, con thread

```
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
```

Uscita server C, 64 bit, senza thread

```
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
```

Uscita server C, 64 bit, con thread

```
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
```

Preparazione dei programmi COBOL in Linux

Informazioni sulla preparazione di programmi COBOL in Linux e sulla preparazione di programmi COBOL utilizzando Micro Focus COBOL.

MQ_INSTALLATION_PATH rappresenta la directory di livello superiore in cui è installato IBM WebSphere MQ .

1. I copy book COBOL a 32 bit sono installati nella seguente directory:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   e i collegamenti simbolici vengono creati in:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. Sulle piattaforme a 64 bit, i copy book COBOL a 64 bit sono installati nella seguente directory:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. Nei seguenti esempi impostare COBCPY su:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

per applicazioni a 32 bit e:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

per applicazioni a 64 bit.

È necessario collegare il programma a uno dei seguenti:

File di libreria	Tipo di programma / uscita
libmqmcb.so	Server per COBOL
libmqicb.so	Client per COBOL
libmqmcb_r.so	Server per COBOL (applicazione con thread)
libmqicb_r.so	Client per COBOL (applicazione con thread)

Preparazione dei programmi COBOL utilizzando Micro Focus COBOL

Impostare le variabili di ambiente prima di compilare il proprio programma come segue:

```
export COBCPY=<COBCPY>
export LIB=MQ_INSTALLATION_PATHlib:$LIB
```

Per compilare un programma COBOL a 32 bit, dove supportato, utilizzando Micro Focus COBOL, immettere:

```
$ 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
```

Per compilare un programma COBOL a 64 bit utilizzando Micro Focus COBOL, immettere:

```
$ 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
```

dove `amqsput` è un programma di esempio

Per una descrizione delle variabili di ambiente necessarie, consultare la documentazione Micro Focus COBOL.

Creazione dell'applicazione su Solaris

Queste informazioni descrivono le attività aggiuntive e le modifiche alle attività standard che è necessario eseguire durante la creazione di applicazioni WebSphere MQ per Solaris da eseguire in Solaris.

Sono supportati i linguaggi di programmazione COBOL, C e C++. Per informazioni sulla preparazione dei programmi C++, consultare Utilizzo di C++.

Oltre a codificare le chiamate MQI nel codice sorgente, è necessario aggiungere i file di inclusione appropriati. Acquisire familiarità con il contenuto di questi file. Consultare “File di definizione dati IBM WebSphere MQ” a pagina 81 per una descrizione completa.

In questo argomento, il carattere barra retroversa (\) viene utilizzato per suddividere i comandi lunghi su più di una riga. Non immettere questo carattere, immettere ogni comando come riga singola.

Preparazione di programmi C in Solaris

I programmi C precompilati vengono forniti nella directory *MQ_INSTALLATION_PATH*/samp/bin .

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Per ulteriori informazioni sulla programmazione di applicazioni a 64 bit, consultare Coding standards on 64 - bit platforms.

Se si desidera utilizzare i programmi su una macchina su cui è installato solo il client MQI WebSphere MQ per Solaris, compilare i programmi per collegarli alla libreria client (-lmqic).

Se si utilizza il compilatore non supportato ?usr?ucb?cc, l'applicazione potrebbe essere compilata e collegata correttamente. Tuttavia, quando si esegue l'applicazione, l'operazione ha esito negativo quando tenta di connettersi al gestore code.

Nota: I client SSL e TLS Solaris x86 a 32 bit configurati per il funzionamento conforme a FIPS 140-2 non vengono eseguiti correttamente sui sistemi Intel. Questo errore si verifica perché il file di libreria a 32 bit GSKit-Crypto Solaris x86 conforme a FIPS 140-2 non viene caricato sul chipset Intel. Sui sistemi interessati, viene riportato l'errore AMQ9655 nel log degli errori del client. Per risolvere questo problema, disabilitare la conformità a FIPS 140-2 o ricompilare l'applicazione client a 64 bit, poiché il codice a 64 bit non viene interessato.

Collegamento di librerie

È necessario collegarsi con le librerie WebSphere MQ appropriate per il tipo di applicazione:

File di libreria	Tipo di programma / uscita
libmqm.so	Server per C
libmqic.so & libmqm.so	Client per C

Nota:

1. Se si sta scrivendo un servizio installabile (per ulteriori informazioni, consultare Amministrazione), collegarsi alla libreria libmqmzf.so .
2. Se si sta producendo un'applicazione per il coordinamento esterno da parte di un gestore transazioni compatibile con XA come IBM TXSeries Encinao BEA Tuxedo, è necessario collegarsi alle librerie libmqmxa.so (o libmqmxa64.so se il gestore transazioni considera il tipo 'long' come 64 bit) e libmqz.so .
3. È necessario collegare le librerie WebSphere MQ prima di qualsiasi altra libreria del prodotto.

Creazione di applicazioni su x86-64

Questo argomento contiene esempi di comandi utilizzati per creare programmi in vari ambienti sulla piattaforma x86-64 .

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Applicazione client C, a 32 bit

```
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
```

Applicazione client C, 64 bit

```
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
```

Applicazione server C, 32 bit

```
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
```

Applicazione server C, 64 bit

```
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
```

Applicazione client C++, a 32 bit

```
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
```

Applicazione client C + +, 64 bit

```
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
```

Applicazione server C++, a 32 bit

```
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
```

Applicazione server C++, 64 bit

```
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
```

Uscita client C, 32 bit

```
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
```

Uscita client C, 64 bit

```
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
```

Uscita server C, a 32 bit

```
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
```

Uscita server C, 64 bit

```
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
```

Creazione di applicazioni su SPARC

Questo argomento contiene esempi di comandi utilizzati per creare programmi in vari ambienti sulla piattaforma SPARC.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Applicazione client C, a 32 bit

```
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
```

Applicazione client C, 64 bit

```
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
```

Applicazione server C, 32 bit

```
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
```

Applicazione server C, 64 bit

```
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
```

Applicazione client C++, a 32 bit

```
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
```

Applicazione client C + +, 64 bit

```
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
```

Applicazione server C++, a 32 bit

```
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
```

Applicazione server C++, 64 bit

```
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
```

Uscita client C, 32 bit

```
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
```

Uscita client C, 64 bit

```
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
```

Uscita server C, a 32 bit

```
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
```

Uscita server C, 64 bit

```
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
```

Preparazione dei programmi COBOL in Solaris

Informazioni sulla preparazione dei programmi COBOL in Solaris.

MQ_INSTALLATION_PATH rappresenta la directory di livello superiore in cui è installato IBM WebSphere MQ .

1. I copy book COBOL a 32 bit sono installati nella seguente directory:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   e i collegamenti simbolici vengono creati in:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. I copy book COBOL a 64 bit sono installati nella seguente directory:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. Nei seguenti esempi impostare COBCPY su:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   per applicazioni a 32 bit e:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

   per applicazioni a 64 bit.

Compilare i programmi utilizzando il compilatore Micro Focus. I file di copia che dichiarano le strutture si trovano in *MQ_INSTALLATION_PATH*/inc:

```
$ export LIB=MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="<COBCPY>"
```

Compilazione di programmi a 32 bit:

- `$ cob32 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb`

  Server per COBOL
- `$ cob32 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb`

  Client per COBOL
- `$ cob32 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r`

  Server con thread per COBOL
- `$ cob32 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r`

  Client con thread per COBOL

Compilazione di programmi a 64 bit:

- `$ cob64 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb`

  Server per COBOL
- `$ cob64 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb`

  Client per COBOL
- `$ cob64 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r`

  Server con thread per COBOL
- `$ cob64 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r`

  Client con thread per COBOL

dove *amqs0put0.cbl* è un programma di esempio.

È necessario collegare il programma a uno dei seguenti:

- libmqmcb.so

  Server per COBOL
- libmqicb.so

  Client per COBOL

Preparazione dei programmi CICS in Solaris

Informazioni sulla preparazione dei programmi CICS in Solaris.

Viene fornito un modulo switch XA per consentire il collegamento di CICS con WebSphere MQ:

Tabella 67. Codice essenziale per applicazioni CICS (Solaris)

Descrizione	**C (origine)**	**C (exec)**
Routine di inizializzazione XA	amqzscix.c	amqzsc - TXSeries per Solaris

Collegare sempre le transazioni con la libreria thread safe WebSphere MQ libmqm.so.

È possibile trovare ulteriori informazioni sul supporto delle transazioni CICS in Amministrazione.

Supporto TXSeries CICS

WebSphere MQ per Solaris supporta TXSeries CICS utilizzando l'interfaccia XA.

Scrivere i programmi WebSphere MQ caricati nella stessa regione CICS in C o COBOL. Non è possibile effettuare una combinazione di chiamate C e COBOL MQI nella stessa regione CICS . La maggior parte delle chiamate MQI nella seconda lingua utilizzata ha esito negativo con un codice motivo MQRC_HOBJ_ERROR.

Preparazione dei programmi COBOL CICS utilizzando Micro Focus COBOL

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Per utilizzare Micro Focus COBOL, attenersi alla seguente procedura:

1. Aggiungere WebSphere MQ modulo della libreria di runtime COBOL alla libreria di runtime utilizzando il seguente comando:

   ```
   cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
               MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe
   ```

   Nota: Con `cicsmkcobol`, WebSphere MQ non consente di effettuare chiamate MQI nel linguaggio di programmazione C dall'applicazione COBOL.

   Se le applicazioni esistenti hanno chiamate di questo tipo, spostare queste funzioni dalle applicazioni COBOL alla propria libreria, ad esempio, `myMQ.so`. Dopo lo spostamento, queste funzioni non includono la libreria WebSphere MQ `libmqmcbrt.o` durante la creazione dell'applicazione COBOL per CICS.

   Inoltre, se l'applicazione COBOL non effettua alcuna chiamata COBOL MQI, non collegare `libmqmz_r` con `cicsmkcobol`.

   Ciò crea il file del metodo del linguaggio Micro Focus COBOL e consente alla libreria COBOL di runtime CICS di chiamare WebSphere MQ su sistemi UNIX and Linux .

   Nota: Eseguire `cicsmkcobol` solo quando si installa uno dei seguenti prodotti:

   - Nuova versione o release di Micro Focus COBOL
   - Nuova versione o release di TXSeries per Solaris
   - Nuova versione o release di qualsiasi prodotto database supportato (solo per transazioni COBOL)
   - Nuova versione o release di WebSphere MQ
2. Esportare la seguente variabile di ambiente:

   ```
   COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
   ```
3. Convertire, compilare e collegare il programma immettendo:

   ```
   cicstcl -l COBOL -e <yourprog>.ccp
   ```

Preparazione dei programmi C CICS

Creare programmi CICS C utilizzando le funzioni standard di CICS :

1. Esportare ***una*** delle seguenti variabili di ambiente:
   - LDFLAGS = "-L*MQ_INSTALLATION_PATH*lib -lmqm_r" export LDFLAGS
   - USERLIB = "-L*MQ_INSTALLATION_PATH*lib -lmqm_r" export USERLIB
2. Convertire, compilare e collegare il programma immettendo:

```
cicstcl -l C amqscic0.ccs
```

Transazione di esempio CICS C

L'origine C di esempio per una transazione CICS WebSphere MQ è fornita da AMQSCIC0.CCS. La transazione legge i messaggi dalla coda di trasmissione SYSTEM.SAMPLE.CICS.WORKQUEUE sul gestore code predefinito e le inserisce nella coda locale con un nome coda contenuto nell'intestazione di trasmissione del messaggio. Eventuali errori vengono inviati alla coda SYSTEM.SAMPLE.CICS.DLQ. Utilizzare lo script MQSC di esempio AMQSCIC0.TST per creare queste code e le code di input di esempio.

Creazione dell'applicazione su sistemi Windows

Le pubblicazioni sui sistemi Windows descrivono come creare applicazioni eseguibili dai programmi scritti.

Questo argomento descrive le attività aggiuntive e le modifiche alle attività standard che è necessario eseguire quando si creano applicazioni WebSphere MQ per Windows da eseguire su sistemi Windows . Sono supportati i linguaggi di programmazione ActiveX, C, C + +, COBOL e Visual Basic. Per informazioni sulla preparazione dei programmi ActiveX , consultare Utilizzo dell'interfaccia COM (Component Object Model) (classi di automazione di WebSphere MQ per ActiveX). Per informazioni sulla preparazione dei programmi C++, consultare Utilizzo di C++.

Le attività che è necessario eseguire per creare un'applicazione eseguibile utilizzando WebSphere MQ per Windows variano in base al linguaggio di programmazione in cui è scritto il codice sorgente. Oltre a codificare le chiamate MQI nel codice di origine, è necessario aggiungere le istruzioni di linguaggio appropriate per includere i file di inclusione WebSphere MQ per Windows per la lingua che si sta utilizzando. Acquisire familiarità con il contenuto di questi file. Consultare “File di definizione dati IBM WebSphere MQ” a pagina 81 per una descrizione completa.

Creazione di applicazioni a 64 bit su Windows

Le applicazioni a 32 bit e a 64 bit sono supportate su IBM WebSphere MQ for Windows Version 7.5. L'eseguibile IBM WebSphere MQ e i file della libreria vengono forniti in entrambi i formati a 32 bit e a 64 bit, utilizzare la versione appropriata a seconda dell'applicazione che si sta gestendo.

File e librerie eseguibili

Le versioni a 32 bit e a 64 bit delle librerie IBM WebSphere MQ vengono fornite nelle seguenti ubicazioni:

Tabella 68. Ubicazione delle librerie IBM WebSphere MQ

Versione libreria	**Directory contenente i file della libreria**
32 bit	*MQ_INSTALLATION_PATH*\Tools\Lib
64 bit	*MQ_INSTALLATION_PATH*\Tools\Lib64

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Le applicazioni a 32 bit continuano a funzionare normalmente dopo la migrazione. I file a 32 bit si trovano nella stessa directory delle versioni precedenti del prodotto.

Se si desidera creare una versione a 64 bit, è necessario assicurarsi che il proprio ambiente sia configurato per utilizzare i file della libreria in *MQ_INSTALLATION_PATH*\Tools\Lib64. Assicurarsi che la variabile di ambiente LIB non sia impostata per la ricerca nella cartella contenente le librerie a 32 bit.

Preparazione di programmi C in Windows

Lavorare nel tipico ambiente Windows ; WebSphere MQ per Windows non richiede nulla di speciale.

Per ulteriori informazioni sulla programmazione di applicazioni a 64 bit, consultare Coding standards on 64 - bit platforms.

- Collegare i programmi con le librerie appropriate fornite da WebSphere MQ:

File di libreria	Tipo di programma / uscita
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	server per C a 32 bit
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	client per C a 32 bit
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	client per C a 32 bit con coordinamento della transazione
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	server per C a 64 bit
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	client per C a 64 bit
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	client per C a 64-bit con coordinamento delle operazioni

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Il seguente comando fornisce un esempio di compilazione del programma di esempio amqsget0 (utilizzando il compilatore Microsoft Visual C++).

Per applicazioni a 32 bit:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib
```

Per le applicazioni a 64 bit:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib
```

Nota:

- Se si sta scrivendo un servizio installabile (per ulteriori informazioni, consultare Amministrazione), è necessario collegarsi alla libreria mqmzf.lib .
- Se si sta producendo un'applicazione per il coordinamento esterno da un gestore transazioni conforme a XA come IBM TXSeries Encinao BEA Tuxedo, è necessario collegarsi alla libreria mqmxa.lib o mqmxa.lib .
- Se si sta scrivendo un'uscita CICS , collegarsi alla libreria mqmcics4.lib .
- È necessario collegare le librerie WebSphere MQ prima di qualsiasi altra libreria del prodotto.

- Le DLL devono trovarsi nel percorso (PATH) specificato.
- Se si utilizzano caratteri minuscoli quando possibile, è possibile spostarsi da WebSphere MQ per Windows a WebSphere MQ su sistemi UNIX and Linux , dove è necessario utilizzare caratteri minuscoli.

Preparazione di programmi CICS e Transaction Server

L'origine C di esempio per una transazione CICS WebSphere MQ è fornita da AMQSCIC0.CCS. Lo si crea utilizzando le funzionalità CICS standard. Ad esempio, per TXSeries per Windows 2000:

1. Impostare la variabile di ambiente (immettere il codice seguente su una sola riga):

```
set CICS_IBMC_FLAGS=-IMQ_INSTALLATION_PATH\Tools\C\Include;
%CICS_IBMC_FLAGS%
```

2. Impostare la variabile di ambiente USERLIB:

```
set USERLIB=MQM.LIB;%USERLIB%
```

3. Convertire, compilare e collegare il programma di esempio:

```
cicstcl -l IBMC amqscic0.ccs
```

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Ciò è descritto nel manuale *Transaction Server for Windows NT Application Programming Guide (CICS) V4*.

È possibile trovare ulteriori informazioni sul supporto delle transazioni CICS in Amministrazione.

Preparazione dei programmi COBOL in Windows

Utilizzare queste informazioni per preparare i programmi COBOL in Windowse preparare i programmi CICS e Transaction Server.

1. I copy book COBOL a 32 bit sono installati nella seguente directory: *MQ_INSTALLATION_PATH*\Tools\cobol\CopyBook.
2. I copybook COBOL a 64 bit sono installati nella seguente directory: *MQ_INSTALLATION_PATH*\Tools\cobol\CopyBook64
3. Nei seguenti esempi impostare CopyBook su:

```
CopyBook
```

   per applicazioni a 32 bit e:

```
CopyBook64
```

   per applicazioni a 64 bit.

MQ_INSTALLATION_PATH rappresenta la directory di livello superiore in cui è installato IBM WebSphere MQ .

Per preparare i programmi COBOL sui sistemi Windows , collegare il proprio programma a una delle seguenti librerie fornite da IBM WebSphere MQ:

File di libreria	Tipo di programma o uscita
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb	server a 32 bit per IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb	Server a 32 bit per Micro Focus COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb	Client a 32 bit per IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb	client a 32 bit per Micro Focus COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb	Server a 64 bit per IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb	Server a 64 bit per Micro Focus COBOL

File di libreria	Tipo di programma o uscita
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb	Client a 64 bit per IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb	Client a 64 bit per Micro Focus COBOL

Quando si esegue un programma nell'ambiente del client MQI, assicurarsi che la libreria DOSCALLS sia visualizzata prima di qualsiasi libreria COBOL o IBM WebSphere MQ .

È possibile utilizzare il compilatore IBM COBOL Set o Micro Focus COBOL a seconda del programma:

- I programmi che iniziano con `amqi` sono adatti per il compilatore IBM COBOL Set,
- I programmi che iniziano con `amqm` sono adatti per il compilatore Micro Focus COBOL e
- I programmi che iniziano con `amq0` sono adatti per il compilatore.

IBM e Micro Focus COBOL

Ricollegare qualsiasi programma IBM WebSphere MQ Micro Focus COBOL a 32 bit esistente utilizzando `mqmcb.lib` o `mqiccb.lib`, piuttosto che le librerie `mqmcbb` e `mqiccbb` .

Per compilare, ad esempio, il programma di esempio `amq0put0`, utilizzando IBM VisualAge COBOL:

1. Impostare la variabile di ambiente SYSLIB per includere il percorso ai copybook IBM WebSphere MQ VisualAge COBOL (immettere il seguente codice su una sola riga):

   ```
   set SYSLIB=MQ_INSTALLATION_PATH\
   Tools\Cobol\Copybook\VAcobol;%SYSLIB%
   ```

2. Per l'utilizzo sul server IBM WebSphere MQ :

   ```
   cob2 amq0put0.cbl -qlib "MQ_INSTALLATION_PATH\
   Tools\Lib\mqmcbb.lib"
   ```

3. Per l'utilizzo sul client IBM WebSphere MQ :

   ```
   cob2 amq0put0.cbl -qlib "MQ_INSTALLATION_PATH\
   Tools\Lib\mqiccbb.lib"
   ```

   Nota: Sebbene sia necessario utilizzare l'opzione del compilatore `CALLINT (SYSTEM)`, questo è il valore predefinito per `cob2`.

Per compilare, ad esempio, il programma di esempio `amq0put0`, utilizzando Micro Focus COBOL:

1. Impostare la variabile di ambiente COBCPY in modo che punti ai copybook COBOL IBM WebSphere MQ (immettere il seguente codice su una riga):

   ```
   set COBCPY=MQ_INSTALLATION_PATH\
   Tools\Cobol\Copybook
   ```

2. Compilare il programma per fornire un file oggetto:

   ```
   cobol amq0put0 LITLINK
   ```

3. Collegare il file oggetto al sistema di runtime.
   - Impostare la variabile di ambiente LIB per puntare alle librerie COBOL del compilatore.
   - Collegare il file oggetto da utilizzare sul server IBM WebSphere MQ :

     ```
     cbllink amq0put0.obj mqmcb.lib
     ```

- Oppure collegare il file oggetto da utilizzare sul client IBM WebSphere MQ :

```
cbllink amq0put0.obj mqiccb.lib
```

Preparazione dei programmi CICS e Transaction Server

Per compilare e collegare un programma TXSeries per Windows NT, V5.1 utilizzando IBM VisualAge COBOL:

1. Impostare la variabile di ambiente (immettere il codice seguente su una sola riga):

```
set CICS_IBMCOB_FLAGS=MQ_INSTALLATION_PATH\
Cobol\Copybook\VAcobol;%CICS_IBMCOB_FLAGS%
```

2. Impostare la variabile di ambiente USERLIB:

```
set USERLIB=MQMCBB.LIB
```

3. Tradurre, compilare e collegare il programma:

```
cicstcl -l IBMCOB myprog.ccp
```

Ciò è descritto in *Transaction Server for Windows NT, V4 Application Programming* Guide.

Per compilare e collegare un programma CICS per Windows V5 utilizzando Micro Focus COBOL:

- Impostare la variabile INCLUDE:

```
set
INCLUDE=<drive>:\<programname>\ibm\websphere\tools\c\include;
        <drive>:\opt\cics\include;%INCLUDE%
```

- Impostare la variabile di ambiente COBCPY:

```
 setCOBCPY=<drive>:\<programname>\ibm\websphere\tools\cobol\copybook;
           <drive>:\opt\cics\include
```

- Impostare le opzioni COBOL:
  - `set`
  - `COBOPTS=/LITLINK /NOTRUNC`

ed eseguire il seguente codice:

```
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
```

Preparazione dei programmi Visual Basic in Windows

Utilizzare queste informazioni quando si considerano programmi Visual Basic in Windows.

Nota: Non vengono fornite versioni a 64 bit dei file del modulo Visual Basic.

Per preparare programmi Visual Basic su Windows:

1. Creare un nuovo progetto.
2. Aggiungere il file del modulo fornito, CMQB.BAS, al progetto.
3. Aggiungere altri file del modulo forniti, se necessario:

CMQBB.BAS	Supporto MQAI
CMQCFB.BAS	Supporto PCF
CMQXB.BAS	Supporto uscite canale
CMQPSB.BAS	Pubblicazione/sottoscrizione

Consultare “Codifica in Visual Basic” a pagina 87 per informazioni sull'utilizzo della chiamata MQCONNXAny da Visual Basic.

Richiamare la procedura MQ_SETDEFAULTS prima di effettuare qualsiasi chiamata MQI nel codice del progetto. Questa procedura imposta le strutture predefinite richieste dalle chiamate MQI.

Specificare se si sta creando un server o client WebSphere MQ , prima di compilare o eseguire il progetto, impostando la variabile di compilazione condizionale *MqType*. Impostare *MqType* in un progetto Visual Basic su 1 per un server o su 2 per un client nel modo seguente:

1. Selezionare il menu Progetto.
2. Selezionare *Name* Proprietà (dove *Name* è il nome del progetto corrente).
3. Selezionare la scheda Crea nella finestra di dialogo.
4. Nel campo Argomenti di compilazione condizionale, immettere quanto segue per un server:

   ```
   MqType=1
   ```

   o questo per un client:

   ```
   MqType=2
   ```

Uscita di sicurezza SSPI

WebSphere MQ per Windows fornisce un'uscita di sicurezza sia per il client WebSphere MQ MQI che per il server WebSphere MQ . Si tratta di un programma di uscita del canale che fornisce l'autenticazione per canali WebSphere MQ utilizzando SSPI (Security Services Programming Interface). SSPI fornisce le funzionalità di sicurezza integrate dei sistemi Windows .

I pacchetti di sicurezza vengono caricati da security.dll o secur32.dll. Queste DLL vengono fornite con il sistema operativo.

L'autenticazione unidirezionale viene fornita utilizzando i servizi di autenticazione NTLM. L'autenticazione bidirezionale viene fornita utilizzando i servizi di autenticazione Kerberos .

Il programma di uscita di sicurezza viene fornito in formato origine e oggetto. È possibile utilizzare il codice oggetto così com' è oppure è possibile utilizzare il codice origine come punto di partenza per creare i propri programmi di uscita utente.

Vedi anche “Utilizzo dell'uscita di sicurezza SSPI su sistemi Windows” a pagina 170.

Introduzione alle uscite di sicurezza

Un'uscita di sicurezza forma una connessione sicura tra due programmi di uscita di sicurezza, in cui un programma funga da MCA (message channel agent) mittente e da MCA ricevente.

Il programma che avvia la connessione sicura, ovvero il primo programma che acquisisce il controllo dopo aver stabilito la sessione MCA, è noto come *iniziatore del contesto*. Il programma partner è noto come *acceptor di contesto*.

La seguente tabella mostra alcuni tipi di canale che sono iniziatori di contesto e i relativi accettatori di contesto associati.

Tabella 69. Iniziatori di contesto e relativi accettatori di contesto associati	
Iniziatore contesto	**Acceptor contesto**
CLNTCONN MQCHT	SVRCONN MQCHT
MQCH_DESTINATARIO	MQCH_SENDER
CLUSRCVR MQCHT	MQCHT_CLUSSDR

Il programma di uscita di sicurezza ha due punti di ingresso:

- **SC_NTLM**

  Utilizza i servizi di autenticazione NTLM, che forniscono l'autenticazione unidirezionale. NTLM consente ai server di verificare le identità dei propri client. Non consente ai client di verificare l'identità di un server o a un server di verificare l'identità di un altro. L'autenticazione NTLM è stata progettata per un ambiente di rete in cui si presume che i server siano originali.

- **SCY_KERBEROS**

  Utilizza i servizi di autenticazione reciproca Kerberos . Il protocollo Kerberos non presuppone che i server in un ambiente di rete siano originali. Le parti alle due estremità di una connessione di rete possono verificare l'identità dell'altra parte. In altre parole, i server possono verificare l'identit ... dei client e di altri server e i client possono verificare l'identit ... di un server.

Cosa fa l'uscita di sicurezza

Questo argomento descrive le operazioni dei programmi di uscita canale SSPI.

I programmi di uscita canale forniti forniscono l'autenticazione unidirezionale o bidirezionale (reciproca) di un sistema partner quando viene stabilita una sessione. Per un particolare canale, ogni programma di uscita ha un *principal* associato (simile a un ID utente, consultare “WebSphere MQ e principal Windows” a pagina 468). Una connessione tra due programmi di uscita è un'associazione tra i due principal.

Una volta stabilita la sessione sottostante, viene stabilita una connessione sicura tra due programmi di uscita di sicurezza (uno per l'MCA mittente e uno per l'MCA ricevente). La sequenza delle operazioni è la seguente:

1. Ogni programma è associato ad un particolare principal, ad esempio come risultato di un'operazione di login esplicita.
2. L'iniziatore del contesto richiede una connessione sicura con il partner dal pacchetto di sicurezza (per Kerberos, il partner indicato) e riceve un token (denominato token1). Il token viene inviato, utilizzando la sessione sottostante già stabilita, al programma partner.
3. Il programma partner (il programma di accettazione del contesto) passa token1 al pacchetto di sicurezza, che verifica che l'iniziatore del contesto sia autentico. Per NTLM, la connessione è ora stabilita.
4. Per l'uscita di sicurezza fornita da Kerberos(ovvero, per l'autenticazione reciproca), il pacchetto di sicurezza genera anche un secondo token (denominato token2), che l'acceptor del contesto restituisce all'iniziatore del contesto utilizzando la sessione sottostante.
5. L'iniziatore di contesto utilizza token2 per verificare che l'acceptor di contesto sia autentico.
6. A questo punto, se entrambe le applicazioni sono soddisfatte dell'autenticità del token del partner, viene stabilita la connessione sicura (autenticata).

WebSphere MQ e principal Windows

Il controllo accessi fornito da WebSphere MQ si basa sull'utente e sul gruppo. L'autenticazione fornita da Windows è basata sui principal, come utente e SPN (servicePrincipalName). Nel caso del nome servicePrincipal, molti di questi potrebbero essere associati a un singolo utente.

L'uscita di sicurezza SSPI utilizza i principal Windows rilevanti per l'autenticazione. Se l'autenticazione di Windows ha esito positivo, l'uscita passa l'ID utente associato al principal Windows a WebSphere MQ per il controllo accessi.

I principal Windows rilevanti per l'autenticazione variano a seconda del tipo di autenticazione utilizzata.

- Per l'autenticazione NTLM, il principal Windows per l'iniziatore del contesto è l'ID utente associato al processo in esecuzione. Poiché questa autenticazione è un modo, il principal associato a Context Acceptor è irrilevante.
- Per l'autenticazione Kerberos , sui canali CLNT, il principal Windows è l'ID utente associato con il processo in esecuzione. Altrimenti, il principal Windows è il nome servicePrincipalformato aggiungendo il prefisso seguente al nome QueueManager.

```
ibmMQSeries/
```

Utilizzo dei servizi LDAP (lightweight directory access protocol) con WebSphere MQ per Windows

Questo argomento spiega cos' è un servizio directory e la parte svolta da un DAP (directory access protocol). Spiega inoltre come le applicazioni WebSphere MQ possono utilizzare una directory LDAP (lightweight directory access protocol) utilizzando un programma di esempio come guida.

Nota: Il programma di esempio è progettato per una persona che ha già familiarità con LDAP.

I seguenti argomenti forniscono ulteriori informazioni sui servizi di directory, LDAP e sull'utilizzo di LDAP con WebSphere MQ.

- “Servizio di directory” a pagina 468
- “LDAP (Lightweight Directory Access Protocol)” a pagina 469
- “Utilizzo di LDAP con WebSphere MQ” a pagina 469

Servizio di directory

Una directory è un archivio di informazioni sugli oggetti, che è organizzato in modo tale che è facile trovare le informazioni su un oggetto specifico.

Un esempio comune è un elenco telefonico, in cui vengono memorizzate informazioni (indirizzo e numero di telefono) su persone e aziende. Un altro esempio è una rubrica per un sistema email, in cui gli indirizzi email e, facoltativamente, altre informazioni come i numeri di telefono, vengono memorizzati per le persone.

Sui sistemi di computer, le directory possono memorizzare informazioni sulle risorse del computer, come le stampanti o i dischi condivisi. Ad esempio, è possibile utilizzare una directory per individuare la posizione della stampante a colori più vicina. In un'applicazione WebSphere MQ è possibile utilizzare una directory per fornire l'associazione tra un servizio dell'applicazione (ad esempio, l'elaborazione dei crediti esigibili) e la coda da utilizzare per i messaggi che richiedono tale servizio (possibilmente identificati tramite il nome della coda e il nome del gestore code host).

Le directory sono implementate come sistemi client-server, in cui il server di directory contiene tutte le informazioni e risponde alle richieste dei client. I client possono essere programmi di interfaccia utente, che forniscono le informazioni direttamente all'utente, o programmi di applicazione che hanno bisogno di individuare le risorse per completare il loro lavoro. Un servizio directory comprende il server di directory, i programmi di gestione e le librerie client e i programmi necessari per configurare, aggiornare e leggere la directory.

LDAP (Lightweight Directory Access Protocol)

Esistono molti servizi di directory, come Novell Directory Services, DCE Cell Directory Service, Banyan StreetTalk, Windows Directory Services, X.500e i servizi di rubrica associati ai prodotti email. X.500 è stato proposto come standard per i servizi di directory globali dall'ISO (International Standards Organization). Richiede una pila di protocolli OSI per le sue comunicazioni, e in gran parte a causa di questo, il suo uso è stato limitato a grandi organizzazioni e istituzioni accademiche. Un server di directory X.500 comunica con i propri client utilizzando DAP (Directory Access Protocol).

LDAP (Lightweight Directory Access Protocol) è stato creato come versione semplificata di DAP. È più facile da implementare, omette alcune delle funzioni meno utilizzate di DAP e viene eseguito su TCP/IP. Come risultato di queste modifiche viene rapidamente adottato come protocollo di accesso alla directory per la maggior parte degli scopi, sostituendo la moltitudine di protocolli proprietari precedentemente utilizzati. I client LDAP possono ancora accedere a un server X.500 tramite un gateway (X.500 richiede ancora lo stack del protocollo OSI) o, in misura crescente, le implementazioni X.500 generalmente includono il supporto nativo per LDAP e l'accesso DAP.

Le directory LDAP possono essere distribuite e possono utilizzare la replica per consentire un accesso efficiente al contenuto.

Per una descrizione più completa di LDAP, consultare *Understanding LDAP*, una pubblicazione IBM Redbooks .

Utilizzo di LDAP con WebSphere MQ

Nelle configurazioni WebSphere MQ , le informazioni che definiscono le code di trasmissione e di messaggi sono memorizzate localmente. Ciò significa che in una rete WebSphere MQ le varie definizioni vengono distribuite, senza alcuna directory centrale di queste informazioni disponibile per la ricerca. La messaggistica remota tra applicazioni WebSphere MQ è generalmente ottenuta mediante l'utilizzo di definizioni locali di code remote. L'applicazione emette prima una chiamata MQOPEN utilizzando il nome specificato nella definizione locale della coda remota. Per inserire il messaggio nella coda remota, l'applicazione emette MQPUT, specificando l'handle restituito dalla chiamata MQOPEN. La definizione della coda remota fornisce il nome della coda di destinazione, il gestore code di destinazione e, facoltativamente, una coda di trasmissione. In questa tecnica l'applicazione deve conoscere al runtime il nome specificato nella definizione della coda locale.

Una variazione rispetto al precedente evita l'uso di definizioni locali di code remote. L'applicazione può specificare il nome completo della coda di destinazione, che include il nome gestore code remoto come parte di MQOPEN. L'applicazione deve quindi conoscere questi due nomi al runtime. Il gestore code locale deve essere configurato correttamente con la definizione della coda locale e con una coda di trasmissione opportunamente denominata (o predefinita) e un canale associato che viene consegnato alla destinazione.

Nel caso in cui i gestori code di origine e di destinazione siano definiti come membri dello stesso cluster, gli aspetti della coda di trasmissione e del canale dei precedenti due scenari possono essere ignorati. Se la coda di trasmissione di destinazione è una coda cluster, anche una definizione locale di una coda remota non è richiesta. Tuttavia, analogamente ai casi precedenti descritti, l'applicazione deve ancora conoscere il nome della coda di destinazione.

Un servizio di directory può essere utilizzato per rimuovere questa dipendenza dell'applicazione sui nomi della coda (o la combinazione dei nomi della coda e del gestore code). L'associazione tra criteri dell'applicazione e nomi oggetto WebSphere MQ può essere conservata in una directory e aggiornata in modo dinamico e indipendentemente dalle applicazioni. In fase di runtime, l'applicazione WebSphere MQ che desidera inviare un messaggio interroga prima la directory utilizzando i criteri basati sull'applicazione, ad esempio, dove: nome_servizio = "conti esigibili", richiama i nomi oggetto WebSphere MQ pertinenti e utilizza questi valori restituiti nella chiamata MQOPEN.

Un altro esempio di utilizzo di una directory è per un'azienda che ha molti piccoli depot o uffici, i client WebSphere MQ MQI possono essere utilizzati per inviare messaggi ai server WebSphere MQ che si trovano negli uffici più grandi. I client devono conoscere il nome della macchina host, il canale MQI e il nome coda per ogni server a cui inviano messaggi. Occasionalmente, potrebbe essere necessario

spostare un server WebSphere MQ su un'altra macchina; ogni client che comunica con il server dovrebbe essere a conoscenza della modifica. Un servizio di directory LDAP può essere utilizzato per memorizzare i nomi delle macchine host (e i nomi di canale e coda) e i programmi client possono richiamare le informazioni dalla directory ogni volta che desiderano inviare un messaggio a un server. In questo caso, è necessario aggiornare solo la directory se è stato modificato un nome host (o un nome canale o coda).

Più destinazioni per un messaggio dell'applicazione potrebbero essere memorizzate in una directory, con quella scelta che dipende da considerazioni sulla disponibilità o sulla condivisione del carico.

WebSphere MQ può anche utilizzare una directory LDAP per memorizzare le informazioni di autenticazione da utilizzare con SSL (Secure Sockets Layer). Le classi WebSphere MQ per Java possono anche memorizzare le informazioni in una directory LDAP.

Programma di esempio LDAP

Il programma di esempio è progettato per chi ha familiarità con LDAP e probabilmente lo utilizza già. Ha lo scopo di mostrare in che modo le applicazioni WebSphere MQ possono utilizzare una directory LDAP.

Creazione del programma di esempio

Questo programma è stato creato e testato solo su Windows utilizzando TCP/IP. Oltre alle considerazioni generali riportate in “Preparazione di programmi C in Windows” a pagina 461, tenere presente quanto segue:

- Questo programma è progettato per essere eseguito come programma client, quindi deve essere collegato a MQIC MQIC.LIB .
- Oltre alle librerie e ai file di intestazione WebSphere MQ , questo programma deve essere creato utilizzando le librerie e i file di intestazione del client LDAP.

  Ad esempio, utilizzando il client IBM eNetwork , collegare il programma alle librerie LIBLDAPSTATICE.LIB e LIBLBERSTATICSSL.LIB .

configurazione della directory

Prima di poter eseguire il programma di esempio, è necessario configurare un server di directory LDAP con i dati di esempio.

Il file MQuser.ldif, nella directory `tools\c\samples` , contiene alcuni dati di esempio in LDIF (LDAP Data Interchange Format). È possibile modificare questo file in base alle proprie esigenze. Contiene dati per una società fittizia chiamata MQuser che ha un dipartimento dei trasporti composto da tre uffici. Ciascuno di questi uffici dispone di una macchina che esegue un server WebSphere MQ .

Come minimo, è necessario modificare le tre linee che contengono i nomi host delle macchine su cui sono in esecuzione WebSphere MQ server: righe 18, 27 e 36:

```
host: LondonHost
  ...
host: SydneyHost
  ...
host: WashingtonHost
```

È necessario modificare `LondonHost`, `SydneyHoste` `WashingtonHost` con i nomi di tre macchine su cui sono in esecuzione i server WebSphere MQ . È anche possibile modificare i nomi dei canali e delle code, se lo si desidera (l'esempio utilizza i nomi dei valori predefiniti del sistema). È inoltre possibile aumentare o diminuire il numero di uffici nei dati di esempio.

Configurazione del server di directory IBM Tivoli

Fare riferimento al manuale IBM Tivoli Directory Server (ITDS) Administrator's Guide per informazioni sull'installazione della directory. Nell'argomento `Installing and Configuring Server`, consultare le sezioni `Installing Server` e `Basic Server Configuration`. Se necessario, leggere l'argomento `Administrator Interface` per familiarizzare con il funzionamento dell'interfaccia.

Nell'argomento Configuring - How Do I, seguire le istruzioni per avviare l'amministratore, quindi consultare la sezione Configure Database e creare un database predefinito. Ignorare la sezione Configure replica e utilizzare la sezione Work with Suffixes, aggiungere un suffisso o=MQuser.

Prima di aggiungere qualsiasi voce al database, è necessario estendere lo schema di directory aggiungendo alcune definizioni di attributo e una definizione di classe oggetto. Ciò è descritto nel manuale IBM Tivoli Directory Server Administrator's Guide nel capitolo Reference Information nella sezione Directory Schema. Sono inclusi due file di esempio. Il file mq.at.conf include le definizioni di attributo che è possibile aggiungere al file ?etc?slapd.at.conf. Eseguire questa azione includendo il file di esempio modificando slapd.at.conf e aggiungendo una riga:

```
include <pathname>/mq.at.conf
```

In alternativa, è possibile modificare il file slapd.at.conf e aggiungere il contenuto del file di esempio direttamente ad esso, ovvero aggiungere le seguenti righe:

```
# 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
```

Allo stesso modo per la definizione della classe di oggetti, è possibile includere il file di esempio modificando etc?slapd.oc.conf e aggiungere la riga:

```
include <pathname>/mq.oc.conf
```

oppure è possibile aggiungere il contenuto del file di esempio direttamente a slapd.oc.conf, , ovvero aggiungere le righe:

```
# MQ object classdefinition
objectclass mqApplication
      requires
              objectClass,
              cn,
              host,
              mqChannel,
              mqQueue
      allows
              mqQueueManager,
              mqPort,
              description,
              l,
              ou,
              seeAlso
```

È ora possibile avviare il server di directory (Amministrazione, Server, Avvio) e aggiungere le voci di esempio. Per aggiungere le voci di esempio, andare alla pagina Amministrazione, Aggiungi voci dell'amministratore, immettere il percorso completo del file di esempio MQuser.ldif e fare clic su Inoltra.

Il server di directory è ora in esecuzione e caricato con i dati adatti per l'esecuzione del programma di esempio.

Configurazione del server di directory Netscape

Utilizzando la pagina Amministrazione di Netscape Server, fare clic su **Crea nuovo Netscape Directory Server**.

Ora dovrebbe essere presentato un modulo contenente le informazioni di configurazione. Modificare il suffisso di directory in **o = MQuser** e aggiungere una password per l'utente non limitato. È anche possibile modificare qualsiasi altra informazione per adattarla alla propria installazione. Fare clic su **OK**e la directory deve essere creata correttamente. Fare clic su **Torna a Amministrazione server** e avviare il

server di directory. Fare clic sul nome della directory per avviare il server di gestione di Directory Server per la nuova directory.

Prima di aggiungere qualsiasi voce al database, estendere lo schema di directory aggiungendo alcune definizioni di attributo e una definizione di classe oggetto. Fare clic sulla scheda **Schema** della pagina Directory Server. Viene ora visualizzato un modulo che consente di aggiungere nuovi attributi. Aggiungere i seguenti attributi (lasciare vuoto l'OID attributo per tutti):

```
Attribute Name              Syntax
--------------              ------
mqChannel                   Case Exact String
mqQueueManager              Case Exact String
mqQueue                     Case Exact String
mqPort                      Integer
```

Aggiungere un nuovo objectClass facendo clic su **Crea ObjectClass** nel pannello laterale. Immettere **mqApplication** come ObjectClass Name, selezionare **applicationProcess** come ObjectClass parent e lasciare vuoto l' **ObjectClass OID** . Ora aggiungere alcuni attributi a objectClass. Selezionare **host**, **mqChannel**e **mqQueue** come Attributi richiesti e selezionare **mqQueueManager** e **mqPort** come Attributi consentiti. Premere il pulsante **Crea nuova ObjectClass** per creare objectClass.

Per aggiungere i dati di esempio, fare clic sulla scheda **Gestione database** e selezionare **Aggiungi voci** dal pannello laterale. Immettere il percorso del file di dati di esempio `<pathname>\MQuser.ldif`, immettere la password e fare clic su **OK**.

Il programma di esempio viene eseguito come utente non autorizzato e, per impostazione predefinita, Netscape Directory non consente agli utenti non autorizzati di ricercare la directory. Modificare questa impostazione facendo clic sulla scheda **Controllo accessi** . Immettere la password per l'utente senza limitazioni e fare clic su **OK** per caricare le voci di controllo accessi per la directory. Tali valori devono essere attualmente vuoti. Premere il pulsante **Nuovo ACI** per creare una nuova voce di controllo accessi. Nella casella di immissione visualizzata, fare clic su **Nega** (sottolineata) e nella finestra risultante, modificarla in **Consenti**. Aggiungere un nome, ad esempio, **MQuser - access**e fare clic su **scegli un suffisso** per selezionare **o = MQuser**. Immettere **o = MQuser** come destinazione, immettere la password per l'utente senza limitazioni e fare clic su **Inoltra**.

Il server di directory è ora in esecuzione e caricato con i dati adatti per l'esecuzione del programma di esempio.

Esecuzione del programma di esempio

È necessario disporre di un server di directory LDAP in esecuzione e popolato con i dati di esempio.
I dati specificano tre macchine host, tutte su cui devono essere in esecuzione server WebSphere MQ . Assicurarsi che il gestore code predefinito sia in esecuzione su ciascuna macchina (a meno che non siano stati modificati i dati di esempio per specificare un gestore code differente).

Inoltre, avviare il programma listener WebSphere MQ su ciascuna macchina; l'esempio utilizza il protocollo TCP/IP con il numero di porta predefinito WebSphere MQ , in modo che sia possibile avviare il listener con il comando:

```
runmqlsr -t tcp
```

Per verificare l'esempio, è anche possibile eseguire un programma per leggere i messaggi in arrivo su ciascun server WebSphere MQ , ad esempio, è possibile utilizzare il programma di esempio amqstrg:

```
amqstrg SYSTEM.DEFAULT.LOCAL.QUEUE
```

Il programma di esempio utilizza tre variabili di ambiente, una obbligatoria e due facoltative. La variabile richiesta è LDAP_BASEDN, che specifica il DN (Distinguished Name) di base per la ricerca nell'indirizzario.

Per gestire i dati di esempio, impostarli su `ou=Transport, o=MQuser`, ad esempio, su un prompt dei comandi su sistemi Windows , immettere:

```
set LDAP_BASEDN=ou=Transport, o=MQuser
```

Le variabili facoltative sono LDAP_HOST e LDAP_VERSION. La variabile LDAP_HOST specifica il nome dell'host su cui è in esecuzione il server LDAP; se non è specificato, assume il valore predefinito dell'host locale. La variabile LDAP_VERSION specifica la versione del protocollo LDAP da utilizzare e può essere 2 o 3. La maggior parte dei server LDAP ora supporta la versione 3 del protocollo; tutti supportano la versione precedente 2. Questo esempio funziona ugualmente bene con entrambe le versioni del protocollo e, se non viene specificato, il valore predefinito è la versione 2.

È ora possibile eseguire l'esempio immettendo il nome del programma seguito dal nome dell'applicazione WebSphere MQ a cui si desidera inviare i messaggi, nel caso dei dati di esempio i nomi dell'applicazione sono Londra, Sydney e Washington. Ad esempio, per inviare messaggi all'applicazione di Londra:

```
amqsldpc London
```

Se il programma non riesce a connettersi al server WebSphere MQ , viene visualizzato un messaggio di errore appropriato. Se si connette correttamente, è possibile iniziare a digitare i messaggi, ogni riga immessa (terminata da < return> o < enter>) viene inviata come un messaggio separato, una riga vuota termina il programma.

Progettazione del programma

Il programma ha due parti distinte: la prima parte utilizza le variabili di ambiente e il valore della riga comandi per interrogare un server di directory LDAP; la seconda parte stabilisce la connessione WebSphere MQ utilizzando le informazioni restituite dalla directory e invia i messaggi.

Le chiamate LDAP utilizzate nella prima parte del programma differiscono leggermente a seconda che si stia utilizzando LDAP versione 2 o 3 e sono descritte in dettaglio dalla documentazione fornita con le librerie del client LDAP. Questa sezione fornisce una breve descrizione.

La prima parte del programma verifica che sia stato richiamato correttamente e legge le variabili di ambiente. Stabilisce quindi una connessione con il server di directory LDAP sull'host specificato:

```
if (ldapVersion == LDAP_VERSION3)
{
  if ((ld = ldap_init(ldapHost, LDAP_PORT)) == NULL)
     ...
}
else
{
  if ((ld = ldap_open(ldapHost, LDAP_PORT)) == NULL )
     ...
}
```

Una volta stabilita una connessione, il programma imposta alcune opzioni sul server con la chiamata "ldap_set_option" e quindi si autentica sul server collegandosi ad esso:

```
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)
     ...
}
```

Nel programma di esempio `bindDN` e `password` sono impostati su NULL, il che significa che il programma si autentica come utente anonimo, vale a dire che non dispone di diritti di accesso speciali

e può accedere solo alle informazioni disponibili pubblicamente. In pratica, la maggior parte delle organizzazioni limita l'accesso alle informazioni che memorizzano nelle directory in modo che solo gli utenti autorizzati possano accedervi.

Il primo parametro della chiamata di bind `ld` è un handle utilizzato per identificare questa particolare sessione LDAP nel resto del programma. Dopo l'autenticazione, il programma ricerca nella directory le voci che corrispondono al nome dell'applicazione:

```
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           */
```

Si tratta di una semplice chiamata sincrona al server che restituisce direttamente i risultati. Esistono altri tipi di ricerca più appropriati per query complesse o quando è previsto un numero elevato di risultati. Il primo parametro della ricerca è l'handle `ld` che identifica la sessione. Il secondo parametro è il DN (distinguished name) di base, che specifica dove deve iniziare la ricerca nella directory e il terzo parametro è l'ambito della ricerca, ovvero le voci relative al punto di partenza vengono ricercate. Questi due parametri insieme definiscono quali voci nell'indirizzario vengono ricercate. Il parametro successivo, `filterPattern` , specifica cosa si sta cercando. Il parametro `attrs` elenca gli attributi che si desidera richiamare dall'oggetto quando lo si trova. L'attributo successivo indica se si desiderano solo gli attributi o i relativi valori; l'impostazione su FALSE indica che si desiderano i valori degli attributi. Il parametro finale viene utilizzato per restituire il risultato.

Il risultato potrebbe contenere molte voci di directory, ciascuna con gli attributi specificati e i relativi valori. Dobbiamo estrarre i valori che vogliamo dal risultato. In questo programma di esempio ci si aspetta che venga trovata una sola voce, quindi si guarda solo la prima voce nel risultato:

```
ldapEntry = ldap_first_entry(ld, ldapResult);
```

Questa chiamata restituisce un handle che rappresenta la prima voce e viene impostato un loop for per estrarre tutti gli attributi dalla voce:

```
for (attribute = ldap_first_attribute(ld, ldapEntry, &ber);
     attribute != NULL;
     attribute = ldap_next_attribute(ld, ldapEntry, ber ))
{
```

Per ciascuno di questi attributi, vengono estratti i relativi valori associati. Ancora una volta si prevede un solo valore per attributo, quindi si utilizza solo il primo valore; si determina quale attributo si ha e si memorizza il valore nella variabile di programma appropriata:

```
values = ldap_get_values(ld, ldapEntry, attribute);
if (values != NULL && values[0] != NULL)
{
  if (stricmp(attribute, MQ_HOST_ATTR) == 0)
  {
    mqHost = strdup(values[0]);
    ...
```

Infine si riordinano liberando memoria (ldap_value_free, ldap_memfree, ldap_msgfree) e si chiude la sessione *unbinding* dal server:

```
ldap_unbind(ld);
```

Verifichiamo di aver trovato tutti i valori di WebSphere MQ di cui abbiamo bisogno dalla directory e, in tal caso, richiamiamo sendMessages() per connettersi al server WebSphere MQ e inviare i messaggi WebSphere MQ .

La seconda parte del programma di esempio è la routine sendMessages() che contiene tutte le chiamate WebSphere MQ . Ciò è modellato sul programma di esempio amqsput0 , le differenze sono che i parametri del programma sono stati estesi e viene utilizzato MQCONNX invece della chiamata MQCONN.

Sviluppo di applicazioni IBM WebSphere MQ Telemetry

Le applicazioni di telemetria integrano i dispositivi di rilevamento e controllo con altre fonti di informazione disponibili su Internet e nelle imprese.

Sviluppare applicazioni per IBM WebSphere MQ Telemetry utilizzando modelli di progettazione, esempi di lavoro, programmi di esempio, concetti di programmazione e informazioni di riferimento. Utilizzare il daemon IBM WebSphere MQ Telemetry per le unità per semplificare la connessione di molte piccole unità a IBM WebSphere MQ.

Concetti correlati

WebSphere MQ Telemetry

Concetti e scenari di telemetria per monitoraggio e controllo

Attività correlate

Installazione di WebSphere MQ Telemetry

Amministrazione di WebSphere MQ Telemetry

Risoluzione dei problemi per WebSphere MQ Telemetry

Riferimenti correlati

Riferimento WebSphere MQ Telemetry

IBM WebSphere MQ Telemetry Programmi di esempio

Gli script di esempio vengono forniti per dimostrare l'utilizzo di base di MQ Telemetry Transport v3 Client. Utilizzare gli script per pubblicare un messaggio e sottoscrivere un argomento.

Prima di iniziare

Avviare il servizio di telemetria (MQXR) per eseguire i programmi di esempio.

L'ID utente deve essere un membro del gruppo utenti mqm.

Eseguire prima lo script SampleMQM , seguito da uno script MQTTV3Sample per eseguire una pubblicazione e sottoscrizione. Eseguire lo script di esempio CleanupMQM per eliminare il gestore code creato dallo script SampleMQM .

Poiché lo script SampleMQM crea e utilizza un gestore code denominato QM1, non viene eseguito inalterato su un sistema con un gestore code QM1 . Qualsiasi modifica apportata potrebbe avere implicazioni per la configurazione del gestore code esistente.

Informazioni su questa attività

- L'applicazione SampleMQM crea e avvia un gestore code abilitato alla telemetria denominato QM1. Lo script, inoltre, imposta una coda di trasmissione predefinita per QM1e crea e avvia un canale predefinito in ascolto sulla porta 1883. Questo canale non esegue alcuna autenticazione dei client ad esso connessi. Il canale ha l'attributo MCAUSER (message channel agent user identifier), impostato su 'guest' su sistemi Windows o 'nobody' su sistemi Linux . I client connessi al canale vengono trattati come l'utente 'guest' o 'nobody', a seconda del sistema su cui è in esecuzione. Lo script autorizza 'guest' sui sistemi Windows e 'nobody ' sui sistemi Linux a pubblicare e sottoscrivere qualsiasi argomento su QM1
- L'applicazione MQTTV3Sample si trova nella seguente ubicazione;
  - Su Windows *MQ_INSTALLATION_PATH*\mqxr\samples

    dove *MQ_INSTALLATION_PATH* è l'ubicazione in cui è installato IBM WebSphere MQ .
  - Su Linux *MQ_INSTALLATION_PATH*/mqxr/samples

L'applicazione MQTTV3Sample funziona come un publisher, inviando un singolo messaggio ad un argomento sul server. Funge anche da sottoscrittore, ascoltando i messaggi dal server.

- Lo script di esempio CleanupMQM termina ed elimina QM1 creato dallo script SampleMQM . Utilizzare lo script di esempio CleanupMQM se si desidera eseguire nuovamente lo script SampleMQM o rimuovere QM1.

Procedura

1. Immettere il seguente comando su una riga comandi per eseguire lo script SampleMQM
   - Su Windows, il comando per eseguire lo script SampleMQM è il seguente:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\SampleMQM.bat
     ```

   - Su AIX e Linux, il comando per eseguire lo script SampleMQM è il seguente:

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/SampleMQM.sh
     ```

   dove *MQ_INSTALLATION_PATH* è l'ubicazione in cui è installato IBM WebSphere MQ .

   Viene creato un gestore code denominato MQXR_SAMPLE_QM.
2. Immettere il seguente comando per eseguire la prima parte dello script MQTTV3Sample ;
   - Su Windows, su una riga comandi, immettere il comando seguente:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\RunMQTTV3Sample.bat -a subscribe
     ```

   - Su AIX e Linux, in una finestra della shell, immettere il comando seguente:

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh -a subscribe
     ```

3. Digitare il seguente comando per eseguire la seconda parte dello script MQTTV3Sample ;
   - Su Windows, su un'altra riga comandi, immettere il seguente comando;

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\RunMQTTV3Sample.bat -m "Hello from an MQTT v3
     application"
     ```

   - Su AIX e Linux, in un'altra finestra della shell, immettere il comando seguente:

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh -m "Hello from an MQTT v3 application"
     ```

4. Per rimuovere il gestore code creato dallo script di SampleMQM , è possibile eseguire lo script CleanupMQM utilizzando il seguente comando;
   - Su Windows, immettere il seguente comando;

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\CleanupMQM.bat
     ```

   - Su AIX e Linux in un'altra finestra della shell, immettere il seguente comando;

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/CleanupMQM.sh
     ```

Risultati

Il messaggio `Hello from an MQTT v3 application` immesso nella seconda finestra verrà pubblicato dall'applicazione e ricevuto dall'applicazione nella prima finestra. L'applicazione nella prima finestra la mostrerà sullo schermo.

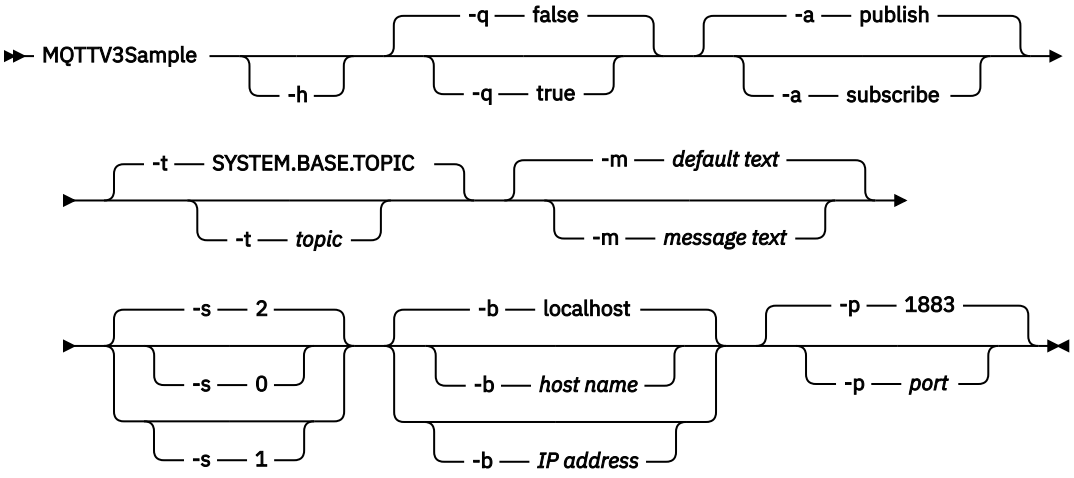
Programma MQTTV3Sample

Informazioni di riferimento sulla sintassi di esempio e sui parametri per il programma MQTTV3Sample .

Finalità

Il programma MQTTV3Sample può essere utilizzato per pubblicare un messaggio e sottoscrivere un argomento.

MQTTV3Sample syntax



Parametri

-h
Stampare questo testo di aiuto e uscire

-q
Impostare la modalità silenziosa, invece di utilizzare la modalità predefinita false.

-a
Impostare la pubblicazione o la sottoscrizione, invece di assumere l'azione predefinita di pubblicazione.

-t
Pubblica o sottoscrivi l'argomento, invece di pubblicare o sottoscrivere l'argomento predefinito

-m
Pubblica il testo del messaggio invece di inviare il testo della pubblicazione predefinito, "Hello from an MQTT v3 application".

-s
Impostare QoS invece di utilizzare il valore predefinito QoS, 2.

-b
Connettersi a questo nome host o indirizzo IP invece di connettersi al nome host predefinito, localhost.

-p
Utilizzare questa porta invece di utilizzare il valore predefinito, 1883.

Eseguire il programma MQTTV3Sample

Per sottoscrivere un argomento su Windows, utilizzare il comando:

```
runMQTTV3Sample -a subscribe
```

Per pubblicare un messaggio su Windows, utilizzare il comando:

```
runMQTTV3Sample
```

Per ulteriori informazioni sull'esecuzione degli script di esempio forniti, consultare “IBM WebSphere MQ Telemetry Programmi di esempio” a pagina 475.

Creazione della tua prima applicazione publisher di MQ Telemetry Transport utilizzando Java

I passi per creare un'applicazione client MQTT sono descritti in modo didattico. Ogni riga di codice viene spiegata. Alla fine dell'attività, verrà creato un publisher MQTT. È possibile sfogliare le pubblicazioni utilizzando WebSphere MQ Explorer.

Prima di iniziare

Installa la funzione WebSphere MQ Telemetry su un server su cui è installato IBM WebSphere MQ Version 7.1 o versioni successive.

L'applicazione client utilizza il pacchetto `com.ibm.mq.micro.client.mqttv3` in IBM WebSphere MQ Telemetry SDK (Software Development Toolkit). L'SDK fa parte dell'installazione di IBM WebSphere MQ Telemetry . Il client si connette alla funzione IBM WebSphere MQ Telemetry per scambiare messaggi con IBM WebSphere MQ.

È inoltre necessario installare gli aggiornamenti di telemetria per IBM WebSphere MQ Explorer Version 7.1 per amministrare IBM WebSphere MQ Telemetry. Gli aggiornamenti fanno parte dell'installazione di IBM WebSphere MQ Telemetry .

Un client MQTT, in esecuzione su Java SE, richiede la versione 6.0 di Java SE o successiva. IBM Java SE v6.0 fa parte dell'installazione di IBM WebSphere MQ Version 7.1 . Si trova in *WebSphere MQ installation directory*`\java\jre`

Informazioni su questa attività

L'esempio è un'applicazione di pubblicazione, PubSync. PubSync pubblica `Hello World` sull'argomento `MQTT Examples`e attende la conferma che la pubblicazione è stata consegnata al gestore code.

Impostando una sottoscrizione durevole a `MQTT Examples` , è possibile verificare che l'applicazione funzioni.

La procedura utilizza Eclipse per sviluppare, creare ed eseguire il client. È possibile scaricare Eclipse dal sito Web del progetto Eclipse all'indirizzo www.eclipse.org.

Per creare l'applicazione, è possibile creare i file Java e compilarli ed eseguirli utilizzando la riga comandi.

In una nuova directory, creare il percorso di directory `.\com\ibm\mq\id`. Creare due file Java, `Example.java` e PubSync`.java`. Copiare il codice da “Codice di esempio” a pagina 482 nei file Java.

Compilare il codice Java utilizzando il comando,

```
javac -cp jar_dir\com.mq.micro.client.mqttv3.jar
       com.ibm.mq.id.PubSync.java com.ibm.mq.id.Example.java
```

Eseguire PubSync utilizzando il comando,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubSync
```

Procedura

1. Creare un progetto Java in Eclipse.
   a) **File> Nuovo> Progetto Java** e immettere un nome progetto. Fare clic su **Avanti**.

      Verificare che il JRE sia alla versione corretta o successiva. Java SE deve essere alla versione 6.0 o successiva.

   b) Nella pagina Impostazioni Java, fare clic su **Librerie> Aggiungi jar esterni ...**
   c) Passare alla directory in cui è stata installata la cartella WebSphere MQ Telemetry SDK. Individuare la cartella SDK`\clients\java` e selezionare tutti i `.jar` file> **> Apri> Fine**.
2. Installare il Javadoc client MQTT.

Con il Javadoc del client MQTT installato, l'editor Java fornisce assistenza con le classi MQTT v3 .

a) Nel progetto Java, aprire **Esplora package> Librerie di riferimento**. Fare clic con il pulsante destro del mouse `com.ibm.micro.client.mqttv3.jar` **> Proprietà**.

b) Nel navigator Proprietà, fare clic su **Ubicazione Javadoc**.

c) Nella pagina Ubicazione Javadoc, fare clic su **URL Javadoc> Sfoglia** e trovare la cartella *WMQ Installation directory*`\mqxr\SDK\clients\java\doc\javadoc` **> OK**.

d) Fare clic su **Convalida ...> OK**

Viene richiesto di aprire un browser per visualizzare la documentazione.

3. Creare la classe, `PubSync`, utilizzando la procedura guidata Classe Java.

a) Fare clic con il tasto destro del mouse sul progetto Java creato> **Nuovo** > **Classe**.

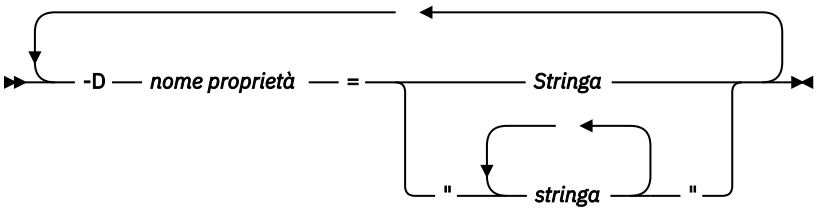
b) Immettere il nome del pacchetto, `com.ibm.mq.id`

c) Immettere il nome classe, `PubSync`

d) Selezionare la casella stub del metodo, **public static void main (String [] args)**

4. Creare un file, `Example.java` nel pacchetto `com.ibm.mq.id`. Copiare il codice in Figura 89 a pagina 483 nel file.

Tutti i parametri utilizzati negli esempi sono impostati come proprietà. È possibile sovrascrivere i valori modificando i valori predefiniti in `Example.javao` fornendo le proprietà come opzioni sulla riga comandi Java utilizzando il parametro `-D` :



L'identificativo client utilizzato in questo esempio e gli esempi “Creazione di un publisher asincrono per MQ Telemetry Transport utilizzando Java” a pagina 483 , è un nome utente con suffisso di una stringa casuale.

5. Seguire la procedura per creare il codice oppure copiare il codice da Figura 88 a pagina 482.

La procedura che segue spiega il codice in `Pubsync.java`.

6. Creare un blocco `try-catch` .

```
try { ...
 } catch (Exception e) {
  e.printStackTrace();
}
```

Il client MQTT genera `MqttException`, `MqttPersistenceException` o `MqttSecurityException`. `MqttPersistenceException` e `MqttSecurityException` sono sottoclassi di `MqttException`.

Utilizzare il metodo `MqttException.getReasonCode` per individuare il motivo dell'eccezione. Se viene generato un `MqttPersistenceException` o un `MqttSecurityException` , utilizzare il metodo `getCause` per restituire l'eccezione throwable sottostante.

7. Creare una nuova istanza `MqttClient` .

```
MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
```

Fornire al cliente un indirizzo server, utilizzato successivamente per connettersi a WebSphere MQ. Impostare l'identificativo client per denominare il client.

- Facoltativamente, è possibile fornire un'implementazione dell'interfaccia `MqttClientPersistence` per sostituire l'implementazione predefinita. L'implementazione

`MqttPersistence` predefinita memorizza QoS 1 e 2 messaggi in attesa di consegna come file; consultare “Persistenza del messaggio nei client MQTT” a pagina 536.

- La porta TCP/IP IBM WebSphere MQ predefinita per MQTT è 1883. Per SSL, è 8883. Nell'esempio, l'indirizzo predefinito è impostato su `tcp://localhost:1883`.
- In genere, è importante poter identificare un client fisico specifico utilizzando l'identificativo client. L'identificativo client deve essere univoco tra tutti i client che si collegano a un server; consultare “Identificativo client” a pagina 532. L'uso dello stesso identificativo client di un'istanza precedente indica che l'istanza presente è un'istanza dello stesso client. Se si duplica un identificativo client in due client in esecuzione, viene generata un'eccezione in entrambi i client e un client viene terminato.
- La lunghezza dell'identificativo client è limitata a 23 byte. Se la lunghezza viene superata, viene generata un'eccezione. L'ID client deve contenere solo i caratteri consentiti in un nome gestore code; ad esempio, senza trattini o spazi.
- Finché non si richiama il metodo `MqttClient.connect` , non viene eseguita alcuna elaborazione del messaggio.

Utilizzare l'oggetto client per pubblicare e sottoscrivere argomenti e recuperare informazioni sulle pubblicazioni che non sono state ancora consegnate.

8. Creare un argomento su cui eseguire la pubblicazione.

```
MqttTopic topic = client.getTopic(Example.topicString);
```

Una stringa argomento è limitata a 64 K byte, che supera la lunghezza massima di una stringa argomento IBM WebSphere MQ . In caso contrario, una stringa di argomenti segue le stesse regole delle stringhe di argomenti WebSphere MQ ; consultare Stringhe di argomenti. L'esempio imposta la stringa di argomenti `MQTT Examples`.

9. Creare un messaggio di pubblicazione.

```
MqttMessage message = new MqttMessage(Example.publication.getBytes());
```

La stringa `"Hello World"` viene convertita in un array di byte e utilizzata per creare un `MqttMessage`.

- Un payload del messaggio MQTT è sempre un array byte. Il metodo `getBytes` converte un oggetto stringa in UTF-8. Il metodo `MqttMessage` ha una convenienza `toString` per restituire il payload del messaggio come stringa. Equivale a `new string(message.getPayload)`
- Un messaggio di pubblicazione viene inviato al gestore code con un'intestazione RFH2 e i dati del messaggio vengono inviati come messaggio `jms-bytes` .
- L'oggetto del messaggio ha QoS (quality of service) e attributi conservati. La qualità del servizio (QoS) determina l'affidabilità del trasferimento del messaggio tra il client MQTT e il gestore code; consultare “Qualità del servizio fornito da un client MQTT” a pagina 541. L'attributo conservato controlla se una pubblicazione viene memorizzata dal gestore code per i sottoscrittori futuri. Se una pubblicazione non viene conservata, viene inviata solo ai sottoscrittori correnti; consultare “Pubblicazioni conservate e client MQTT” a pagina 543. Le impostazioni `MqttMessage` predefinite sono "I messaggi vengono consegnati almeno una volta e non vengono conservati" .

10. Collegarsi al server.

```
client.connect();
```

L'esempio si connette al server utilizzando le opzioni di connessione predefinite. Una volta connessi, è possibile avviare la pubblicazione. Le opzioni di connessione predefinite sono:

- Un piccolo messaggio "keep-alive" viene inviato ogni 15 secondi per evitare che la connessione TCP/IP venga chiusa.
- La sessione viene avviata senza verificare il completamento delle pubblicazioni precedenti.
- L'intervallo tra i tentativi di invio di un messaggio è di 15 secondi.

- Non viene creato alcun messaggio di testamento e di ultima volontà per la connessione.
- Il SocketFactory standard viene utilizzato per creare la connessione.

Modificare le opzioni di connessione creando un oggetto `ConnectionOptions` e trasmettendolo come parametro aggiuntivo a `client.connect`.

11. Pubblica.

```
MqttDeliveryToken token = topic.publish(message);
```

L'esempio invia la pubblicazione "Hello World" sull'argomento "Esempi MQTT" al gestore code.

- Quando viene restituito il metodo `publish` , il messaggio viene trasferito in modo sicuro sul client MQTT, ma non ancora trasferito sul server. Se il messaggio dispone di QoS 1 o 2, il messaggio viene memorizzato localmente, nel caso in cui il client non riesca prima del completamento della consegna.
- `publish` restituisce un token di consegna, che viene utilizzato per controllare se è stato ancora ricevuto un riconoscimento dal server.

12. Attendere il riconoscimento dal server.

```
token.waitForCompletion(Example.timeout);
```

L'esempio PubSync attende un riconoscimento dal server, che conferma che il messaggio è stato consegnato.

- Senza il timeout, il client attenderebbe per un periodo di tempo indefinito. L'attività “Creazione di un publisher asincrono per MQ Telemetry Transport utilizzando Java” a pagina 483 mostra come ricevere i riconoscimenti senza attendere utilizzando un oggetto di callback.

13. Scollegare il client dal server.

```
client.disconnect();
```

Il client si disconnette dal server e attende il completamento di tutti i metodi `MqttCallback` in esecuzione. Attende quindi fino a 30 secondi per terminare il lavoro rimanente. È possibile specificare un timeout di inattività come parametro aggiuntivo.

14. Salvare le modifiche in `PubSync.java` e `Example.Java`

Eclipse compila automaticamente Java. Ora si è pronti a vedere i risultati eseguendo il programma.

Risultati

Per visualizzare le pubblicazioni utilizzando WebSphere MQ, creare un argomento, una coda e una sottoscrizione durevole, tutti denominati `"MQTTExampleTopic"` utilizzando lo script in Figura 87 a pagina 481. Eseguire il client per la pubblicazione sull'argomento `MQTT Examples` , quindi eseguire il programma di esempio **amqsbcg** per ricercare le pubblicazioni nella coda `MQTTExamples` .

1. Avviare un gestore code e avviare il suo servizio di telemetria (MQXR) in esecuzione. Assicurarsi che l'indirizzo TCP/IP e la porta configurati per il canale di telemetria corrispondano ai valori utilizzati nell'applicazione MQTT.
2. Configurare una sottoscrizione durevole creando lo script di comandi `mqttexamples.txt` ed eseguendolo utilizzando **runmqsc:**

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Figura 87. `mqttExampleTopic.txt`

Per eseguire lo script su Windows, immettere il comando:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Eseguire il client come applicazione Java da Eclipseo eseguendo Java in una finestra comandi:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Nota: La finestra dei comandi deve essere aperta nella directory contenente il percorso, `com\ibm\mq\id`.

4. Ricercare i risultati utilizzando WebSphere MQ Explorer oppure eseguire il comando:

```
amqsbcg MQTTExampleQueue queue manager name
```

Codice di esempio

PubSync.java è un elenco completo del codice descritto in Procedura. Modificare la classe `Example` in Figura 89 a pagina 483 per sostituire i parametri predefiniti utilizzati in 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();
    }
  }
}
```

Figura 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;
  }
}
```

Figura 89. Example.java

Concetti correlati

Applicazioni di pubblicazione / sottoscrizione MQTT

Creazione di un publisher asincrono per MQ Telemetry Transport utilizzando Java

In questa attività, si segue un'esercitazione per modificare la prima applicazione del publisher. Le modifiche consentono all'applicazione di inviare pubblicazioni senza attendere le conferme di consegna. I riconoscimenti di consegna vengono ricevuti da una classe callback creata dall'utente.

Prima di iniziare

Installa la funzione WebSphere MQ Telemetry su un server su cui è installato IBM WebSphere MQ Version 7.1 o versioni successive.

L'applicazione client utilizza il pacchetto `com.ibm.mq.micro.client.mqttv3` in IBM WebSphere MQ Telemetry SDK (Software Development Toolkit). L'SDK fa parte dell'installazione di IBM WebSphere MQ Telemetry . Il client si connette alla funzione IBM WebSphere MQ Telemetry per scambiare messaggi con IBM WebSphere MQ.

È inoltre necessario installare gli aggiornamenti di telemetria per IBM WebSphere MQ Explorer Version 7.1 per amministrare IBM WebSphere MQ Telemetry. Gli aggiornamenti fanno parte dell'installazione di IBM WebSphere MQ Telemetry .

Un client MQTT, in esecuzione su Java SE, richiede la versione 6.0 di Java SE o successiva. IBM Java SE v6.0 fa parte dell'installazione di IBM WebSphere MQ Version 7.1 . Si trova in *WebSphere MQ installation directory*\java\jre

Informazioni su questa attività

L'esempio è un'applicazione di pubblicazione, PubAsync. PubAsync pubblica `Hello World` sull'argomento `MQTT Examples`, senza attendere la conferma che la pubblicazione è stata consegnata al gestore code. I riconoscimenti di consegna vengono ricevuti in una classe di callback, `CallBack`.

Impostando una sottoscrizione durevole a `MQTT Examples` , è possibile verificare che l'applicazione funzioni.

La procedura utilizza Eclipse per sviluppare, creare ed eseguire il client. È possibile scaricare Eclipse dal sito Web del progetto Eclipse all'indirizzo www.eclipse.org.

I passaggi in Procedura modificano l'applicazione PubSync`.java` in “Creazione della tua prima applicazione publisher di MQ Telemetry Transport utilizzando Java” a pagina 478.

In alternativa, è possibile copiare il codice “Codice di esempio” a pagina 486in una nuova directory `.\com\ibm\mq\id`. Creare tre file Java, `Example.java`, `CallBack.javae` PubAsync`.java`. Compilare gli esempi utilizzando il comando,

```
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
```

Eseguire PubAsync utilizzando il comando,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubAsync.class
```

Procedura

1. Nel pacchetto `com.ibm.mq.id` , crea un file, `CallBack.java`. Copiare il codice in Figura 92 a pagina 487 nel file.

   `Callback.java` implementa l'interfaccia `MqttCallBack` . Nell'esempio, un costruttore aggiuntivo inizializza il callback con alcuni dati di istanza.
2. Nel `com.ibm.mq.id` package, fare clic con il tasto destro del mouse su PubSync`.java` e copiarlo. Incollarlo nello stesso pacchetto, ridenominandolo PubAsync.
3. Prima della riga di codice `client.connect();` , creare un'istanza della classe `CallBack` , passando l'identificativo client.

   ```
   CallBack callback = new CallBack(Example.clientId);
   client.setCallback(callback);
   ```

   - La classe `CallBack` implementa `MqttCallBack`. È richiesta un'istanza di callback, per identificativo client. In questo esempio, il costruttore passa l'identificativo client da salvare come dati di istanza. Viene utilizzato nel callback per identificare quale istanza del callback è stata avviata.
   - È necessario implementare tre metodi nella classe callback:

     `public void messageArrived(MqttTopic topic, MqttMessage message)`
     Riceve una pubblicazione a cui è stata effettuata la sottoscrizione.

     `public void connectionLost(Throwable cause)`
     Richiamato quando la connessione viene persa.

public void deliveryComplete(MqttDeliveryToken token))
Richiamato quando un token di consegna viene ricevuto per un messaggio QoS 1 o 2 che è stato pubblicato.

- Il callback è attivato da `MqttClient.connect`.

4. Disconnetti client
   a) Rimuovere l'istruzione contenente l'espressione `token.waitForCompletion` .
   Il thread principale continua senza attendere la consegna della pubblicazione.
   b) Verificare se il client è già disconnesso.
   Il client MQTT si disconnette a seguito di un errore restituito al metodo `lostConnection` in `MqttCallback`oppure l'applicazione client potrebbe disconnettersi. Verificare se è presente una connessione aperta.
   c) Utilizzare la costante, `Example.quiesceTimeout`, per impostare il tempo massimo per sospendere il client.

```
if (client.isConnected())
    client.disconnect(Example.quiesceTimeout);
```

Il client termina quando si verifica una combinazione delle seguenti tre condizioni:

a. Il callback è stato richiamato per tutti i messaggi che sono stati pubblicati in questa sessione o, se la sessione è stata riavviata, nelle sessioni precedenti.
b. I messaggi sono in fase di elaborazione e l'intervallo di sospensione è scaduto. Per impostazione predefinita, l'intervallo di sospensione è 30 secondi. È possibile modificare il timeout di sospensione passando il numero di millesimi di secondo di attesa come parametro di `client.disconnect`.
c. `client.disconnect` è stato richiamato dopo la pubblicazione e l'accodamento di alcuni messaggi da parte del client, ma prima dell'invio dei messaggi. I messaggi accodati non sono ancora in corso. Se la sessione è riavviabile, i messaggi vengono reinviati al riavvio della sessione.

Risultati

Per visualizzare le pubblicazioni utilizzando WebSphere MQ, creare un argomento, una coda e una sottoscrizione durevole, tutti denominati "`MQTTExampleTopic`" utilizzando lo script in Figura 90 a pagina 485. Eseguire il client per la pubblicazione sull'argomento `MQTT Examples` , quindi eseguire il programma di esempio **amqsbcg** per ricercare le pubblicazioni nella coda `MQTTExamples` .

1. Avviare un gestore code e avviare il suo servizio di telemetria (MQXR) in esecuzione. Assicurarsi che l'indirizzo TCP/IP e la porta configurati per il canale di telemetria corrispondano ai valori utilizzati nell'applicazione MQTT.
2. Configurare una sottoscrizione durevole creando lo script di comandi `mqttexamples.txt` ed eseguendolo utilizzando **runmqsc:**

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Figura 90. `mqttExampleTopic.txt`

Per eseguire lo script su Windows, immettere il comando:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Eseguire il client come applicazione Java da Eclipseo eseguendo Java in una finestra comandi:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Nota: La finestra dei comandi deve essere aperta nella directory contenente il percorso, `com\ibm\mq\id`.

4. Ricercare i risultati utilizzando WebSphere MQ Explorer oppure eseguire il comando:

```
amqsbcg MQTTExampleQueue queue manager name
```

Codice di esempio

```
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();
    }
  }
}
```

Figura 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();
    }
  }
}
```

Figura 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;
  }
}
```

Figura 93. Example.java

Creazione di un publisher asincrono recuperabile per MQ Telemetry Transport mediante Java

In questa attività, si segue un'esercitazione per modificare l'applicazione del publisher asincrono. Le modifiche consentono all'applicazione di completare la consegna di pubblicazioni non riconosciute l'ultima volta che il client è stato eseguito.

Prima di iniziare

Installa la funzione WebSphere MQ Telemetry su un server su cui è installato IBM WebSphere MQ Version 7.1 o versioni successive.

L'applicazione client utilizza il pacchetto `com.ibm.mq.micro.client.mqttv3` in IBM WebSphere MQ Telemetry SDK (Software Development Toolkit). L'SDK fa parte dell'installazione di IBM WebSphere MQ Telemetry . Il client si connette alla funzione IBM WebSphere MQ Telemetry per scambiare messaggi con IBM WebSphere MQ.

È inoltre necessario installare gli aggiornamenti di telemetria per IBM WebSphere MQ Explorer Version 7.1 per amministrare IBM WebSphere MQ Telemetry. Gli aggiornamenti fanno parte dell'installazione di IBM WebSphere MQ Telemetry .

Un client MQTT, in esecuzione su Java SE, richiede la versione 6.0 di Java SE o successiva. IBM Java SE v6.0 fa parte dell'installazione di IBM WebSphere MQ Version 7.1 . Si trova in *WebSphere MQ installation directory*\java\jre

Informazioni su questa attività

L'esempio è un'applicazione di pubblicazione, PubAsyncRestartable. PubAsyncRestartable pubblica Hello World sull'argomento MQTT Examples, senza attendere la conferma che la pubblicazione è stata consegnata al gestore code. I riconoscimenti di consegna vengono ricevuti in una classe di callback, CallBack. È possibile esaminare qualsiasi token di consegna per le pubblicazioni che non sono state completate in un'istanza precedente. Vengono elaborati anche dalla classe callback.

Impostando una sottoscrizione durevole a MQTT Examples , è possibile verificare che l'applicazione funzioni.

La procedura utilizza Eclipse per sviluppare, creare ed eseguire il client. È possibile scaricare Eclipse dal sito Web del progetto Eclipse all'indirizzo www.eclipse.org.

I passaggi in Procedura modificano l'applicazione PubAsync.java in “Creazione di un publisher asincrono per MQ Telemetry Transport utilizzando Java” a pagina 483.

In alternativa, è possibile copiare il codice “Codice di esempio” a pagina 492in una nuova directory .\com\ibm\mq\id. Creare tre file Java, Example.java, CallBack.javae PubAsyncRestartable.java. Compilare gli esempi utilizzando il comando,

```
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
```

Eseguire PubAsyncRestartable utilizzando il comando,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubAsyncRestartable.class
```

Procedura

1. Nel com.ibm.mq.id package, fare clic con il tasto destro del mouse su PubAsync.java e copiarlo. Incollarlo nello stesso pacchetto, ridenominandolo PubAsyncRestartable.
2. Creare un identificativo client riutilizzabile.

   ```
   Example.clientId = String.format(
       "%-23.23s",
       (System.getProperty("user.name") + "_" + (System.getProperty(
           "clientId", "PubAsyncRestartable."))).trim()).replace('-', '_');
   ```

   Figura 94. Identificativo client riutilizzabile

   Le applicazioni in “Creazione della tua prima applicazione publisher di MQ Telemetry Transport utilizzando Java” a pagina 478 e “Creazione di un publisher asincrono per MQ Telemetry Transport utilizzando Java” a pagina 483 hanno utilizzato un nuovo identificativo client per ogni connessione client. Per un publisher o sottoscrittore riavviabile, è necessario utilizzare lo stesso identificativo client ogni volta che il client è connesso, ma client differenti devono utilizzare identificativi differenti; consultare “Identificativo client” a pagina 532. L'identificativo client riutilizzabile viene creato dal nome utente e dal nome della classe. È limitato a 23 byte di lunghezza. Deve contenere solo caratteri validi nei nomi oggetto del gestore code. Il codice rimuove i trattini che potrebbero essere stati inseriti.
3. Il QoS del messaggio è impostato su 2 anziché sul valore predefinito, 1, per evitare messaggi duplicati.

   ```
   message.setQos(Example.QoS);
   ```

È necessario modificare il valore di `Example.QoS` in 2oppure passare la proprietà `QoS` come argomento utilizzando l'opzione `-DQoS=2` sulla riga comandi Java.

4. Creare un oggetto `MqttConnectOptions` e impostare il suo attributo `cleanSession` su `false`.
   a) Creare un oggetto `MqttConnectOptions` .

   ```
   MqttConnectOptions conOptions = new MqttConnectOptions();
   ```

   `conOptions` è un parametro di opzione sul costruttore `MqttClient` .
   b) Impostare l'attributo clearSession .

   ```
   conOptions.setCleanSession(Example.cleanSession);
   ```

   Per impostazione predefinita, il parametro `Example.cleanSession` è impostato su `true`, che corrisponde all'impostazione predefinita di `MqttConnectionOptions.cleanSession`.

   Quando `PubAsyncRestartable` viene riavviato, può iniziare con una "sessione pulita" e cancellare qualsiasi token di consegna in sospeso per i messaggi di QoS 1 o 2.

   Impostare `Example.cleanSession` su `false` per conservare tutti i token di consegna in sospeso. I token vengono elaborati dalla classe `MqttCallBack` quando il client viene nuovamente connesso.
5. Se la sessione viene riavviata, richiamare i token di consegna in sospeso e stamparne il contenuto.

```
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. Passare il parametro `conOptions` al costruttore `MqttClient` .

```
client.connect(conOptions);
```

7. In fase di disconnessione, impostare un intervallo di disconnessione massimo.

```
client.disconnect(Example.timeout);
```

Per poter mostrare i token di consegna in sospeso in fase di elaborazione, un'istanza precedente deve terminare senza completare la consegna. Per eseguire l'esempio con la possibilità di non riconoscere le pubblicazioni prima del termine di `PubAsyncRestartable` , impostare `Example.timeout` su 0.

Risultati

Per visualizzare le pubblicazioni utilizzando WebSphere MQ, creare un argomento, una coda e una sottoscrizione durevole, tutti denominati `"MQTTExampleTopic"` utilizzando lo script in Figura 95 a pagina 491. Eseguire il client per la pubblicazione sull'argomento `MQTT Examples` , quindi eseguire il programma di esempio **amqsbcg** per ricercare le pubblicazioni nella coda `MQTTExamples` .

1. Avviare un gestore code e avviare il suo servizio di telemetria (MQXR) in esecuzione. Assicurarsi che l'indirizzo TCP/IP e la porta configurati per il canale di telemetria corrispondano ai valori utilizzati nell'applicazione MQTT.
2. Configurare una sottoscrizione durevole creando lo script di comandi `mqttexamples.txt` ed eseguendolo utilizzando **runmqsc:**

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Figura 95. mqttExampleTopic.txt

Per eseguire lo script su Windows, immettere il comando:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Eseguire il client come applicazione Java da Eclipseo eseguendo Java in una finestra comandi:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Nota: La finestra dei comandi deve essere aperta nella directory contenente il percorso, `com\ibm\mq\id`.

4. Ricercare i risultati utilizzando WebSphere MQ Explorer oppure eseguire il comando:

```
amqsbcg MQTTExampleQueue queue manager name
```

Codice di esempio

```
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();
    }
  }
}
```

Figura 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();
    }
  }
}
```

Figura 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;
  }
}
```

Figura 98. Example.java

Creazione di un sottoscrittore per MQ Telemetry Transport utilizzando Java

In questa attività, si segue un'esercitazione per creare un'applicazione del sottoscrittore. Il sottoscrittore crea una sottoscrizione a un argomento e riceve le pubblicazioni per la sottoscrizione.

Prima di iniziare

Installa la funzione WebSphere MQ Telemetry su un server su cui è installato IBM WebSphere MQ Version 7.1 o versioni successive.

L'applicazione client utilizza il pacchetto `com.ibm.mq.micro.client.mqttv3` in IBM WebSphere MQ Telemetry SDK (Software Development Toolkit). L'SDK fa parte dell'installazione di IBM WebSphere MQ Telemetry . Il client si connette alla funzione IBM WebSphere MQ Telemetry per scambiare messaggi con IBM WebSphere MQ.

È inoltre necessario installare gli aggiornamenti di telemetria per IBM WebSphere MQ Explorer Version 7.1 per amministrare IBM WebSphere MQ Telemetry. Gli aggiornamenti fanno parte dell'installazione di IBM WebSphere MQ Telemetry .

Un client MQTT, in esecuzione su Java SE, richiede la versione 6.0 di Java SE o successiva. IBM Java SE v6.0 fa parte dell'installazione di IBM WebSphere MQ Version 7.1 . Si trova in *WebSphere MQ installation directory*\java\jre

Informazioni su questa attività

L'esempio è un'applicazione sottoscrittore, Subscribe. Subscribe crea un argomento di sottoscrizione, MQTT Examplese attende le pubblicazioni sulla sottoscrizione per 30 secondi.

Un sottoscrittore può creare una sottoscrizione e attendere le pubblicazioni. Può anche ricevere pubblicazioni inviate a una sottoscrizione creata precedentemente, per lo stesso identificativo client. L'attributo booleano MqttConnectionOptions.cleanSession controlla se le pubblicazioni inviate in precedenza vengono ricevute o meno; consultare "Sottoscrizione" a pagina 544.

È possibile utilizzare i programmi di esempio di pubblicazione per creare pubblicazioni oppure utilizzare WebSphere MQ explorer per creare una pubblicazione di verifica sull'argomento MQTT Examples .

La procedura utilizza Eclipse per sviluppare, creare ed eseguire il client. È possibile scaricare Eclipse dal sito Web del progetto Eclipse all'indirizzo www.eclipse.org.

Le istruzioni contenute in Procedura presumono che il pacchetto com.ibm.mq.id sia già stato creato in una delle precedenti attività e copiato nelle classi Example.java e Callback.java .

Procedura

1. Creare la classe, Subscribe nel package com.ibm.mq.id.
2. Creare un identificativo client riutilizzabile.

```
Example.clientId = String.format(
    "%-23.23s",
    (System.getProperty("user.name") + "_" + (System.getProperty(
        "clientId", "Subscribe."))).trim()).replace('-', '_');
```

Figura 99. Identificativo client riutilizzabile

Le applicazioni in "Creazione della tua prima applicazione publisher di MQ Telemetry Transport utilizzando Java" a pagina 478 e "Creazione di un publisher asincrono per MQ Telemetry Transport utilizzando Java" a pagina 483 hanno utilizzato un nuovo identificativo client per ogni connessione client. Per un publisher o sottoscrittore riavviabile, è necessario utilizzare lo stesso identificativo client ogni volta che il client è connesso, ma client differenti devono utilizzare identificativi differenti; consultare "Identificativo client" a pagina 532. L'identificativo client riutilizzabile viene creato dal nome utente e dal nome della classe. È limitato a 23 byte di lunghezza. Deve contenere solo caratteri validi nei nomi oggetto del gestore code. Il codice rimuove i trattini che potrebbero essere stati inseriti.

3. Creare un blocco try-catch .

```
try { ...
 } catch (Exception e) {
  e.printStackTrace();
}
```

Il client MQTT genera MqttException, MqttPersistenceException o MqttSecurityException. MqttPersistenceException e MqttSecurityException sono sottoclassi di MqttException.

Utilizzare il metodo MqttException.getReasonCode per individuare il motivo dell'eccezione. Se viene generato un MqttPersistenceException o un MqttSecurityException , utilizzare il metodo getCause per restituire l'eccezione throwable sottostante.

4. Creare una nuova istanza MqttClient .

```
MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
```

Fornire al cliente un indirizzo server, utilizzato successivamente per connettersi a WebSphere MQ. Impostare l'identificativo client per denominare il client.

- Facoltativamente, è possibile fornire un'implementazione dell'interfaccia `MqttClientPersistence` per sostituire l'implementazione predefinita. L'implementazione `MqttPersistence` predefinita memorizza QoS 1 e 2 messaggi in attesa di consegna come file; consultare “Persistenza del messaggio nei client MQTT” a pagina 536.
- La porta TCP/IP IBM WebSphere MQ predefinita per MQTT è 1883. Per SSL, è 8883. Nell'esempio, l'indirizzo predefinito è impostato su `tcp://localhost:1883`.
- In genere, è importante poter identificare un client fisico specifico utilizzando l'identificativo client. L'identificativo client deve essere univoco tra tutti i client che si collegano a un server; consultare “Identificativo client” a pagina 532. L'uso dello stesso identificativo client di un'istanza precedente indica che l'istanza presente è un'istanza dello stesso client. Se si duplica un identificativo client in due client in esecuzione, viene generata un'eccezione in entrambi i client e un client viene terminato.
- La lunghezza dell'identificativo client è limitata a 23 byte. Se la lunghezza viene superata, viene generata un'eccezione. L'ID client deve contenere solo i caratteri consentiti in un nome gestore code; ad esempio, senza trattini o spazi.
- Finché non si richiama il metodo `MqttClient.connect` , non viene eseguita alcuna elaborazione del messaggio.

Utilizzare l'oggetto client per pubblicare e sottoscrivere argomenti e recuperare informazioni sulle pubblicazioni che non sono state ancora consegnate.

5. Prima della riga di codice `client.connect();` , creare un'istanza della classe `CallBack` , passando l'identificativo client.

```
CallBack callback = new CallBack(Example.clientId);
client.setCallback(callback);
```

- La classe `CallBack` implementa `MqttCallBack`. È richiesta un'istanza di callback, per identificativo client. In questo esempio, il costruttore passa l'identificativo client da salvare come dati di istanza. Viene utilizzato nel callback per identificare quale istanza del callback è stata avviata.
- È necessario implementare tre metodi nella classe callback:

  `public void messageArrived(MqttTopic topic, MqttMessage message)`
  Riceve una pubblicazione a cui è stata effettuata la sottoscrizione.

  `public void connectionLost(Throwable cause)`
  Richiamato quando la connessione viene persa.

  `public void deliveryComplete(MqttDeliveryToken token))`
  Richiamato quando un token di consegna viene ricevuto per un messaggio QoS 1 o 2 che è stato pubblicato.
- Il callback è attivato da `MqttClient.connect`.

6. Creare un oggetto `MqttConnectOptions` e impostare l'attributo `cleanSession` .
   a) Creare un oggetto `MqttConnectOptions` .

   ```
   MqttConnectOptions conOptions = new MqttConnectOptions();
   ```

   `conOptions` è un parametro di opzione sul costruttore `MqttClient` .
   b) Impostare l'attributo clearSession .

   ```
   conOptions.setCleanSession(Example.cleanSession);
   ```

   Per impostazione predefinita, il parametro `Example.cleanSession` è impostato su `true`, che corrisponde all'impostazione predefinita di `MqttConnectionOptions.cleanSession`.

Se si utilizza il valore predefinito `MqttConnectOptions` o si imposta `MqttConnectOptions.cleanSession` su `true` prima di connettere il client, tutte le sottoscrizioni precedenti del client vengono rimosse alla connessione del client. Qualsiasi nuova sottoscrizione effettuata dal client durante la sessione viene rimossa quando si disconnette.

Se si imposta `MqttConnectOptions.cleanSession` su `false` prima della connessione, qualsiasi sottoscrizione creata dal client viene aggiunta a tutte le sottoscrizioni esistenti per il client prima della sua connessione. Tutte le sottoscrizioni restano attive alla disconnessione del client.

Un altro modo per comprendere come l'attributo `cleanSession` influisce sulle sottoscrizioni è considerarlo come un attributo modale. Nella sua modalità predefinita, `cleanSession=true`, il client crea sottoscrizioni e riceve pubblicazioni solo nell'ambito della sessione. Nella modalità alternativa, `cleanSession=false`, le sottoscrizioni sono durevoli. Il client può connettersi e disconnettersi e le sue sottoscrizioni rimangono attive. Quando il client si riconnette, riceve tutte le pubblicazioni non consegnate. Mentre è connesso, può modificare l'insieme di sottoscrizioni attive per suo conto.

È necessario impostare la modalità `cleanSession` prima della connessione; la modalità è valida per l'intera sessione. Per modificarne l'impostazione, è necessario disconnettere e riconnettere il client. Se si modifica la modalità da `cleanSession=false` a `cleanSession=true`, tutte le sottoscrizioni precedenti destinate al client e le eventuali pubblicazioni non ricevute vengono eliminate.

7. Passare il parametro `conOptions` al costruttore `MqttClient` .

```
client.connect(conOptions);
```

8. Creare una sottoscrizione.

```
      client.subscribe(Example.topicString, Example.QoS);
```

L'esempio utilizza un metodo `MqttClient.subscribe` che passa un filtro argomento con un'opzione `QoS` . Il metodo `MqttClient.subscribe` ha quattro firme ed è possibile passare array di filtri di sottoscrizione e un singolo filtro.

L'esempio utilizza la stringa argomento utilizzata dagli esempi di pubblicazione come filtro argomento, in modo che riceva tutte le pubblicazioni che creano.

Ogni volta che si esegue l'esempio, `subscribe.java`, viene creata una sottoscrizione. Se non si modifica Example.topicString, si ricrea nuovamente la stessa sottoscrizione. Se una sottoscrizione viene ricreata, non risulta in due sottoscrizioni identiche. Un client non riceve copie duplicate di pubblicazioni che corrispondono a una sottoscrizione identica.

Le sottoscrizioni sono descritte in “Sottoscrizione” a pagina 544e i filtri in “Stringhe argomento e filtri argomento nei client MQTT” a pagina 546.

9. Attendere l'arrivo di alcune pubblicazioni e scollegare il client.

```
      Thread.sleep(Example.sleepTimeout);
      client.disconnect();
```

Le pubblicazioni vengono ricevute dall'implementazione del metodo `MqttCallback.messageArrived` .

L'applicazione di sottoscrizione non ha pubblicato alcun messaggio e quindi non attende alcun token di consegna. `client.disconnect` si svolge senza alcun ritardo.

Codice di esempio

```
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();
    }
  }
}
```

Figura 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();
    }
  }
}
```

Figura 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;
  }
}
```

Figura 102. Example.java

Concetti correlati

Applicazioni di pubblicazione / sottoscrizione MQTT

Autenticazione di un client MQTT Java mediante JAAS

Informazioni su come autenticare un client utilizzando JAAS. Modificare il programma di esempio `JAASLoginModule.java` e il programma Java di esempio `PubSync.java`. Configurare un canale di telemetria per richiedere l'autenticazione JAAS ed eseguire il publisher modificato, verificandone nome `utente` e `password` utilizzando JAAS.

Prima di iniziare

Si presume che siano stati installati i file jar del client MQTT v3 , Javadoc Eclipse, i canali di telemetria configurati e codificati ed eseguire PubSync.java prima di eseguire questa attività. Hai uno spazio di lavoro Eclipse che includa una versione in esecuzione di PubSync.java.

L'attività viene scritta per Windows. Modificare i percorsi di directory per Linux.

Informazioni su questa attività

L'attività si basa sulla modifica della classe JAASLoginModule di esempio in *WMQ Installation directory*\mqxr\samples\JAASLoginModule.java per creare MyLogin.java. Nell'attività, è anche possibile modificare il codice di esempio, PubSync.java in "Creazione della tua prima applicazione publisher di MQ Telemetry Transport utilizzando Java" a pagina 478 per impostare un nome utente e una password. Come test, MyLogin.java accetta o rifiuta in modo casuale username e password.

I passi nell'attività sono scritti come un esercizio di programmazione. È necessario adattare la procedura per eseguire l'autenticazione reale in un ambiente di produzione.

In una tipica spiegazione di come programmare l'autenticazione JAAS , si presume che il modulo login stia autenticando il contesto che ha caricato JAAS. Quando il servizio di telemetria (MQXR) richiama JAAS, il contesto che ha caricato JAAS è il servizio di telemetria (MQXR). Non ha senso autenticare il contesto del servizio di telemetria (MQXR); è sempre mqm. Invece, il servizio di telemetria (MQXR) imposta il nome utente e la password del client in modo che siano disponibili per la classe del modulo di login. Il nome utente e la password vengono passati al modulo di login utilizzando due callback.

```
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();
```

Il nome utente e la password del client sono le uniche informazioni sul client disponibili per il modulo di login.

Procedura

1. Creare due package, samples e security.jaas nello stesso progetto Java di PubSync.java.

   Il package samples viene utilizzato solo come riferimento. Apportare le modifiche al codice nel pacchetto security.jaas .
2. Importare JAASLoginModule.java e JAASPrincipal.java in entrambi i package.

   Se necessario, eseguire il refactoring delle istruzioni del pacchetto nell'origine Java per eliminare gli errori di compilazione.
3. Eseguire il refactoring del nome classe, JAASLoginModule, nel pacchetto security.jaas in MyLogin
4. In MyLogin.java, sostituire parte del codice nel metodo login per mostrare il funzionamento del modulo.
   a) Sostituire il codice:

   ```
   // Accept everything.
   if (true)
     loggedIn = true;
   else
     throw new javax.security.auth.login.FailedLoginException("Login failed");
   ```

   b) Con il codice:

   ```
   // 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);
```

L'origine completa per `MyLogin.java` si trova in Figura 105 a pagina 503. L'origine per `JAASPrincipal.java`, con il nome del pacchetto refactoring in `security.jaas` si trova in Figura 106 a pagina 504.

5. Impostare il percorso classe in `service.env` in modo che punti alla directory contenente il percorso a `security/jaas/MyLogin.class` e `security/jaas/JAASPrincipal.class`.

```
CLASSPATH=C:\WMQTelemetryApps\MQTTSecureExamples\bin
```

Consultare Configurazione del canale di telemetria JAAS per informazioni sull'utilizzo di `service.env` per passare un percorso di classe a un servizio WebSphere MQ .

6. Aggiungere una stanza del modulo di login a `jaas.config`.

```
MyLoginExample {
  security.jaas.MyLogin required debug=true;
  };
```

Consultare Configurazione del canale di telemetria JAAS per informazioni sull'utilizzo `jaas.config` di un modulo di login JAAS .

7. Aggiungere un canale di telemetria utilizzando la procedura guidata **Nuovo canale di telemetria** in WebSphere MQ Explorer, configurando il canale in modo da richiedere l'autenticazione JAAS . Fare riferimento alla stanza `MyLoginExample` .

Ad esempio, adattare le informazioni immesse nella procedura guidata da questa stanza nel file `mqxr_win.properties` . Se si utilizza Linux , il file viene denominato `mqxr_unix.properties`. Non modificare direttamente il file delle proprietà di telemetria; utilizzare la procedura guidata.

```
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
```

Nota: Se si modificano i parametri del canale di telemetria o la classe `security.jaas.Mylogin` , è necessario arrestare e riavviare il servizio di telemetria (MQXR). Solo quando si riavvia il servizio, le modifiche diventano effettive.

8. Creare una copia di `PubSync.java` nel pacchetto `com.ibm.mq.id` e denominare la copia `PubSyncJAAS.java`.

Consulta “Creazione della tua prima applicazione publisher di MQ Telemetry Transport utilizzando Java” a pagina 478 per i passi per creare PubSync.java nel package `com.ibm.mq.id` .

9. Impostare `MqttConnectOptions.username` e `MqttConnectOptions.password` nel programma `PubSyncJAAS.java` e passare `MqttConnectOptions` come parametro di `MqttClient.connect`.

```
MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setUserName(Example.username);
conOptions.setPassword(Example.password);
client.connect(conOptions);
```

Consultare il codice in corsivo in PubSyncJAAS.Java utilizzando le costanti impostate in Example.java.

10. Impostare `Example.TCPAddress` sull'indirizzo socket del canale di telemetria configurato per utilizzare la configurazione JAAS , `MyLoginExample`. Ad esempio, utilizzare 1884 come numero di porta.
11. Eseguire PubSyncJAAS diverse volte per visualizzare il client che si collega e che viene accettato o rifiutato.

Viene generata un'eccezione ogni volta che il tentativo di login viene rifiutato.

Risultati

Figura 103 a pagina 502 mostra il risultato dell'esecuzione di PubSyncJAAS.Java due volte. I record di log vengono mostrati in Figura 104 a pagina 502.

```
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
```

Figura 103. Output della Console da PubSyncJAAS.java

Il file di log `MyLogin.log` è memorizzato in *`WMQ Data directory`*; ad esempio, `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
```

Figura 104. MyLogin.log

Esempi

Il codice in corsivo in Figura 105 a pagina 503 è la modifica all'esempio `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;
  }
}
```

Figura 105. MyLogin.java

Figura 106 a pagina 504 è il codice di esempio JAASLoginPrincipal.java, copiato nel pacchetto security.jaas. Lo scopo di JAASLoginPrincipal è implementare l'interfaccia

`java.security.Principal` per conservare un record degli utenti che hanno eseguito correttamente l'accesso da `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();
  }
}
```

Figura 106. JAASLoginPrincipal.java

Il codice in PubSync.java modificato per aggiungere un nome utente e una `password` è in corsivo in Figura 107 a pagina 504.

```
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();
    }
  }
}
```

Figura 107. PubSyncJAAS.Java

Modificare le costanti in Example.java in modo che corrispondano alla configurazione. Ignorare le impostazioni SSL per questo esempio.

```
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;
  }
}
```

Figura 108. Example.java

Autenticazione di una connessione di telemetria SSL mediante certificati autofirmati

Utilizzare i certificati autofirmati generati utilizzando **Keytool** per autenticare una connessione SSL. È possibile autenticare il canale di telemetria o il canale di telemetria e i client che vi si collegano. I messaggi in transito sulla connessione vengono crittografati.

Prima di iniziare

Eseguire l'attività, “Creazione della tua prima applicazione publisher di MQ Telemetry Transport utilizzando Java” a pagina 478 prima di iniziare, per ottenere PubSync.java che utilizza una connessione TCP/IP non protetta. In questa attività, si modifica PubSync`.java` per utilizzare una connessione SSL.

Informazioni su questa attività

I passi nell'attività sono scritti come un esercizio di programmazione. È necessario adattare la procedura per eseguire l'autenticazione reale in un ambiente di produzione.

L'attività viene scritta per Windows. Modificare i percorsi di directory per Linux.

Procedura

1. Eseguire l'attività, “Modifica di PubSync.java per utilizzare SSL” a pagina 506, per modificare PubSync.java per utilizzare SSL.
2. Configurare il canale di telemetria e creare i keystore per utilizzare SSL.

   Autenticare solo il canale di telemetria o il canale e i client che si collegano ad esso:

   - Eseguire l'attività, “Autenticazione del canale di telemetria” a pagina 507, per connettersi con SSL, autenticando il canale di telemetria.
   - Eseguire l'attività, “Autenticazione del canale di telemetria e dei client” a pagina 508, per connettersi con SSL, autenticando il canale di telemetria e i client che si collegano ad essa.
3. Arrestare e riavviare il servizio di telemetria (MQXR) per acquisire le modifiche alle configurazioni del canale di telemetria.
4. Eseguire il programma client per verificare se la configurazione funziona.

Modifica di PubSync.java per utilizzare SSL

Modificare il primo esempio di programma publisher per connettersi a un canale di telemetria utilizzando SSL. Impostare le proprietà SSL utilizzate dal programma modificato.

Prima di iniziare

Si presume che siano stati installati i file jar del client MQTT v3 , Javadoc Eclipse, i canali di telemetria configurati e codificati ed eseguire PubSync.java prima di eseguire questa attività. Hai uno spazio di lavoro Eclipse che includa una versione in esecuzione di PubSync.java.

Informazioni su questa attività

L'attività utilizza come base il client publisher, PubSync.java, creato in “Creazione della tua prima applicazione publisher di MQ Telemetry Transport utilizzando Java” a pagina 478 . Per utilizzare SSL sono necessarie solo piccole modifiche; consultare Figura 109 a pagina 507 e Figura 110 a pagina 507.

Procedura

1. Creare una copia di `PubSync.java` nel pacchetto `com.ibm.mq.id` e denominare la copia `PubSyncSSL.java`.

   Consulta “Creazione della tua prima applicazione publisher di MQ Telemetry Transport utilizzando Java” a pagina 478 per i passi per creare PubSync.java nel package `com.ibm.mq.id` .
2. Impostare `Example.SSLAddress` sull'indirizzo del socket del canale di telemetria configurato per l'utilizzo per la configurazione SSL.
3. Modificare il parametro dell'indirizzo socket del costruttore client per utilizzare `Example.SSLAddress`.

   ```
   MqttClient client = new MqttClient(Example.SSLAddress, Example.clientId);
   ```
4. Impostare `MqttConnectOptions.SSLProperties` in `PubSyncSSL.javae` passare `MqttConnectOptions` come parametro di `MqttClient.connect`.

   ```
   MqttConnectOptions conOptions = new MqttConnectOptions();
   conOptions.setSSLProperties(Example.getSSLSettings());
   client.connect(conOptions);
   ```

   Consultare il codice in corsivo `PubSyncSSL.Java` utilizzando le costanti impostate in `Example.java`.

Esempi

Le modifiche a PubSync.java per aggiungere SSL sono mostrate in Figura 109 a pagina 507 in corsivo.

```
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();
    }
  }
}
```

Figura 109. PubSyncSSL.Java

Le modifiche a Example.java vengono mostrate in Figura 110 a pagina 507.

```
 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;
```

Figura 110. Modifiche a Example.java

Autenticazione del canale di telemetria

I client autenticano il canale di telemetria per codificare il contenuto dei messaggi che fluiscono sul canale e per garantire che un client si connetta al canale di telemetria corretto. Il server non autentica il client.

Informazioni su questa attività

È possibile utilizzare diversi editor keystore per creare e gestire certificati autofirmati. L'attività utilizza il comando **keytool** della riga comandi, che fa parte di JRE. È possibile utilizzare il tool GUI **iKeyman**, fornito con WebSphere MQ per sfogliare i keystore e generare chiavi. Avviare **iKeyman** utilizzando il comando **strmqikm**.

Procedura

1. Creare un canale di telemetria, SSLSSOptClients che richiede una connessione SSL utilizzando la procedura guidata **Nuovo canale di telemetria** . Il canale accetta client anonimi.

   Adattare la configurazione del canale dalla seguente sezione di configurazione. Non modificare direttamente il file delle proprietà di telemetria; utilizzare la procedura guidata.

   ```
   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. Generare le chiavi per il client per autenticare il canale di telemetria.

   a) Generare una coppia di chiavi autofirmata per il canale di telemetria in un nuovo keystore, 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) Esportare il certificato pubblico come file ASCII, utilizzando l'opzione -rfc :

   ```
   Keytool -export -noprompt -alias SSServerPrivate -file SSServerPublic.cer
           -keystore SSServerOptKey.jks -storepass password -rfc
   ```

   Se si sta eseguendo l'attività su Windows, fare doppio clic su SSServerPublic.cer per esaminarne il contenuto.

   c) Importare il certificato pubblico in un nuovo truststore client, SSClientTrust.jks:

   ```
   Keytool -import -noprompt -alias SSServerPublic -file SSServerPublic.cer
           -keystore SSClientTrust.jks -storepass password
   ```

   d) Creare un keystore client vuoto, SSClientKey.jks.

   Keytool non contiene un comando per creare un keystore vuoto. Sono disponibili due opzioni:

   i) Eseguire **strmqikm**e creare un keystore, SSClientKey.jks, ma non aggiungere alcuna chiave.

   ii) Eseguire il passaggio 3a in “Autenticazione del canale di telemetria e dei client” a pagina 508, ma non utilizzare ancora le chiavi.

Autenticazione del canale di telemetria e dei client

I client autenticano il canale di telemetria e il canale di telemetria autentica i client che vi si collegano. I messaggi in transito sul canale vengono crittografati.

Informazioni su questa attività

È possibile utilizzare diversi editor keystore per creare e gestire certificati autofirmati. L'attività utilizza il comando **keytool** della riga comandi, che fa parte di JRE. È possibile utilizzare il tool GUI **iKeyman**, fornito con WebSphere MQ per sfogliare i keystore e generare chiavi. Avviare **iKeyman** utilizzando il comando **strmqikm**.

Il canale di telemetria è configurato con un keystore differente per l'attività, “Autenticazione del canale di telemetria” a pagina 507. È possibile utilizzare lo stesso keystore e omettere il passo “2” a pagina 509 per aggiungere le chiavi al keystore.

Procedura

1. Creare un canale di telemetria, SSLSSReqClients che richiede una connessione SSL utilizzando la procedura guidata **Nuovo canale di telemetria** . Il canale accetta solo client autenticati.

   Adattare la configurazione del canale dalla seguente sezione di configurazione:

   ```
   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. Generare le chiavi per il client per autenticare il canale di telemetria.

   a) Generare una coppia di chiavi autofirmata per il canale di telemetria in un nuovo keystore, 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) Esportare il certificato pubblico come file ASCII, utilizzando l'opzione -rfc :

   ```
   Keytool -export -noprompt -alias SSServerPrivate -file SSServerPublic.cer
           -keystore SSServerReqKey.jks -storepass password -rfc
   ```

   Se si sta eseguendo l'attività su Windows, fare doppio clic su SSServerPublic.cer per esaminarne il contenuto.

   c) Importare il certificato pubblico in un nuovo truststore client, SSClientTrust.jks:

   ```
   Keytool -import -noprompt -alias SSServerPublic -file SSServerPublic.cer
           -keystore SSClientTrust.jks -storepass password
   ```

3. Generare le chiavi per il canale di telemetria per autenticare un client.

   a) Generare una coppia di chiavi autofirmata per il client in un nuovo keystore, 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) Esportare il certificato pubblico come file ASCII, utilizzando l'opzione -rfc :

   ```
   Keytool -export -noprompt -alias SSClientPrivate -file SSClientPublic.cer
           -keystore SSClientKey.jks -storepass password -rfc
   ```

   Se si sta eseguendo l'attività su Windows, fare doppio clic su SSClientPublic.cer per esaminarne il contenuto.

   c) Importare il certificato pubblico nel keystore del server, SSServerReqKey.jks:

   ```
   Keytool -import -noprompt -alias SSClientPublic -file SSClientPublic.cer
           -keystore SSServerReqKey.jks -storepass password
   ```

   I canali di telemetria utilizzano lo stesso archivio sia per le chiavi private che per i certificati attendibili.

Autenticazione di una connessione di telemetria SSL mediante una catena di certificati

Utilizzare i certificati firmati ottenuti da un'autorità di certificazione o dall'implementazione della propria procedura di certificazione, per autenticare una connessione SSL. È possibile autenticare il canale di

telemetria o il canale di telemetria e i client che vi si collegano. I messaggi in transito sulla connessione vengono crittografati.

Prima di iniziare

Eseguire l'attivit ..., “Autenticazione di una connessione di telemetria SSL mediante certificati autofirmati” a pagina 505 prima di iniziare, per fare in modo che PubSyncSSL . Java utilizzi una connessione TCP/IP protetta utilizzando certificati autofirmati.

Informazioni su questa attività

In questa attività, modificare le attività “Autenticazione del canale di telemetria” a pagina 507e “Autenticazione del canale di telemetria e dei client” a pagina 508 in “Autenticazione di una connessione di telemetria SSL mediante certificati autofirmati” a pagina 505per gestire le chiavi certificate da una catena di certificati.

È possibile ottenere i certificati per questa attività da un'autorità di certificazione oppure è possibile utilizzare un sito Web come http://www.openca.org/ per ottenere i certificati. Le autorità di certificazione commerciali generalmente forniscono gratuitamente i certificati di prova per un breve periodo. Questa attività è stata verificata utilizzando certificati ottenuti commercialmente.

Un'altra opzione è quella di creare il proprio processo di certificazione ed eseguirlo sui propri computer, utilizzando gli strumenti di siti Web come https://www.openssl.org/.

I truststore JRE `cacerts` non vengono utilizzati in questa attività. È possibile utilizzare il truststore JRE `cacerts` sul client nell'attività, “Autenticazione del canale di telemetria” a pagina 510, anziché utilizzare il truststore specificato. La catena di certificati potrebbe essere firmata da un'autorità di certificazione nota che ha già il proprio certificato root nell'archivio `cacerts` sul client. In questo caso, non specificare un truststore sul client. Assicurarsi che, se sul client sono installati più JRE, si gestisca l'archivio `cacerts` corretto.

Procedura

1. Se non è stato ancora fatto, eseguire l'attività, “Modifica di PubSync.java per utilizzare SSL” a pagina 506, per modificare PubSync.java per utilizzare SSL.
2. Configurare il canale di telemetria e creare i keystore per utilizzare SSL.

   Autenticare solo il canale di telemetria o il canale e i client che si collegano ad esso:

   - Eseguire l'attività, “Autenticazione del canale di telemetria” a pagina 510, per connettersi con SSL, autenticando il canale di telemetria.
   - Eseguire l'attività, “Autenticazione del canale di telemetria e dei client” a pagina 512, per connettersi con SSL, autenticando il canale di telemetria e i client che si collegano ad essa.
3. Arrestare e riavviare il servizio di telemetria (MQXR) per acquisire le modifiche alle configurazioni del canale di telemetria.
4. Eseguire il programma client per verificare se la configurazione funziona.

Autenticazione del canale di telemetria

I client autenticano il canale di telemetria per crittografare il contenuto dei messaggi che fluiscono sul canale e per garantire che un client si connetta al canale di telemetria corretto. Il server non autentica il client.

Informazioni su questa attività

È possibile utilizzare diversi editor keystore per creare e gestire certificati. L'attività utilizza il comando **keytool** della riga comandi, che fa parte di JRE. È possibile utilizzare il tool GUI **iKeyman**, fornito con WebSphere MQ per sfogliare i keystore e generare chiavi. Avviare **iKeyman** utilizzando il comando **strmqikm**.

Procedura

1. Creare un canale di telemetria, `SSLCAOptClients` che richiede una connessione SSL utilizzando la procedura guidata **Nuovo canale di telemetria** . Il canale accetta client anonimi.

   Adattare la configurazione del canale dalla seguente sezione di configurazione. Non modificare direttamente il file delle proprietà di telemetria; utilizzare la procedura guidata.

   ```
   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. Generare una chiave firmata dalla CA per il client per autenticare il canale di telemetria.

   a) Generare una coppia di chiavi autofirmata per il canale di telemetria in un nuovo keystore, `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
   ```

   L'algoritmo chiave è impostato su RSA perché alcune autorità di certificazione lo richiedono. Il nome comune del certificato deve essere univoco, alcune autorità di certificazione non emettono chiavi con nomi comuni identici.

   b) Crea una CSR (certificate signing request) come file 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) Eseguire il software dell'autorità di certificazione o accedere al relativo sito Web. Incollare il contenuto di `CAServerOptKey.csr` quando viene richiesto il file CSR.

   d) L'autorità di certificazione restituisce uno o due certificati e un file di risposta firmato come file ASCII. Incollare il contenuto in due o tre file:

   Certificato root
   Incolla in `CARoot.cer`

   Certificato intermedio
   Incolla in `CAInter.cer`

   File di risposte firmato dal server
   Incolla in `CAServerOpt.rsp`

   L'archivio certificati JRE non è utilizzato in questa attività. Se hai ricevuto un certificato root e una risposta firmata dalla CA, utilizza il certificato root e la risposta firmata nei seguenti passaggi. Se hai ricevuto un certificato root e un certificato intermedio, utilizza il certificato intermedio e la risposta firmata.

   e) Ricevere la risposta del server firmato nel keystore del server da cui è stata emessa la richiesta di certificato.

   La ricezione della risposta modifica il certificato autofirmato in modo che sia firmato dalla CA. Se si guarda il certificato nel keystore prima e dopo aver ricevuto la risposta, il firmatario cambia. In caso contrario, viene riportato un errore dallo strumento di gestione delle chiavi. Prima di utilizzare il certificato, controllarlo e verificare che il firmatario sia ora la CA.

   ```
   Keytool -import -noprompt -alias CAServer -file CAServerOpt.rsp
           -keystore CAServerOptKey.jks -storepass password
   ```

   In alcuni software di gestione delle chiavi, come **iKeyman**, si ricevono, anziché importare, i file di risposta.

f) Importare il certificato CA nel truststore client.

Importare il certificato intermedio se sono stati ricevuti due certificati dalla CA o il certificato root, se è stato ricevuto solo un certificato.

Le alternative sono:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAClientTrust.jks -storepass password
```

Oppure:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAClientTrust.jks -storepass password
```

Autenticazione del canale di telemetria e dei client

I client autenticano il canale di telemetria e il canale di telemetria autentica i clienti che vi si collegano. I messaggi in transito sul canale vengono crittografati.

Informazioni su questa attività

È possibile utilizzare diversi editor keystore per creare e gestire certificati. L'attività utilizza il comando **keytool** della riga comandi, che fa parte di JRE. È possibile utilizzare il tool GUI **iKeyman**, fornito con WebSphere MQ per sfogliare i keystore e generare chiavi. Avviare **iKeyman** utilizzando il comando **strmqikm**.

Il canale di telemetria è configurato con un keystore diverso da quello dell'attività, “Autenticazione del canale di telemetria” a pagina 510. È possibile utilizzare lo stesso keystore e omettere il passo “2” a pagina 512 per aggiungere le chiavi al keystore.

Procedura

1. Creare un canale di telemetria, `SSLCAReqClients` che richiede una connessione SSL utilizzando la procedura guidata **Nuovo canale di telemetria** . Il canale accetta solo client autenticati.

   Adattare la configurazione del canale dalla seguente sezione di configurazione. Non modificare direttamente il file delle proprietà di telemetria; utilizzare la procedura guidata.

   ```
   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. Generare una chiave firmata dalla CA per il client per autenticare il canale di telemetria.

   a) Generare una coppia di chiavi autofirmata per il canale di telemetria in un nuovo keystore, `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
   ```

   L'algoritmo chiave è impostato su RSA perché alcune autorità di certificazione lo richiedono. Il nome comune del certificato deve essere univoco, alcune autorità di certificazione non emettono chiavi con nomi comuni identici.

   b) Crea una CSR (certificate signing request) come file 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) Eseguire il software dell'autorità di certificazione o accedere al relativo sito Web. Incollare il contenuto di `CAServerReqKey.csr` quando viene richiesto il file CSR.

d) L'autorità di certificazione restituisce uno o due certificati e un file di risposta firmato come file ASCII. Incollare il contenuto in due o tre file:

Certificato root
Incolla in `CARoot.cer`

Certificato intermedio
Incolla in `CAInter.cer`

File di risposte firmato dal server
Incolla in `CAServerReq.rsp`

L'archivio certificati JRE non è utilizzato in questa attività. Se hai ricevuto un certificato root e una risposta firmata dalla CA, utilizza il certificato root e la risposta firmata nei seguenti passaggi. Se hai ricevuto un certificato root e un certificato intermedio, utilizza il certificato intermedio e la risposta firmata.

e) Ricevere la risposta del server firmato nel keystore del server da cui è stata emessa la richiesta di certificato.

La ricezione della risposta modifica il certificato autofirmato in modo che sia firmato dalla CA. Se si guarda il certificato nel keystore prima e dopo aver ricevuto la risposta, il firmatario cambia. In caso contrario, l'errore viene riportato dallo strumento di gestione delle chiavi. Prima di utilizzare il certificato, controllarlo e verificare che il firmatario sia ora la CA.

```
Keytool -import -noprompt -alias CAServer -file CAServerReq.rsp
        -keystore CAServerReqKey.jks -storepass password
```

In alcuni software di gestione delle chiavi, come **iKeyman**, si ricevono, anziché importare, i file di risposta.

f) Importare il certificato CA nel truststore client.

Importare il certificato intermedio se sono stati ricevuti due certificati dalla CA o il certificato root, se è stato ricevuto solo un certificato.

Le alternative sono:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAClientTrust.jks -storepass password
```

Oppure:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAClientTrust.jks -storepass password
```

3. Generare una chiave firmata CA per il canale di telemetria per autenticare i client.

a) Generare una coppia di chiavi autofirmata per i client in un nuovo keystore, `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
```

L'algoritmo chiave è impostato su RSA perché alcune autorità di certificazione lo richiedono. Il nome comune del certificato deve essere univoco, alcune autorità di certificazione non emettono chiavi con nomi comuni identici.

b) Crea una CSR (certificate signing request) come file 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) Eseguire il software dell'autorità di certificazione o accedere al relativo sito Web. Incollare il contenuto di `CAClientKey.csr` quando viene richiesto il file CSR.

d) L'autorità di certificazione restituisce uno o due certificati e un file di risposta firmato come file ASCII. Incollare il contenuto in due o tre file:

Certificato root
Incolla in `CARoot.cer`

Certificato intermedio
Incolla in `CAInter.cer`

File di risposta firmato dal client
Incolla in `CAClient.rsp`

L'archivio certificati JRE non è utilizzato in questa attività. Se hai ricevuto un certificato root e una risposta firmata dalla CA, utilizza il certificato root e la risposta firmata nei seguenti passaggi. Se hai ricevuto un certificato root e un certificato intermedio, utilizza il certificato intermedio e la risposta firmata.

e) Ricevere la risposta client firmata nel keystore client da cui è stata emessa la richiesta di certificato.

La ricezione della risposta modifica il certificato autofirmato in modo che sia firmato dalla CA. Se si guarda il certificato nel keystore prima e dopo aver ricevuto la risposta, il firmatario cambia. In caso contrario, l'errore viene riportato dallo strumento di gestione delle chiavi. Prima di utilizzare il certificato, controllarlo e verificare che il firmatario sia ora la CA.

```
Keytool -import -noprompt -alias CAClient -file CAClient.rsp
        -keystore CAClientKey.jks -storepass password
```

In alcuni software di gestione delle chiavi, come **iKeyman**, si ricevono, anziché importare, i file di risposta.

f) Importare il certificato CA nel keystore server.

Importare il certificato intermedio se sono stati ricevuti due certificati dalla CA o il certificato root, se è stato ricevuto solo un certificato.

Le alternative sono:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAServerReqKey.jks -storepass password
```

Oppure:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAServerReqKey.jks -storepass password
```

Creazione della tua prima applicazione publisher MQ Telemetry Transport utilizzando C

I passi per creare un'applicazione publisher del client MQTT sono descritti in modo didattico. Viene spiegata ogni riga di codice C. Alla fine dell'attività, verrà creato un publisher MQTT.

Prima di iniziare

L'applicazione client sviluppata utilizza le librerie client MQTT v3 C. L'applicazione si connette al daemon WebSphere MQ Telemetry per i dispositivi per pubblicare i messaggi. Consultare Creazione del primo publisher per un esempio di client che comunica con WebSphere MQ Telemetry.

Informazioni su questa attività

L'esempio è un'applicazione di pubblicazione, `pubsync.c`. Il programma `pubsync.c` pubblica un messaggio con il payload `Hello World!` nell'argomento `MQTT Examplee` attende la conferma della consegna della pubblicazione al daemon.

Per semplicità, i codici di ritorno di alcune funzioni utilizzate non vengono verificati per il corretto completamento. Nel codice di produzione, i codici di ritorno possono essere controllati per assicurarsi che

il programma si comporti come previsto. Se si verifica un errore imprevisto, è necessario intraprendere l'azione appropriata.

Impostando un sottoscrittore a `MQTT Example` , è possibile controllare che l'applicazione funzioni.

Utilizzare l'ambiente di sviluppo C selezionato per sviluppare, creare ed eseguire il client. Se si preferisce, è possibile copiare il codice direttamente dagli esempi.

Procedura

1. Creare un nuovo file di origine vuoto, `pubsync.c`
2. Creare un file, `settings.h`. Copiare il codice nella Figura 2 nel file.

   Tutti i parametri utilizzati nel programma sono definiti in `settings.h`. È possibile sovrascrivere i valori modificando i valori nel file.
3. I passi che seguono spiegano il codice. Seguire la procedura o copiare il codice dalla Figura 1 in `pubsync.c`.
4. Aggiungere le istruzioni di inclusione del file di intestazione per le librerie standard richieste e i file `MQTTClient.h` e `settings.h` .

   ```
   #include "stdio.h"
   #include "stdlib.h"
   #include "string.h"
   #include "MQTTClient.h"
   #include "settings.h"
   ```
5. Avviare la definizione della funzione `main()` .

   ```
   int main(int argc, char* argv[])
   {
   ```
6. Definire le variabili locali utilizzate nel programma.

   ```
   MQTTClient client;
   MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
   MQTTClient_message pubmsg;
   MQTTClient_deliveryToken token;
   int rc;
   ```

   Nota: Le opzioni di connessione sono richieste dalla funzione `MQTTClient_connect` . `MQTTClient_connectOptions_initializer` contiene le opzioni predefinite.
7. Creare un client.

   ```
   MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
   ```

   - `& client` è un puntatore a un handle per il client appena creato. Quando questa funzione restituisce un codice di ritorno 0, contiene un handle per il nuovo client. L'esempio presuppone l'esito positivo. Verificare il corretto completamento del codice di errore nel codice di produzione.
   - ADDRESS è l'URI della porta MQTT che il daemon controlla per le richieste di connessione client in entrata.
   - `CLIENTID` è il nome utilizzato per identificare il client sul daemon. Ogni cliente attivo deve avere un nome univoco. Se si duplica un identificativo client in due client in esecuzione, viene generata un'eccezione in entrambi i client e un client viene terminato. Il nome viene utilizzato dal daemon per riconoscere un client tjhat che si sta ricollegando dopo una disconnessione, consultare L'identificativo client.
   - `MQTTCLIENT_PERSISTENCE_NONE` specifica che lo stato client è conservato in memoria e viene perso se si verifica un errore di sistema. `MQTTCLIENT_PERSISTENCE_DEFAULT` specifica la persistenza basata su file system, fornendo una protezione contro gli errori. Per applicazioni più specializzate, è possibile utilizzare `MQTTCLIENT_PERSISTENCE_USER`, che fornisce un'interfaccia per implementare il proprio meccanismo di persistenza. Per ulteriori dettagli, consulta la documentazione API per `MQTTClientPersistence.h`. Se la persistenza è richiesta è una

domanda di progettazione dell'applicazione. Per ulteriori dettagli, consultare Persistenza del messaggio.

- La porta TCP/IP del daemon predefinito per MQTT è 1883. Nell'esempio, l'indirizzo predefinito è impostato su `tcp://localhost:1883`.
- Finché non si richiama la funzione `MQTTClient_connect` , non viene eseguita alcuna elaborazione del messaggio.

8. Collegare il client al daemon.

```
if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
     printf("Failed to connect, return code %d\n", rc);
     exit(-1);
}
```

- Viene richiamata la funzione `MQTTClient_connect` , passando l'handle del client e un puntatore alle opzioni di connessione come argomenti.
- Il codice di ritorno dalla chiamata `MQTTClient_connect` viene testato per accertarsi che la richiesta di connessione abbia esito positivo.
- Se `MQTTClient_connect` non riesce, il programma termina con un codice di errore di -1.
- Dopo che l'applicazione si connette, è possibile avviare la pubblicazione e la sottoscrizione.
- Un piccolo messaggio "keep-alive" viene inviato ogni 20 secondi per impedire la chiusura della connessione TCP/IP. Questa opzione è impostata da `conn_opts.keepAliveInterval`.
- La sessione viene avviata senza controllare il completamento dei messaggi inflight rimanenti da una precedente connessione poiché `conn_opts.cleansession` è impostata su true. Per ulteriori dettagli, consultare Pulisci sessioni.
- Non viene creato alcun messaggio di testamento e di ultima volontà per la connessione. Per ulteriori dettagli, consultare Ultimo testamento.

9. Popolare la struttura *MQTTClient_message* con i dati per definire il payload del messaggio e i suoi attributi.

```
pubmsg.payload = PAYLOAD;
pubmsg.payloadlen = strlen(PAYLOAD);
pubmsg.qos = QOS;
pubmsg.retained = 0;
```

- PAYLOAD è il nostro contenuto del messaggio.
- L'esempio utilizza un payload stringa ma i payload MQTT sono array di byte. La lunghezza della stringa è richiesta per specificare la dimensione del payload.
- L'esempio pubblica un messaggio QoS=1 , quindi impostare il valore di conseguenza
- L'attributo conservato è impostato su false (0) poiché il messaggio non deve essere conservato dal daemon. Per ulteriori dettagli, consultare Pubblicazione conservata.

10. Pubblicare il messaggio.

```
MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
```

- La funzione publish specifica il client, l'argomento e il payload da inviare al daemon.
- `TOPIC` è definito in `settings.h` come `MQTT Example`.
- La funzione viene passata anche un puntatore a *MQTTClient_deliveryToken*. Questo puntatore è popolato con un token che rappresenta il messaggio quando la funzione viene restituita.
- Il messaggio viene ora trasferito in modo sicuro al client MQTT, ma non ancora trasferito al daemon. Se il messaggio ha QoS=1 o 2, il messaggio viene memorizzato localmente, nel caso in cui il client abbia esito negativo prima del completamento della consegna.
- Questa funzione restituisce un codice di errore che è possibile verificare per il corretto completamento del codice di produzione.

11. Attendere il riconoscimento dal server.

```
rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
```

- L'esempio `pubsync.c` attende una conferma dal server, che conferma che il messaggio è stato consegnato.
- Gli argomenti client e token identificano il messaggio specifico per cui il programma è in attesa di completamento.
- `TIMEOUT` limita il tempo di attesa del programma per il completamento della consegna del messaggio. L'attività Creazione di un publisher asincrono per MQ Telemetry Transport utilizzando C mostra come ricevere i riconoscimenti senza attendere utilizzando le funzioni di callback.
- Questa funzione restituisce un codice di errore che può essere verificato per il corretto completamento nel codice di produzione.

12. Scollegare il client dal daemon.

```
MQTTClient_disconnect(client, 10000);
```

- Il client si disconnette dal server e attende il completamento di tutte le funzioni di callback (non utilizzate in questo esempio) per il completamento dei messaggi in corso.
- Il secondo argomento specifica un timeout di inattività in millisecondi. L'esempio attende fino a 10 secondi per terminare qualsiasi altro lavoro che deve eseguire prima di disconnettersi.
- Questa funzione restituisce un codice di errore che deve essere verificato per il corretto completamento nel codice di produzione.

13. Liberare la memoria utilizzata dal client e terminare il programma.

```
MQTTClient_destroy(&client);
}
```

Risultati

Per visualizzare le pubblicazioni inviate da questo client, creare un sottoscrittore all'argomento `MQTT Example` . Per ulteriori dettagli, consultare Creazione di un sottoscrittore per MQ Telemetry Transport utilizzando C

Esempio

La Figura 1 è un elenco completo del codice descritto in Procedura. Il file `settings.h` nella Figura 3 consente di modificare i parametri predefiniti utilizzati in `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);
}
```

Figura 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
```

Figura 112. settings.h

Creazione di un publisher asincrono per MQ Telemetry Transport utilizzando C

La procedura per creare un'applicazione di pubblicazione asincrona del client MQTT è descritta in modo didattico. Viene spiegata ogni riga di codice C. Alla fine dell'attività, verrà creato un publisher asincrono MQTT.

In questa attività, si segue un'esercitazione per modificare la prima applicazione del publisher. Le modifiche consentono all'applicazione di inviare pubblicazioni senza attendere le conferme di consegna. I riconoscimenti di consegna vengono ricevuti da una funzione di callback creata dall'utente.

Prima di iniziare

L'applicazione client sviluppata utilizza le librerie client MQTT v3 C. L'applicazione si connette al daemon WebSphere MQ Telemetry per i dispositivi per pubblicare i messaggi. Consultare Creazione del primo publisher per un esempio di client che comunica con WebSphere MQ Telemetry.

Informazioni su questa attività

L'esempio è un'applicazione di pubblicazione, `pubasync.c`. Il programma `pubasync.c` pubblica un messaggio con il payload `Hello World!` nell'argomento `MQTT Example`, senza attendere la conferma che la pubblicazione è stata consegnata al daemon. I riconoscimenti di consegna vengono ricevuti in una funzione di callback, `MQTTClient_deliveryComplete`.

Per semplicità, i codici di ritorno di alcune funzioni utilizzate non vengono verificati per il corretto completamento. Nel codice di produzione, i codici di ritorno possono essere controllati per assicurarsi che

il programma si comporti come previsto. Se si verifica un errore imprevisto, è necessario intraprendere l'azione appropriata.

Impostando un sottoscrittore a `MQTT Example` , è possibile controllare che l'applicazione funzioni.

Utilizzare l'ambiente di sviluppo C selezionato per sviluppare, creare ed eseguire il client.

I passi in Procedura modificano l'applicazione `pubsync.c` da “Creazione della tua prima applicazione publisher MQ Telemetry Transport utilizzando C” a pagina 514. Se si preferisce, è possibile copiare il codice direttamente dagli esempi.

Procedura

1. Creare un nuovo file di origine vuoto, `callback.h`.
2. Copiare il codice nella Figura 2 nel file.
   - `callback.h` dichiara i tre metodi di callback necessari per l'operazione client asincrona.
   - Viene dichiarata anche una variabile, *deliveredtoken*. A questo si accede dal programma principale e dal callback su diversi thread di esecuzione. È pertanto dichiarato volatile. Quando si utilizzano i callback, assicurarsi che si acceda alle variabili pertinenti in modo thread - safe.
3. Creare un nuovo file di origine vuoto, `callback.c`.
4. Copiare il codice nella Figura 3 nel file.
   - `callback.c` implementa i tre metodi di callback utilizzati dal client per l'operazione asincrona, `delivered`, `msgarrvde connlost`.
5. Aggiungere un'istruzione include per `callback.h` dopo le altre inclusioni in `pubasync.c`.

   ```
   #include "callback.h"
   ```

6. Copiare il contenuto di `pubsync.c` in un nuovo file, `pubasync.c`.
7. Prima della chiamata della funzione `MQTTClient_connect` in `pubasync.c`, impostare i metodi di callback per il client.

   ```
   MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);
   ```

   - È necessario specificare tre funzioni di callback. Queste funzioni sono implementate in `callback.c`.
   - `MQTTClient_messageArrived` viene richiamato quando un messaggio viene inviato al client a causa di una sottoscrizione corrispondente. Questo deve restituire true quando il messaggio ricevuto è stato ricevuto correttamente dall'applicazione client. La restituzione di false indica al client che si è verificato un problema con la ricezione del messaggio da parte dell'applicazione.
   - `MQTTClient_connectionLost` viene richiamato quando il client perde la connessione al server.
   - `MQTTClient_deliveryComplete` viene richiamato quando un messaggio QoS1 o QoS2 è arrivato ed è stato riconosciuto dal server. Non viene richiamata per i messaggi QoS0 . Nell'esempio, questa funzione salva il token dal messaggio consegnato in *deliveredtoken* per indicare che è arrivato un messaggio.
   - `MQTTClient_setCallbacks` deve essere chiamato mentre il client è disconnesso dal server.
   - Il secondo argomento consente di passare informazioni contestuali alle funzioni di callback. Non viene utilizzato nell'esempio, quindi è impostato su NULL.
8. Immediatamente prima della chiamata a `MQTTClient_publishMessage`, cancellare *deliveredtoken*. `MQTTClient_deliveryComplete` per impostare *deliveredtoken* quando viene ricevuto un token.

   ```
   deliveredtoken = 0;
   ```

9. Rimuovere la chiamata `MQTTClient_waitForCompletion` e l'istruzione `printf` che la seguono e sostituirla con un loop in attesa di una corrispondenza del token originale e del token ricevuto nel callback.

```
while(deliveredtoken != token);
```

Questo è un esempio e non fa fronte a una serie di situazioni che devono essere sistemate nella progettazione del codice di produzione. Queste situazioni includono:

- Nel caso in cui la consegna non venga completata, è possibile implementare un timeout
- Più messaggi possono essere in fase di ricezione. Il programma di esempio consente di controllare un solo token di consegna alla volta.

10. Scollegare il client dal daemon.

```
MQTTClient_disconnect(client, 10000);
```

- Il client si disconnette dal server e attende il completamento di tutte le funzioni di callback per i messaggi in corso.
- Il secondo argomento specifica un timeout di inattività in millisecondi. L'esempio attende fino a 10 secondi per terminare qualsiasi altro lavoro che deve eseguire prima di disconnettersi.
- Questa funzione restituisce un codice di errore che deve essere verificato per il corretto completamento nel codice di produzione.

11. Liberare la memoria utilizzata dal client e terminare il programma.

```
MQTTClient_destroy(&client);
}
```

Risultati

Per visualizzare la pubblicazione inviata da questo client, creare un sottoscrittore all'argomento `MQTT Example` . Per ulteriori dettagli, consultare Creazione di un sottoscrittore per MQ Telemetry Transport

Esempio

`pubasync.c`, `callbacks.c` e `callbacks.h` sono elenchi completi del codice descritto in Procedura.

```
#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);
}
```

Figura 113. pubasync.c

```
MQTTClient_deliveryComplete delivered;
MQTTClient_messageArrived msgarrvd;
MQTTClient_connectionLost connlost;

extern volatile MQTTClient_deliveryToken deliveredtoken;
```

Figura 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);
}
```

Figura 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
```

Figura 116. settings.h

Creazione di un sottoscrittore per MQ Telemetry Transport utilizzando C

La procedura per creare un'applicazione del sottoscrittore client MQTT è descritta in modo didattico. Viene spiegata ogni riga di codice C. Alla fine dell'attività, verrà creato un sottoscrittore MQTT.

Prima di iniziare

L'applicazione client sviluppata utilizza le librerie client MQTT v3 C. L'applicazione si connette al daemon WebSphere MQ Telemetry per i dispositivi per pubblicare i messaggi. Consultare Creazione del primo publisher per un esempio di client che comunica con WebSphere MQ Telemetry.

Informazioni su questa attività

L'esempio è un'applicazione sottoscrittore, `subscribe.c`. Il programma `subscribe.c` esegue la sottoscrizione all'argomento `MQTT Example` e attende le pubblicazioni che corrispondono alla sottoscrizione fino a quando l'utente termina il programma.

Un sottoscrittore crea una sottoscrizione a un argomento e attende i messaggi che corrispondono all'argomento della sottoscrizione. I messaggi pubblicati mentre il client è disconnesso e che corrispondono a una sottoscrizione creata in precedenza dal client, possono essere ricevuti quando il client si riconnette. Il servizio o daemon MQXR (WebSphere MQ telemetry) per i dispositivi riconosce un client che è stato precedentemente connesso dall'identificativo client. Per ulteriori informazioni, consultare L'identificatore client. L'attributo booleano `MQTTClient_connectOptions.cleansession` controlla se le pubblicazioni inviate in precedenza vengono ricevute o meno. Per ulteriori dettagli, vedere “Ripulisci sessioni” a pagina 531.

Per semplicità, i codici di ritorno di alcune funzioni utilizzate non vengono verificati per il corretto completamento. Nel codice di produzione, è possibile controllare i codici di ritorno per assicurarsi che il programma funzioni come previsto. Se si verifica un errore imprevisto, è possibile intraprendere l'azione appropriata.

È possibile utilizzare i programmi di esempio di pubblicazione precedentemente descritti per inviare pubblicazioni corrispondenti al daemon WebSphere MQ Telemetry per dispositivi. In alternativa, utilizzare WebSphere MQ Explorer per creare pubblicazioni di verifica sull'argomento `MQTT Example` se si desidera collegare il client ad un canale WebSphere MQ Telemetry .

Le istruzioni contenute in Procedura presuppongono che siano già stati creati file `callback.c`, `callback.h` e `settings.h` in una delle attività precedenti.

Utilizzare l'ambiente di sviluppo C selezionato per sviluppare, creare ed eseguire il client. Se si preferisce, è possibile copiare il codice direttamente dagli esempi.

Procedura

1. Creare una copia di `settings.h` per questo esempio e modificare l'istruzione di definizione *CLIENTID* nel seguente modo:

   ```
   #define CLIENTID "ExampleClientSub"
   ```

   - Se due client con lo stesso ID tentano di connettersi a un singolo server, uno di essi viene disconnesso in modo forzato. Di solito, il nuovo tentativo di connessione viene eseguito correttamente e la connessione precedente viene disconnessa.
   - La modifica di `ClientID` consente di utilizzare gli esempi di pubblicazione precedentemente sviluppati per inviare messaggi a questo sottoscrittore.
2. Creare un nuovo file di origine vuoto, `subscribe.c`.
3. I passi che seguono spiegano il codice. Seguire i passi o copiare il codice da Figura 117 a pagina 526 nel file `subscribe.c`.
4. Aggiungere le istruzioni di inclusione del file di intestazione per le librerie standard richieste e i file `MQTTClient.h` e `settings.h` .

   ```
   #include "stdio.h"
   #include "stdlib.h"
   #include "MQTTClient.h"
   #include "settings.h"
   ```
5. Avviare la definizione della funzione `main()` .

   ```
   int main(int argc, char* argv[]) {
   ```
6. Definire le variabili locali utilizzate nel programma.

   ```
   MQTTClient client;
   MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
   MQTTClient_deliveryToken token;
   int rc;
   ```

   Le opzioni di connessione sono richieste dalla funzione `MQTTClient_connect` . `MQTTClient_connectOptions_initializer` contiene le opzioni predefinite.
7. Creare un client.

```
MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
```

- `& client` è un puntatore a un handle per il client appena creato. Quando questa funzione restituisce un codice di ritorno 0, il puntatore contiene un handle per il nuovo client. L'esempio presuppone l'esito positivo. È possibile verificare il corretto completamento del codice di errore nel codice di produzione.
- ADDRESS è l'URI della porta MQTT che il daemon controlla per le richieste di connessione client in entrata.
- `CLIENTID` è il nome utilizzato per identificare il client sul daemon. Ogni cliente attivo deve avere un nome univoco. Se si duplica un identificativo client in due client in esecuzione, viene generata un'eccezione in entrambi i client e un client viene terminato. Il nome viene utilizzato dal daemon per riconoscere che un client si sta riconnettendo dopo una disconnessione, consultare L'identificativo client.
- `MQTTCLIENT_PERSISTENCE_NONE` specifica che lo stato client è conservato in memoria e viene perso se si verifica un errore di sistema. `MQTTCLIENT_PERSISTENCE#_DEFAULT` specifica la persistenza basata sul file system, fornendo una protezione contro gli errori. Per applicazioni più specializzate, è possibile utilizzare `MQTTCLIENT_PERSISTENCE_USER`, che fornisce un'interfaccia per implementare il proprio meccanismo di persistenza. Se la persistenza è richiesta è una domanda di progettazione dell'applicazione. Per ulteriori dettagli, consultare Persistenza del messaggio.
- La porta TCP/IP del daemon predefinito per MQTT è 1883. Nell'esempio, l'indirizzo predefinito è impostato su `tcp://localhost:1883`.
- Finché non si richiama la funzione `MQTTClient_connect` , non viene eseguita alcuna elaborazione del messaggio.

8. Connetti il client al daemon

```
if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
    printf("Failed to connect, return code %d\n", rc);
    exit(-1);
}
```

- Viene richiamata la funzione `MQTTClient_connect` , passando l'handle del client e un puntatore alle opzioni di connessione come argomenti.
- Il codice di ritorno dalla chiamata `MQTTClient_connect` viene testato per accertarsi che la richiesta di connessione abbia esito positivo.
- Se la chiamata di connessione ha esito negativo, il programma termina con un codice di errore di -1.
- Una volta che l'applicazione si connette, può iniziare la pubblicazione e la sottoscrizione.
- Un piccolo messaggio "keep-alive" viene inviato ogni 20 secondi per impedire la chiusura della connessione TCP/IP. Questa opzione è impostata da `conn_opts.keepAliveInterval`.
- La sessione viene avviata senza verificare il completamento dei messaggi inflight rimanenti da una connessione precedente perché *conn_opts.cleansession* è impostata su true. Per ulteriori dettagli, consultare Pulisci sessioni.
- Non viene creato alcun messaggio di testamento e di ultima volontà per la connessione. Per ulteriori dettagli, consultare Ultimo testamento

9. Sottoscrivere l'argomento.

```
MQTTClient_subscribe(client, TOPIC, QOS);
```

- Utilizzare la funzione `MQTTClient_subscribe` per sottoscrivere l'applicazione client all'argomento selezionato. Il nome dell'argomento può includere caratteri jolly. Per ulteriori dettagli, vedere “Stringhe argomento e filtri argomento nei client MQTT” a pagina 546.
- L'impostazione QoS determina la QoS (quality of service) massima applicata ai messaggi inviati a questo sottoscrittore. Il server invia i messaggi al valore più basso di questa impostazione e l'impostazione QoS per il messaggio originale.

- Questa funzione restituisce un codice di errore che può essere verificato per il corretto completamento nel codice di produzione.

10. Attendere un loop finché l'utente non immette un carattere 'Q' dalla tastiera.

```
do {
    ch = getchar();
    } while(ch!='Q' && ch != 'q');
```

Il programma ora attende l'arrivo dei messaggi. In questo esempio, tutta la gestione dei messaggi avviene nella funzione di callback `MQTTClient_messageArrived`. Per ulteriori dettagli, vedere “ricezione di messaggi” a pagina 524.

11. Scollegare il client dal daemon.

```
MQTTClient_disconnect(client, 10000);
```

- Il client si disconnette dal server e attende il completamento di tutte le funzioni di callback (non utilizzate in questo esempio) per il completamento dei messaggi in corso.
- Il secondo argomento specifica un timeout di inattività in millisecondi. L'esempio attende fino a 10 secondi per terminare qualsiasi altro lavoro che deve eseguire prima di disconnettersi.
- Questa funzione restituisce un codice di errore che può essere verificato per il corretto completamento nel codice di produzione.

12. Liberare la memoria utilizzata dal client e terminare il programma.

```
MQTTClient_destroy(&client);
}
```

ricezione di messaggi

Informazioni su questa attività

Quando i messaggi arrivano dal server, viene avviata la funzione `MQTTClient_messageArrived` . I passi che seguono spiegano il codice.

Procedura

1. Avviare la definizione della funzione callback. Questa definizione deve corrispondere alla maschera della funzione `MQTTClient_messageArrived` .

```
int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message) {
```

- `context` fornisce l'accesso al contesto passato alla libreria client quando è stata richiamata la funzione `MQTTClient_setCallbacks` . Questa funzione non viene utilizzata nell'esempio.
- `topicName` è un puntatore all'argomento in cui viene pubblicato il messaggio ricevuto. Se è stata effettuata la sottoscrizione utilizzando caratteri jolly, questo parametro identifica l'argomento specifico utilizzato per il messaggio.
- `topicLen` è la lunghezza della stringa argomento. Questa opzione viene fornita per gli utenti che devono integrare caratteri NULL nelle stringhe di argomento.
- `message` è un puntatore alla struttura `MQTTClient_message` contenente il payload e gli attributi del messaggio.

2. Definire le variabili locali utilizzate.

```
int i;
char* payloadptr;
```

Queste variabili vengono utilizzate nell'esempio per stampare il payload iterando su di esso.

3. Stampare un messaggio, visualizzando l'argomento e il payload del messaggio

```
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');
```

- L'esempio presuppone che il payload ricevuto sia una sequenza di caratteri stampabili.
- Un payload MQTT è un array di byte. L'applicazione è responsabile dell'interpretazione del loro significato.

4. Liberare la memoria utilizzata per memorizzare il messaggio.

```
MQTTClient_freeMessage(&message);
MQTTClient_free(topicName);
```

- Nell'esempio, tutta la gestione dei messaggi avviene nella funzione di callback.
- Assicurarsi che le funzioni di callback siano brevi e restituire il controllo al thread di chiamata il più presto possibile.
- Il puntatore del messaggio viene passato per la gestione nella parte principale del programma.
- Il programma principale deve liberare la memoria utilizzata dal messaggio quando l'elaborazione è completa. `MQTTClient_freeMessage()` è una funzione utile che restituisce i due blocchi di memoria utilizzati per contenere la struttura *MQTTClient_message* e il payload del messaggio al sistema. La memoria assegnata al `topicName` deve essere liberata separatamente come mostrato.

5. Restituisce un valore true quando il callback ha gestito correttamente il messaggio

```
    return 1;
}
```

- La restituzione di un valore true indica che la libreria del client può considerare il messaggio come consegnato correttamente.
- Se la funzione di callback non è in grado di elaborare correttamente il messaggio, viene restituito un valore false. Ad esempio, se il callback sta inserendo i messaggi in una coda per il programma principale da elaborare e la coda è piena, la restituzione di false è appropriata.
- Per i messaggi QoS1 e QoS2 , la restituzione di un valore false indica che il messaggio non è stato consegnato e vengono effettuati ulteriori tentativi per consegnarlo.

Codice di esempio

```
#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);
}
```

Figura 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);
}
```

Figura 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
```

Figura 119. settings.h

Concetti di programmazione del client MQTT

I concetti descritti in questa sezione consentono di comprendere le librerie client Java, JavaScript e C per la versione 3.1 di MQTT protocol. I concetti completano la documentazione API che accompagna le librerie client.

`com.ibm.micro.client.mqttv3` contiene le classi che forniscono i metodi pubblici per le librerie client per il protocollo MQTT versione 3.1 . Una versione del pacchetto `com.ibm.micro.client.mqttv3` e i relativi pacchetti che implementano il protocollo per Java SE e ME, viene fornita con l'installazione di IBM WebSphere MQ Telemetry. Per ottenere la versione più recente delle librerie client MQTT (Java, JavaScript e per visualizzare o scaricare la documentazione API, consultare "MQTT client programming reference".

Per sviluppare ed eseguire un client MQTT , è necessario copiare o installare questi pacchetti sulla periferica client. Non è necessario installare un runtime client separato.

Le condizioni di licenza per client sono associate al server a cui si stanno collegando i client.

Le librerie client MQTT sono implementazioni di riferimento della versione 3.1 di MQTT protocol. È possibile implementare i propri client in diverse lingue adatte a diverse piattaforme di dispositivi. Vedere MQ Telemetry Transport format and protocol.

La documentazione API non fa supposizioni su quale server MQTT è connesso al client. Il comportamento del client potrebbe differire leggermente quando si è connessi a server differenti. Le descrizioni che seguono descrivono il comportamento del client quando è connesso al servizio di telemetria IBM WebSphere MQ .

Callback e sincronizzazione nelle applicazioni client MQTT

Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Callback

L'interfaccia `MqttCallback` dispone di tre metodi di callback; vedi un'implementazione di esempio in Callback.java.

`connectionLost(java.lang.Throwable cause)`

`connectionLost` viene chiamato quando un errore di comunicazione porta al rilascio della connessione. Viene anche richiamato se il server elimina la connessione come risultato di un errore sul server dopo che la connessione è stata stabilita. Gli errori del server vengono registrati nel log degli errori del gestore code. Il server elimina la connessione al client e il client richiama `MqttCallback.connectionLost`.

Gli unici errori remoti generati come eccezioni sullo stesso thread dell'applicazione client sono eccezioni da `MqttClient.connect`. Gli errori rilevati dal server una volta stabilita la connessione vengono riportati al metodo di callback `MqttCallback.connectionLost` come `throwables`.

Gli errori tipici del server che risultano in `connectionLost` sono errori di autorizzazione. Ad esempio, il server di telemetria tenta di pubblicare su un argomento per conto di un client che non è autorizzato a pubblicare sull'argomento. Tutto ciò che risulta in un codice condizione `MQCC_FAIL` restituito al server di telemetria può causare l'eliminazione della connessione.

`deliveryComplete(MqttDeliveryToken token)`

`deliveryComplete` viene chiamato dal client MQTT per passare un token di consegna all'applicazione client; consultare “Token di consegna” a pagina 533. Utilizzando il token di consegna, il callback può accedere al messaggio pubblicato con il metodo `token.getMessage`.

Quando il callback dell'applicazione restituisce il controllo al client MQTT dopo essere stato richiamato dal metodo `deliveryComplete` , la consegna viene completata. Fino al completamento della consegna, i messaggi con QoS 1 o 2 vengono conservati dalla classe di persistenza.

La chiamata a deliveryComplete è un punto di sincronizzazione tra l'applicazione e la classe di persistenza. Il metodo deliveryComplete non viene mai richiamato due volte per lo stesso messaggio.

Quando il callback dell'applicazione viene restituito da deliveryComplete al client MQTT , il client richiama MqttClientPersistence.remove per i messaggi con QoS 1 o 2. MqttClientPersistence.remove elimina la copia memorizzata localmente del messaggio pubblicato.

Da una prospettiva di elaborazione della transazione, la chiamata a deliveryComplete è una transazione a fase singola che esegue il commit della consegna. Se l'elaborazione non riesce durante il callback, al riavvio del client MqttClientPersistence.remove viene richiamato di nuovo per eliminare la copia locale del messaggio pubblicato. Il callback non viene richiamato di nuovo. Se si utilizza il callback per memorizzare un log dei messaggi consegnati, non è possibile sincronizzare il log con il client MQTT . Se si desidera memorizzare un log in modo affidabile, aggiornare il log nella classe MqttClientPersistence .

Il token di consegna e il messaggio sono indicati dal thread dell'applicazione principale e dal client MQTT . Il client MQTT annulla il riferimento all'oggetto MqttMessage quando la consegna è completata e l'oggetto token di consegna quando il client si disconnette. L'oggetto MqttMessage può essere raccolto nel garbage collector dopo il completamento della consegna se l'applicazione client lo annulla. Il token di consegna può essere un raccoglitore dati inutilizzati dopo che la sessione è stata disconnessa.

È possibile ottenere gli attributi MqttDeliveryToken e MqttMessage dopo la pubblicazione di un messaggio. Se si tenta di impostare gli attributi MqttMessage dopo che il messaggio è stato pubblicato, il risultato non è definito.

Il client MQTT continua ad elaborare le conferme di consegna se il cliente si riconnette alla precedente sessione con lo stesso ClientIdentifier; consultare “Ripulisci sessioni” a pagina 531. L'applicazione client MQTT deve impostare MqttClient.CleanSession su false per la precedente e impostarla su false nella nuova sessione. Il client MQTT crea nuovi token di consegna e oggetti messaggio nella nuova sessione per le consegne in sospeso. Recupera gli oggetti utilizzando la classe MqttClientPersistence . Se il client dell'applicazione ha ancora riferimenti ai vecchi token di consegna e messaggi, annullarli. La richiamata dell'applicazione viene richiamata nella nuova sessione per tutte le distribuzioni avviate nella sessione precedente e completate in questa sessione.

Il callback dell'applicazione viene richiamato dopo la connessione del client dell'applicazione, quando viene completata una consegna in sospeso. Prima che il client dell'applicazione si colleghi, è possibile richiamare le consegne in sospeso utilizzando il metodo MqttClient.getPendingDeliveryTokens .

Si noti che l'applicazione client ha originariamente creato l'oggetto del messaggio pubblicato e il relativo array di byte del payload. Il client MQTT fa riferimento a questi oggetti. L'oggetto messaggio restituito dal token di consegna nel metodo token.getMessage non è necessariamente lo stesso oggetto messaggio creato dal client. Se una nuova istanza client MQTT ricrea il token di distribuzione, la classe MqttClientPersistence ricrea l'oggetto MqttMessage . Per coerenza token.getMessage restituisce null se token.isCompleted è true, indipendentemente dal fatto che l'oggetto del messaggio sia stato creato dal client dell'applicazione o dalla classe MqttClientPersistence .

messageArrived(MqttTopic topic, MqttMessage message)

messageArrived viene richiamato quando arriva una pubblicazione per il client che corrisponde a un argomento di sottoscrizione. topic è l'argomento di pubblicazione, non il filtro di sottoscrizione. I due possono essere diversi se il filtro contiene caratteri jolly.

Se l'argomento corrisponde a più sottoscrizioni create dal client, il client riceve più copie della pubblicazione. Se un client pubblica su un argomento a cui è anche sottoscrittore, riceve una copia della propria pubblicazione.

Se un messaggio viene inviato con un QoS di 1 o 2, il messaggio viene memorizzato dalla classe MqttClientPersistence prima che il client MQTT chiami messageArrived. messageArrived funziona come deliveryComplete: viene richiamato una sola volta per una pubblicazione e la copia locale della pubblicazione viene rimossa da

`MqttClientPersistence.remove` quando `messageArrived` ritorna al client MQTT . Il client MQTT elimina i riferimenti all'argomento e al messaggio quando `messageArrived` ritorna al client MQTT . L'argomento e gli oggetti del messaggio vengono raccolti nel raccoglitore dati inutilizzati, se il client delle applicazioni non ha mantenuto un riferimento agli oggetti.

Callback, thread e sincronizzazione dell'applicazione client

Il client MQTT richiama un metodo callback su un thread separato al thread dell'applicazione principale. L'applicazione client non crea un thread per il callback, ma viene creata dal client MQTT .

Il client MQTT sincronizza i metodi di callback. Viene eseguita una sola istanza del metodo callback alla volta. La sincronizzazione facilita l'aggiornamento di un oggetto che indica quali pubblicazioni sono state consegnate. Un'istanza di `MqttCallback.deliveryComplete` viene eseguita alla volta, quindi è sicuro aggiornare il conteggio senza ulteriore sincronizzazione. È anche il caso di una sola pubblicazione alla volta. Il codice nel metodo `messageArrived` può aggiornare un oggetto senza sincronizzarlo. Se si fa riferimento al conteggio o all'oggetto che si sta aggiornando, in un altro thread, sincronizzare il conteggio o l'oggetto.

Il token di consegna fornisce un meccanismo di sincronizzazione tra il thread dell'applicazione principale e la distribuzione di una pubblicazione. Il metodo `token.waitForCompletion` attende il completamento della consegna di una pubblicazione specifica o la scadenza di un timeout facoltativo. È possibile utilizzare `token.waitForCompletion` in un paio di semplici modi per elaborare una pubblicazione alla volta:

1. Per sospendere il client applicativo fino al completamento della distribuzione della pubblicazione, consultare Figura 88 a pagina 482.
2. Per sincronizzarsi con il metodo `MqttCallback.deliveryComplete` . Solo quando `MqttCallback.deliveryComplete` ritorna al client MQTT `token.waitForCompletion` viene ripreso. Utilizzando questo meccanismo, è possibile sincronizzare il codice in esecuzione in `MqttCallback.deliveryComplete` prima che il codice venga eseguito nel thread dell'applicazione principale.

Cosa fare se si desidera pubblicare senza attendere la consegna di ogni pubblicazione, ma si desidera una conferma quando tutte le pubblicazioni sono state consegnate? Se si esegue la pubblicazione su un singolo thread, l'ultima pubblicazione da inviare è anche l'ultima da consegnare.

Sincronizzazione delle richieste inviate al server

La Tabella 70 a pagina 529 descrive i metodi nel client MQTT Java che inviano una richiesta al server. A meno che il client delle applicazioni non imposti un timeout indefinito, il client non attende mai indefinitamente il server. Se il client si blocca, si tratta di un problema di programmazione dell'applicazione o di un difetto nel client MQTT .

Tabella 70. Comportamento di sincronizzazione dei metodi che risultano in richieste al server

Metodo	**Sincronizzazione**	**Intervallo di timeout**
`MqttClient.Connect`	Attende che venga stabilita una connessione con il server.	Il valore predefinito è 30 secondi oppure, come impostato da un parametro, genera un'eccezione.
`MqttClient.Disconnect`	Attende che il client MQTT termini il lavoro che deve eseguire e che la sessione TCP/IP si scolleghi.	
`MqttClient.Subscribe`	Attende il completamento del metodo Sottoscrivi o UnSubscribe .	
`MqttClient.UnSubscribe`		

Tabella 70. Comportamento di sincronizzazione dei metodi che risultano in richieste al server (Continua)		
Metodo	**Sincronizzazione**	**Intervallo di timeout**
`MqttClient.Publish`	Ritorna immediatamente al thread dell'applicazione dopo aver passato la richiesta al client MQTT .	Nessuna.
`MqttDeliveryToken.waitForCompletion`	Attende la restituzione del token di consegna.	Indefinito o impostato come parametro.

Concetti correlati

Ripulisci sessioni

Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Identificativo client

Token di consegna

Ultimo testamento e pubblicazione testamentaria

Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Persistenza del messaggio nei client MQTT

Pubblicazioni

Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Qualità del servizio fornito da un client MQTT

Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

Pubblicazioni conservate e client MQTT

Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Sottoscrizione

Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Stringhe argomento e filtri argomento nei client MQTT

Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Ripulisci sessioni

Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Quando si connette un'applicazione client MQTT utilizzando il metodo `MqttClient.connect` , il client identifica la connessione utilizzando l'identificativo client e l'indirizzo del server. Il server verifica se le informazioni sulla sessione sono state salvate da una precedente connessione al server. Se una sessione precedente esiste ancora e `cleanSession=true`, le informazioni sulla sessione precedente sul client e sul server vengono cancellate. Se `cleanSession=false` la sessione precedente viene ripresa. Se non esiste alcuna sessione precedente, viene avviata una nuova sessione.

Nota: L'amministratore WebSphere MQ può chiudere forzatamente una sessione aperta ed eliminare tutte le informazioni sulla sessione. Se il client riapre la sessione con `cleanSession=false`, viene avviata una nuova sessione.

Pubblicazioni

Se si utilizza il valore predefinito `MqttConnectOptions`o si imposta `MqttConnectOptions.cleanSession` su `true` prima di collegare il client, tutte le distribuzioni di pubblicazione in sospeso per il client vengono rimosse quando il client si connette.

L'impostazione di ripulitura della sessione non ha alcun effetto sulle pubblicazioni inviate con `QoS=0`. Per `QoS=1` e `QoS=2`, l'utilizzo di `cleanSession=true` potrebbe comportare la perdita di una pubblicazione.

Sottoscrizione

Se si utilizza il valore predefinito `MqttConnectOptions` o si imposta `MqttConnectOptions.cleanSession` su `true` prima di connettere il client, tutte le sottoscrizioni precedenti del client vengono rimosse alla connessione del client. Qualsiasi nuova sottoscrizione effettuata dal client durante la sessione viene rimossa quando si disconnette.

Se si imposta `MqttConnectOptions.cleanSession` su `false` prima della connessione, qualsiasi sottoscrizione creata dal client viene aggiunta a tutte le sottoscrizioni esistenti per il client prima della sua connessione. Tutte le sottoscrizioni restano attive alla disconnessione del client.

Un altro modo per comprendere come l'attributo `cleanSession` influisce sulle sottoscrizioni è considerarlo come un attributo modale. Nella sua modalità predefinita, `cleanSession=true`, il client crea sottoscrizioni e riceve pubblicazioni solo nell'ambito della sessione. Nella modalità alternativa, `cleanSession=false`, le sottoscrizioni sono durevoli. Il client può connettersi e disconnettersi e le sue sottoscrizioni rimangono attive. Quando il client si riconnette, riceve tutte le pubblicazioni non consegnate. Mentre è connesso, può modificare l'insieme di sottoscrizioni attive per suo conto.

È necessario impostare la modalità `cleanSession` prima della connessione; la modalità è valida per l'intera sessione. Per modificarne l'impostazione, è necessario disconnettere e riconnettere il client. Se si modifica la modalità da `cleanSession=false` a `cleanSession=true`, tutte le sottoscrizioni precedenti destinate al client e le eventuali pubblicazioni non ricevute vengono eliminate.

Concetti correlati

Callback e sincronizzazione nelle applicazioni client MQTT
Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Identificativo client
Token di consegna

Ultimo testamento e pubblicazione testamentaria
Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Persistenza del messaggio nei client MQTT

Pubblicazioni
Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Qualità del servizio fornito da un client MQTT
Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

Pubblicazioni conservate e client MQTT
Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Sottoscrizione
Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Stringhe argomento e filtri argomento nei client MQTT
Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Identificativo client

L'identificativo client è una stringa di 23 byte che identifica un client MQTT . Ogni ID deve essere univoco a un solo client connesso per volta. L'identificativo deve contenere solo caratteri validi in un nome gestore code. All'interno di questi vincoli, è possibile utilizzare qualsiasi stringa di identificazione. È importante disporre di una procedura per l'assegnazione di identificativi client e di un mezzo per la configurazione di un client con l'identificativo scelto.

L'identificativo client viene utilizzato nella gestione di un sistema MQTT . Con potenzialmente centinaia di migliaia di clienti da amministrare, è necessario essere in grado di identificare rapidamente un particolare cliente. Si supponga, ad esempio, che un dispositivo non abbia funzionato correttamente e che l'utente riceva una notifica, forse da un cliente che squilla un help desk. Come il cliente identifica il dispositivo e come si correla tale identificazione con il server che di solito è connesso al client? È necessario consultare un database che associa ogni periferica a un identificativo client e a un server? Il nome della periferica identifica il server a cui è collegata? Quando si sfogliano le connessioni client MQTT , ogni connessione viene etichettata con l'identificativo client. È necessario ricercare una tabella per associare un identificativo client a una periferica fisica?

L'identificativo del client identifica una particolare periferica, un utente o un'applicazione in esecuzione sul client? Se un cliente sostituisce un dispositivo difettoso con uno nuovo, il nuovo dispositivo ha lo stesso identificativo del vecchio dispositivo? Assegnare un nuovo identificatore? Se si modifica un dispositivo fisico, ma si conserva lo stesso identificativo, le pubblicazioni in sospeso e le sottoscrizioni attive vengono automaticamente trasferite al nuovo dispositivo.

Come si garantisce che gli identificatori client siano univoci? Oltre a un sistema per la generazione di identificativi univoci, è necessario disporre di un processo affidabile per l'impostazione dell'identificativo sul client. Forse il dispositivo client è una "black-box", senza interfaccia utente. Si produce il dispositivo con un identificativo client, ad esempio utilizzando il relativo indirizzo MAC? Oppure si dispone di un

processo di installazione e configurazione del software che configura il dispositivo prima che venga attivato?

È possibile creare un identificativo client dall'indirizzo MAC della periferica a 48 bit, per mantenere l'identificativo breve e univoco. Se la dimensione di trasmissione non è un problema critico, è possibile utilizzare i restanti 17 byte per semplificare l'amministrazione dell'indirizzo.

Concetti correlati

Callback e sincronizzazione nelle applicazioni client MQTT
Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Ripulisci sessioni
Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Token di consegna

Ultimo testamento e pubblicazione testamentaria
Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Persistenza del messaggio nei client MQTT

Pubblicazioni
Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Qualità del servizio fornito da un client MQTT
Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

Pubblicazioni conservate e client MQTT
Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Sottoscrizione
Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Stringhe argomento e filtri argomento nei client MQTT
Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Token di consegna

Quando un client pubblica un argomento, viene creato un nuovo token di consegna. Utilizzare il token di consegna per monitorare la consegna di una pubblicazione o per bloccare l'applicazione client fino al completamento della consegna.

Il token è un oggetto `MqttDeliveryToken` . Viene creato richiamando il metodo MqttTopic.publish () e viene conservato dal client MQTT fino a quando la sessione client non viene disconnessa e la consegna non viene completata.

Il normale utilizzo del token è quello di verificare se la consegna è completa. Blocca l'applicazione client fino al completamento della consegna utilizzando il token restituito per richiamare `token.waitForCompletion`. In alternativa, fornire un handler `MqttCallBack` . Quando il client MQTT ha ricevuto tutti i riconoscimenti previsti come parte della consegna della pubblicazione, richiama `MqttCallBack.deliveryComplete` passando il token di consegna come parametro.

Fino a quando la consegna non è completa, puoi ispezionare la pubblicazione utilizzando il token di consegna restituito chiamando `token.getMessage`.

Consegne completate

Il completamento delle consegne è asincrono e dipende dalla qualità del servizio associato alla pubblicazione.

Al massimo una volta

`QoS=0`
La consegna è completa immediatamente al ritorno da `MqttTopic.publish`. `MqttCallback.deliveryComplete` viene richiamato immediatamente.

Almeno una volta

`QoS=1`
Il recapito è completo quando è stato ricevuto un riconoscimento alla pubblicazione dal gestore code. `MqttCallback.deliveryComplete` viene richiamato quando viene ricevuto il riconoscimento. Il messaggio potrebbe essere consegnato più di una volta prima che venga richiamato `MqttCallback.deliveryComplete` , se le comunicazioni sono lente o inaffidabili.

Esattamente una volta

`QoS=2`
La consegna è completa quando il client riceve un messaggio di completamento che indica che la pubblicazione è stata pubblicata per i sottoscrittori. `MqttCallback.deliveryComplete` viene richiamato non appena viene ricevuto il messaggio di pubblicazione. Non attende il messaggio di completamento.

In rari casi, l'applicazione client potrebbe non tornare al client MQTT da `MqttCallback.deliveryComplete` normalmente. Sai che la consegna è stata completata, perché `MqttCallback.deliveryComplete` è stato chiamato. Se il client riavvia la stessa sessione, `MqttCallback.deliveryComplete` non viene richiamato di nuovo.

Consegne incomplete

Se la distribuzione non è completa dopo la disconnessione della sessione client, è possibile connettere nuovamente il client e completare la distribuzione. È possibile completare la consegna di un messaggio solo se il messaggio è stato pubblicato in una sessione con l'attributo `MqttConnectionOptions` impostato su `false`.

Creare il client utilizzando lo stesso identificativo client e lo stesso indirizzo server, quindi connettersi, impostando nuovamente l'attributo `cleanSession` `MqttConnectionOptions` su `false` . Se si imposta `cleanSession` su `true`, i token di consegna in sospeso vengono eliminati.

È possibile verificare se sono presenti recapiti in sospeso chiamando `MqttClient.getPendingDeliveryTokens`. È possibile chiamare `MqttClient.getPendingDeliveryTokens` prima di collegare il client.

Concetti correlati

Callback e sincronizzazione nelle applicazioni client MQTT
Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e

dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Ripulisci sessioni
Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Identificativo client

Ultimo testamento e pubblicazione testamentaria
Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Persistenza del messaggio nei client MQTT

Pubblicazioni
Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Qualità del servizio fornito da un client MQTT
Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

Pubblicazioni conservate e client MQTT
Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Sottoscrizione
Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Stringhe argomento e filtri argomento nei client MQTT
Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Ultimo testamento e pubblicazione testamentaria

Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Creare un argomento per l'ultimo testamento. È possibile creare un argomento come `MQTTManagement/Connections/`*`server URI`*`/`*`client identifer`*`/Lost`.

Impostare un "ultimo testamento" utilizzando il metodo `MqttConnectionOptions.setWill(MqttTopic lastWillTopic, byte [] lastWillPayload, int lastWillQos, boolean lastWillRetained)` .

Considerare la creazione di una data / ora nel messaggio `lastWillPayload` . Includere altre informazioni sul client che consentono di identificare il client e le circostanze della connessione. Passare l'oggetto `MqttConnectionOptions` al costruttore `MqttClient` .

Impostare `lastWillQos` su 1 o 2, per rendere il messaggio persistente in WebSphere MQe per garantire la consegna. Per conservare le informazioni sull'ultima connessione persa, impostare `lastWillRetained` su `true`.

La pubblicazione "Last Will and Testament" viene inviata ai sottoscrittori se la connessione termina in modo imprevisto. Viene inviato se la connessione termina senza che il client chiami il metodo `MqttClient.disconnect` .

Per monitorare le connessioni, completare la pubblicazione "Last Will and Testament" con altre pubblicazioni per registrare le connessioni e le disconnessioni programmate.

Concetti correlati

Callback e sincronizzazione nelle applicazioni client MQTT
Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Ripulisci sessioni
Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Identificativo client

Token di consegna

Persistenza del messaggio nei client MQTT

Pubblicazioni
Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Qualità del servizio fornito da un client MQTT
Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

Pubblicazioni conservate e client MQTT
Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Sottoscrizione
Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Stringhe argomento e filtri argomento nei client MQTT
Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Persistenza del messaggio nei client MQTT

I messaggi di pubblicazione vengono resi persistenti se vengono inviati con una QoS (quality of service) di "almeno una volta" o "esattamente una volta". È possibile implementare il proprio meccanismo di persistenza sul client oppure utilizzare il meccanismo di persistenza predefinito fornito con il client. La persistenza funziona in entrambe le direzioni, per le pubblicazioni inviate al o dal client.

In MQTT, la persistenza del messaggio ha due aspetti: il modo in cui il messaggio viene trasferito e se viene accodato in IBM MessageSight e IBM WebSphere MQ come messaggio persistente.

1. Il client MQTT accoppia la persistenza del messaggio con la QoS (quality of service). A seconda della qualità del servizio scelta per un messaggio, il messaggio viene reso persistente. La persistenza del messaggio è necessaria per implementare la QoS (quality of service) richiesta.

   Se si specifica "al massimo una volta", `QoS=0`, il client elimina il messaggio non appena viene pubblicato. Se si verifica un errore nell'elaborazione upstream del messaggio, il messaggio non viene inviato di nuovo. Anche se il client rimane attivo, il messaggio non viene inviato di nuovo. Il funzionamento dei messaggi `QoS=0` è lo stesso dei messaggi non persistenti veloci IBM WebSphere MQ .
   Se un messaggio viene pubblicato da un client con QoS di 1 o 2, viene reso persistente. Il messaggio viene memorizzato localmente e viene eliminato solo dal client quando non è più necessario per garantire la consegna "almeno una volta", `QoS=1`o "esattamente una volta", `QoS=2`.
2. Se un messaggio è contrassegnato come QoS 1 o 2, viene accodato in IBM MessageSight e IBM WebSphere MQ come messaggio persistente. Se è contrassegnato come `QoS=0`, viene accodato in IBM MessageSight e IBM WebSphere MQ come messaggio non persistente. In IBM WebSphere MQ i messaggi non persistenti vengono trasferiti tra gestori code "esattamente una volta", a meno che il canale dei messaggi non abbia l'attributo `NPMSPEED` impostato su `FAST`.

Una pubblicazione persistente viene memorizzata sul cliente fino a quando non viene ricevuta da un'applicazione client. Per `QoS=2`, la pubblicazione viene eliminata dal client quando il callback dell'applicazione restituisce il controllo. Per `QoS=1` l'applicazione potrebbe ricevere nuovamente la pubblicazione, se si verifica un errore. Per `QoS=0`, il callback riceve la pubblicazione non più di una volta. Potrebbe non ricevere la pubblicazione se si verifica un errore o se il client è disconnesso al momento della pubblicazione.

Quando si sottoscrive un topic, è possibile ridurre il QoS con cui il sottoscrittore riceve i messaggi in modo che corrisponda alle sue capacità di persistenza. Le pubblicazioni create in un QoS superiore vengono inviate con il QoS più elevato richiesto dal sottoscrittore (subscriber).

Memorizzazione dei messaggi

L'implementazione dell'archiviazione dei dati su piccoli dispositivi varia molto. Il modello di salvataggio temporaneo dei messaggi persistenti nell'archivio gestito dal client MQTT potrebbe essere troppo lento o richiedere troppa memoria. Nei dispositivi mobili, il sistema operativo mobile potrebbe fornire un servizio di archiviazione ideale per i messaggi MQTT .

Per fornire flessibilità nel soddisfare i vincoli delle periferiche di piccole dimensioni, il client MQTT ha due interfacce di persistenza. Le interfacce definiscono le operazioni coinvolte nella memorizzazione dei messaggi persistenti. Le interfacce sono descritte nella documentazione API per Client MQTT per Java. Per i collegamenti alla documentazione API client per le librerie client MQTT, vedere Riferimento di programmazione client MQTT. È possibile implementare le interfacce per adattarsi a una periferica. Il client MQTT in esecuzione su Java SE ha un'implementazione predefinita delle interfacce che memorizzano i messaggi persistenti nel filesystem. Utilizza il pacchetto `java.io` . Il client dispone anche di una implementazione predefinita per Java ME, `MqttDefaultMIDPPersistence`.

Classi di persistenza

`MqttClientPersistence`
Passare un'istanza dell'implementazione di `MqttClientPersistence` al client MQTT come parametro del costruttore `MqttClient` . Se si omette il parametro `MqttClientPersistence` dal costruttore `MqttClient` , il client MQTT memorizza i messaggi persistenti utilizzando la classe `MqttDefaultFilePersistence` o `MqttDefaultMIDPPersistence`.

`MqttPersistable`
`MqttClientPersistence` ottiene e inserisce `MqttPersistable` oggetti utilizzando una chiave di memoria. È necessario fornire un'implementazione di `MqttPersistable` e

l'implementazione di `MqttClientPersistence` se non si utilizza `MqttDefaultFilePersistence` o `MqttDefaultMIDPPersistence`.

`MqttDefaultFilePersistence`

Il client MQTT fornisce la classe `MqttDefaultFilePersistence` . Se si crea un'istanza di `MqttDefaultFilePersistence` nell'applicazione client, è possibile fornire la directory per memorizzare i messaggi persistenti come parametro del costruttore `MqttDefaultFilePersistence` .

In alternativa, il client MQTT può creare un'istanza di `MqttDefaultFilePersistence` e inserire i file in una directory predefinita. Il nome della directory è *`client identifier`*`-tcp` *`hostname portnumber`*. `"\", "\\", "/", ":"` e `" "` vengono rimossi dalla stringa del nome directory.

Il percorso della directory è il valore della proprietà di sistema `rcp.data`. Se `rcp.data` non è impostato, il percorso è il valore della proprietà di sistema `usr.data`.

`rcp.data` è una proprietà associata all'installazione di OSGi o Eclipse Rich Client Platform (RCP).

`usr.data` è la directory in cui è stato avviato il comando Java che ha avviato l'applicazione.

`MqttDefaultMIDPPersistence`

`MqttDefaultMIDPPersistence` ha un costruttore predefinito e nessun parametro. Utilizza il package `javax.microedition.rms.RecordStore` per memorizzare i messaggi.

Concetti correlati

Callback e sincronizzazione nelle applicazioni client MQTT
Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Ripulisci sessioni
Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Identificativo client

Token di consegna

Ultimo testamento e pubblicazione testamentaria
Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Pubblicazioni
Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Qualità del servizio fornito da un client MQTT
Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

Pubblicazioni conservate e client MQTT
Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Sottoscrizione

Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Stringhe argomento e filtri argomento nei client MQTT
Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Pubblicazioni

Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Un `MqttMessage` ha un array di byte come payload. Cercare di mantenere i messaggi il più piccoli possibile. La lunghezza massima del messaggio consentita dal protocollo di MQTT è 250 MB.

Di solito, un programma client MQTT utilizza `java.lang.String` o `java.lang.StringBuffer` per manipolare il contenuto del messaggio. Per comodità, la classe `MqttMessage` ha un metodo `toString` per convertire il suo payload in una stringa. Per creare il payload dell'array di byte da un `java.lang.String` o `java.lang.StringBuffer`, utilizzare il metodo `getBytes` .

Il metodo `getBytes` converte una stringa nella serie di caratteri predefinita per la piattaforma. La serie di caratteri predefinita è generalmente `UTF-8`. Le pubblicazioni MQTT che contengono solo testo sono generalmente codificate in `UTF-8`. Utilizzare il metodo `getBytes("UTF8")` per sovrascrivere la serie di caratteri predefinita.

In IBM WebSphere MQ, una pubblicazione MQTT viene ricevuta come messaggio `jms - bytes` . Il messaggio include una cartella `MQRFH2` contenente una cartella `<mqtt>`e una `<mqps>` . La cartella `<mqtt>` contiene `clientId` e `qos`, ma questo contenuto potrebbe cambiare in futuro.

Un `MqttMessage` ha tre attributi aggiuntivi: QoS (quality of service), se è conservato e se è un duplicato. L'indicatore duplicato viene impostato solo se la qualità del servizio è "almeno una volta" o "esattamente una volta". Se il messaggio è stato inviato in precedenza e non è stato riconosciuto abbastanza rapidamente dal client MQTT , il messaggio viene inviato di nuovo, con l'attributo duplicato impostato su `true`.

Pubblicazione

Per creare una pubblicazione in una applicazione client MQTT , creare un `MqttMessage`. Imposta il payload, la qualità del servizio e se viene conservato e richiama il metodo `MqttTopic.publish(MqttMessage message)` ; viene restituito `MqttDeliveryToken` e il completamento della pubblicazione è asincrono.

In alternativa, il client MQTT può creare un oggetto messaggio temporaneo dai parametri sul metodo `MqttTopic.publish(byte [] payload, int qos, boolean retained)` quando crea una pubblicazione.

Se la pubblicazione ha una qualità del servizio "almeno una volta" o "esattamente una volta", `QoS=1` o `QoS=2`, il client MQTT richiama l'interfaccia `MqttClientPersistence` . Richiama `MqttClientPersistence` per memorizzare il messaggio prima di restituire un token di consegna all'applicazione.

L'applicazione può scegliere di bloccare fino a quando il messaggio non viene consegnato al server, utilizzando il metodo `MqttDeliveryToken.waitForCompletion` . In alternativa, l'applicazione può continuare senza bloccare. Se si desidera controllare se le pubblicazioni vengono consegnate, senza bloccare, registrare un'istanza di una classe di callback che implementa `MqttCallback` con il client MQTT . Il client MQTT richiama il metodo `MqttCallback.deliveryComplete` non appena la

pubblicazione è stata consegnata. A seconda della qualità del servizio, la distribuzione potrebbe essere quasi immediata per QoS=0o potrebbe richiedere del tempo per QoS=2.

Utilizzare il metodo `MqttDeliveryToken.isComplete` per eseguire il polling se la consegna è completa. Mentre il valore di `MqttDeliveryToken.isComplete` è `false`, è possibile richiamare `MqttDeliveryToken.getMessage` per ottenere il contenuto del messaggio. Se il risultato della chiamata `MqttDeliveryToken.isComplete` è `true`, il messaggio è stato eliminato e la chiamata `MqttDeliveryToken.getMessage` genera un'eccezione di puntatore null. Non esiste alcuna sincronizzazione integrata tra `MqttDeliveryToken.getMessage` e `MqttDeliveryToken.isComplete`.

Se il client si disconnette prima di ricevere tutti i token di consegna in sospeso, una nuova istanza del client può interrogare i token di consegna in sospeso prima della connessione. Fino a quando il client non si connette, non vengono completate nuove consegne ed è sicuro chiamare `MqttDeliveryToken.getMessage`. Utilizzare il metodo `MqttDeliveryToken.getMessage` per scoprire quali pubblicazioni non sono state consegnate. I token di consegna in sospeso vengono eliminati se ci si connette con `MqttConnectOptions.cleanSession` impostato sul valore predefinito, `true`.

Sottoscrizione

Un gestore code o IBM MessageSight è responsabile della creazione di pubblicazioni da inviare a un sottoscrittore MQTT . Il gestore code verifica se il filtro argomenti in una sottoscrizione creata da un client MQTT corrisponde alla stringa argomenti in una pubblicazione. La corrispondenza può essere una corrispondenza esatta oppure può includere caratteri jolly. Prima che la pubblicazione venga inoltrata al sottoscrittore dal gestore code, il gestore code controlla gli attributi argomento associati alla pubblicazione. Segue la procedura di ricerca descritta in Sottoscrizione mediante una stringa di argomenti contenente caratteri jolly per identificare se un oggetto argomento di gestione concede all'utente l'autorizzazione alla sottoscrizione.

Quando il client MQTT riceve una pubblicazione con QoS (quality of service) "almeno una volta", richiama il metodo `MqttCallback.messageArrived` per elaborare la pubblicazione. Se la qualità del servizio della pubblicazione è "esattamente una volta", QoS=2, il client MQTT richiama l'interfaccia `MqttClientPersistence` per memorizzare il messaggio quando viene ricevuto. Viene quindi chiamato `MqttCallback.messageArrived`.

Concetti correlati

Callback e sincronizzazione nelle applicazioni client MQTT
Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Ripulisci sessioni
Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Identificativo client

Token di consegna

Ultimo testamento e pubblicazione testamentaria
Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Persistenza del messaggio nei client MQTT

Qualità del servizio fornito da un client MQTT

Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

Pubblicazioni conservate e client MQTT
Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Sottoscrizione
Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Stringhe argomento e filtri argomento nei client MQTT
Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Qualità del servizio fornito da un client MQTT

Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

La qualità del servizio di una pubblicazione è un attributo di `MqttMessage`. È impostato dal metodo `MqttMessage.setQos`.

Il metodo `MqttClient.subscribe` può ridurre la qualità del servizio applicato alle pubblicazioni inviate a un client su un argomento. La qualità del servizio di una pubblicazione inoltrata a un sottoscrittore potrebbe essere diversa dalla qualità del servizio della pubblicazione. Il valore inferiore dei due valori viene utilizzato per inoltrare una pubblicazione.

Al massimo una volta
`QoS=0`

Il messaggio viene consegnato al massimo una volta oppure non viene consegnato. Il suo recapito sulla rete non è riconosciuto.

Il messaggio non è memorizzato. Se il client è disconnesso o se si verifica un errore nel server, il messaggio potrebbe andare perso.

`QoS=0` è la modalità di trasferimento più veloce. A volte si chiama "fuoco e dimenticate".

Il protocollo MQTT non richiede che i server inoltrino le pubblicazioni su `QoS=0` a un client. Se il client è disconnesso nel momento in cui il server riceve la pubblicazione, la pubblicazione potrebbe essere eliminata, a seconda del server. Il servizio di telemetria (MQXR) non elimina i messaggi inviati con `QoS=0`. Vengono memorizzati come messaggi non persistenti e vengono eliminati solo se il gestore code viene arrestato.

Almeno una volta
`QoS=1`

`QoS=1` è la modalità predefinita di trasferimento.

Il messaggio viene sempre consegnato almeno una volta. Se il mittente non riceve un riconoscimento, il messaggio viene inviato di nuovo con l'indicatore DUP impostato fino a quando non viene ricevuto un riconoscimento. Come risultato, il destinatario può essere inviato lo stesso messaggio più volte e potrebbe elaborarlo più volte.

Il messaggio deve essere memorizzato localmente sul mittente e sul destinatario fino a quando non viene elaborato.

Il messaggio viene eliminato dal destinatario dopo che questo ha elaborato il messaggio. Se il destinatario è un broker, il messaggio viene pubblicato per i relativi sottoscrittori. Se il destinatario

è un client, il messaggio viene consegnato all'applicazione del sottoscrittore. Una volta eliminato il messaggio, il destinatario invia un riconoscimento al mittente.

Il messaggio viene eliminato dal mittente dopo aver ricevuto un riconoscimento dal destinatario.

Esattamente una volta

QoS=2

Il messaggio viene sempre consegnato esattamente una volta.

Il messaggio deve essere memorizzato localmente sul mittente e sul destinatario fino a quando non viene elaborato.

QoS=2 è la modalità di trasferimento più sicura ma più lenta. Richiede almeno due coppie di trasmissioni tra il mittente e il ricevente prima che il messaggio venga eliminato dal mittente. Il messaggio può essere elaborato dal destinatario dopo la prima trasmissione.

Nella prima coppia di trasmissioni, il mittente trasmette il messaggio e riceve il riconoscimento dal destinatario che ha memorizzato il messaggio. Se il mittente non riceve un riconoscimento, il messaggio viene inviato di nuovo con l'indicatore DUP impostato fino a quando non viene ricevuto un riconoscimento.

Nella seconda coppia di trasmissioni, il mittente comunica al ricevente che può completare l'elaborazione del messaggio, `"PUBREL"`. Se il mittente non riceve un riconoscimento del messaggio `"PUBREL"` , il messaggio `"PUBREL"` viene inviato di nuovo fino a quando non viene ricevuto un riconoscimento. Il mittente elimina il messaggio salvato quando riceve il riconoscimento al messaggio `"PUBREL"`

Il destinatario può elaborare il messaggio nella prima o nella seconda fase, purché non rielabori il messaggio. Se il destinatario è un broker, pubblica il messaggio ai sottoscrittori. Se il destinatario è un client, consegna il messaggio all'applicazione del sottoscrittore. Il destinatario invia un messaggio di completamento al mittente che ha terminato l'elaborazione del messaggio.

Concetti correlati

Callback e sincronizzazione nelle applicazioni client MQTT

Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Ripulisci sessioni

Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Identificativo client

Token di consegna

Ultimo testamento e pubblicazione testamentaria

Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Persistenza del messaggio nei client MQTT

Pubblicazioni

Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Pubblicazioni conservate e client MQTT

Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Sottoscrizione

Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Stringhe argomento e filtri argomento nei client MQTT
Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Pubblicazioni conservate e client MQTT

Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Utilizzare il metodo `MqttMessage.setRetained` per specificare se una pubblicazione su un argomento viene conservata o meno.

Per cancellare una pubblicazione conservata in IBM WebSphere MQ, eseguire il comando MQSC **CLEAR TOPICSTR** .

Se si crea una pubblicazione con un payload null, la pubblicazione vuota viene inoltrata ai sottoscrittori. Altri broker MQTT potrebbero non inoltrare una pubblicazione vuota ai sottoscrittori.

Se si pubblica una pubblicazione non conservata in un argomento che ha una pubblicazione conservata, la pubblicazione conservata non viene influenzata. I sottoscrittori correnti ricevono la nuova pubblicazione. I nuovi sottoscrittori ricevono prima la pubblicazione conservata, quindi le nuove pubblicazioni.

Quando si crea o si aggiorna una pubblicazione conservata, inviare la pubblicazione con un QoS o 1 o 2. Se lo si invia con un QoS pari a 0, IBM WebSphere MQ crea una pubblicazione conservata non persistente. La pubblicazione non viene conservata se il gestore code viene arrestato.

Utilizzare le pubblicazioni conservate per registrare l'ultimo valore di una misurazione. I nuovi sottoscrittori all'argomento conservato ricevono immediatamente il valore più recente della misurazione. Se non viene eseguita alcuna nuova misurazione dall'ultima sottoscrizione del sottoscrittore all'argomento della pubblicazione e se il sottoscrittore effettua nuovamente la sottoscrizione, il sottoscrittore riceve nuovamente la pubblicazione conservata più recente sull'argomento.

Concetti correlati

Callback e sincronizzazione nelle applicazioni client MQTT
Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Ripulisci sessioni
Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Identificativo client

Token di consegna

Ultimo testamento e pubblicazione testamentaria
Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Persistenza del messaggio nei client MQTT

Pubblicazioni
Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Qualità del servizio fornito da un client MQTT
Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

Sottoscrizione
Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Stringhe argomento e filtri argomento nei client MQTT
Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Sottoscrizione

Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Creare sottoscrizioni utilizzando i metodi `MqttClient.subscribe` , passando uno o più filtri argomento e parametri QoS (quality of service). Il parametro QoS (quality of service) imposta la qualità massima del servizio che il sottoscrittore è pronto a utilizzare per ricevere un messaggio. I messaggi inviati a questo client non possono essere consegnati con una QoS (quality of service) superiore. La QoS (quality of service) è impostata sul valore inferiore del valore originale quando il messaggio è stato pubblicato e sul livello specificato per la sottoscrizione. La qualità del servizio predefinita per la ricezione dei messaggi è `QoS=1`, almeno una volta.

La richiesta di sottoscrizione viene inviata con `QoS=1`.

Le pubblicazioni vengono ricevute da un sottoscrittore quando il client MQTT richiama il metodo `MqttCallback.messageArrived` . Il metodo `messageArrived` inoltra anche la stringa di argomenti con cui il messaggio è stato pubblicato al sottoscrittore.

È possibile rimuovere una sottoscrizione o una serie o sottoscrizioni utilizzando i metodi `MqttClient.unsubscribe` .

Un comando WebSphere MQ può rimuovere una sottoscrizione. Elencare le sottoscrizioni utilizzando WebSphere MQ Explorer oppure utilizzando i comandi **runmqsc** o PCF. Tutte le sottoscrizioni dei client MQTT vengono denominate. Viene fornito un nome del formato: *`ClientIdentifier:Topic name`*

Se si utilizza il valore predefinito `MqttConnectOptions` o si imposta `MqttConnectOptions.cleanSession` su `true` prima di connettere il client, tutte le sottoscrizioni precedenti del client vengono rimosse alla connessione del client. Qualsiasi nuova sottoscrizione effettuata dal client durante la sessione viene rimossa quando si disconnette.

Se si imposta `MqttConnectOptions.cleanSession` su `false` prima della connessione, qualsiasi sottoscrizione creata dal client viene aggiunta a tutte le sottoscrizioni esistenti per il client prima della sua connessione. Tutte le sottoscrizioni restano attive alla disconnessione del client.

Un altro modo per comprendere come l'attributo `cleanSession` influisce sulle sottoscrizioni è considerarlo come un attributo modale. Nella sua modalità predefinita, `cleanSession=true`, il client crea sottoscrizioni e riceve pubblicazioni solo nell'ambito della sessione. Nella modalità alternativa,

`cleanSession=false`, le sottoscrizioni sono durevoli. Il client può connettersi e disconnettersi e le sue sottoscrizioni rimangono attive. Quando il client si riconnette, riceve tutte le pubblicazioni non consegnate. Mentre è connesso, può modificare l'insieme di sottoscrizioni attive per suo conto.

È necessario impostare la modalità `cleanSession` prima della connessione; la modalità è valida per l'intera sessione. Per modificarne l'impostazione, è necessario disconnettere e riconnettere il client. Se si modifica la modalità da `cleanSession=false` a `cleanSession=true`, tutte le sottoscrizioni precedenti destinate al client e le eventuali pubblicazioni non ricevute vengono eliminate.

Le pubblicazioni che corrispondono alle sottoscrizioni attive vengono inviate al client non appena vengono pubblicate. Se il client è disconnesso, vengono inviati al client se si riconnette allo stesso server con lo stesso identificativo client e `MqttConnectOptions.cleanSession` impostato su `false`.

Le sottoscrizioni per un particolare cliente vengono identificate dall'identificativo client. È possibile riconnettere il client da un dispositivo client differente allo stesso server e continuare con la stessa sottoscrizione e ricevere pubblicazioni non consegnate.

Concetti correlati

Callback e sincronizzazione nelle applicazioni client MQTT
Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Ripulisci sessioni
Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Identificativo client

Token di consegna

Ultimo testamento e pubblicazione testamentaria
Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Persistenza del messaggio nei client MQTT

Pubblicazioni
Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Qualità del servizio fornito da un client MQTT
Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

Pubblicazioni conservate e client MQTT
Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Stringhe argomento e filtri argomento nei client MQTT
Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Stringhe argomento e filtri argomento nei client MQTT

Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

Le stringhe di argomenti vengono utilizzate per inviare pubblicazioni ai sottoscrittori. Creare una stringa di argomenti utilizzando il metodo `MqttClient.getTopic(java.lang.String topicString)`.

I filtri argomento vengono utilizzati per sottoscrivere argomenti e ricevere pubblicazioni. I filtri argomento possono contenere caratteri jolly. Con i caratteri jolly, è possibile sottoscrivere più argomenti. Creare un filtro argomenti utilizzando un metodo di sottoscrizione; ad esempio, `MqttClient.subscribe(java.lang.String topicFilter)`.

Stringhe argomento

La sintassi di una stringa di argomenti IBM WebSphere MQ è descritta in Stringhe di argomenti. La sintassi delle stringhe di argomenti MQTT è descritta nella classe `MqttClient` nella documentazione API per Client MQTT per Java. Per i collegamenti alla documentazione API client per le librerie client MQTT, vedere Riferimento di programmazione client MQTT.

La sintassi di ciascun tipo di stringa argomento è quasi identica. Ci sono quattro differenze minori:

1. Le stringhe degli argomenti inviate a IBM WebSphere MQ dai client MQTT devono seguire le convenzioni per i nomi dei gestori code. In particolare, le stringhe di argomento non possono contenere trattini.
2. Le lunghezze massime differiscono. Le stringhe di argomenti IBM WebSphere MQ sono limitate a 10.240 caratteri. Un client MQTT può creare stringhe di argomenti fino a un massimo di 65535 byte.
3. Una stringa argomento creata da un client MQTT non può contenere un carattere null.
4. In WebSphere Message Broker, un livello di argomento null, `'...//...'` non era valido. I livelli di argomento null sono supportati da IBM WebSphere MQ.

A differenza della pubblicazione / sottoscrizione IBM WebSphere MQ , il protocollo `mqttv3` non ha un concetto di oggetto argomento di gestione. Non è possibile costruire una stringa argomento da un oggetto argomento e una stringa argomento. Tuttavia, una stringa di argomenti viene associata a un argomento di amministrazione in IBM WebSphere MQ. Il controllo accessi associato all'argomento di gestione determina se una pubblicazione viene pubblicata nell'argomento o scartata. Gli attributi che vengono applicati a una pubblicazione quando viene inoltrata ai sottoscrittori, sono influenzati dagli attributi dell'argomento di gestione.

Filtri argomento

La sintassi di un filtro argomento IBM WebSphere MQ è descritta in Schema dei caratteri jolly basati sugli argomenti. La sintassi dei filtri argomento che puoi creare con un client MQTT è descritta nella classe `MqttClient` nella documentazione API per Client MQTT per Java. Per i collegamenti alla documentazione API client per le librerie client MQTT, vedere Riferimento di programmazione client MQTT.

La sintassi di ciascun tipo di filtro argomento è quasi identica. L'unica differenza sta nel modo in cui i broker MQTT interpretano un filtro argomento. In WebSphere Message Broker V6, un carattere jolly multilivello può essere utilizzato solo alla fine di un filtro argomenti. In IBM WebSphere MQ, un carattere jolly multilivello può essere utilizzato a qualsiasi livello nella struttura ad albero degli argomenti; ad esempio USA/#/`Dutchess County`.

Concetti correlati

Callback e sincronizzazione nelle applicazioni client MQTT
Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

Ripulisci sessioni
Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

Identificativo client

Token di consegna

Ultimo testamento e pubblicazione testamentaria
Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

Persistenza del messaggio nei client MQTT

Pubblicazioni
Le pubblicazioni sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT può creare pubblicazioni da inviare a IBM WebSphere MQe sottoscrivere argomenti su IBM WebSphere MQ MQ per ricevere pubblicazioni.

Qualità del servizio fornito da un client MQTT
Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

Pubblicazioni conservate e client MQTT
Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

Sottoscrizione
Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

Concetti di programmazione del client C

Le differenze tra il client C e Java per la versione 3.1 di MQ Telemetry Transport sono descritte in questo argomento. L'argomento integra i concetti del cliente e le informazioni di riferimento C.

L'argomento è organizzato nello stesso modo di “Concetti di programmazione del client MQTT” a pagina 527. Ogni intestazione corrisponde a un argomento in *WebSphere(r) MQ Concetti di programmazione del client Telemetry Transport*. Le sezioni descrivono le differenze tra il client C e il client Java. Piccole differenze nelle firme tra i metodi Java e le funzioni C non vengono descritte.

Il client C viene spesso utilizzato per implementare un adattatore leggero tra un dispositivo di telemetria e il daemon WebSphere MQ Telemetry per i dispositivi. Il daemon è comunemente utilizzato come concentratore di rete tra i dispositivi di telemetria molto leggeri e il servizio di telemetria (MQXR).

Il daemon WebSphere MQ Telemetry per dispositivi è anche un client C e vengono descritte le differenze nel suo comportamento dal servizio di telemetria (MQXR). Il daemon non fornisce un'implementazione di JAAS o SSL per i client che si collegano ad essa.

`mqttclient.dll` e `mqttclient.lib` sono le librerie Windows a 32 bit che contengono funzioni client per l'implementazione C del protocollo MQ Telemetry Transport versione 3.1 . Le librerie Linux a 32 bit sono `libmqttclient.so` e `libmqttclient.a`. Due file di intestazione contengono la funzione e altre dichiarazioni necessarie alle applicazioni client: `MQTTClient.h` e `MQTTClientPersistence.h`. Questi file vengono forniti con l'installazione di WebSphere MQ Telemetry.

Per sviluppare ed eseguire un client MQ Telemetry Transport è necessario copiare questi file sul dispositivo client. A differenza dei client WebSphere MQ , non è necessario installare un runtime client separato.

Consultare le condizioni di licenza associate con la funzione WebSphere MQ Telemetry che regola la connessione dei client MQ Telemetry Transport a WebSphere MQ e il daemon WebSphere MQ Telemetry per i dispositivi.

Il client C è un'implementazione di riferimento della versione 3.1 di MQ Telemetry Transport. È possibile implementare i propri client in diverse lingue adatte a diverse piattaforme di dispositivi. Fare riferimento a MQ Telemetry Transport format and protocol per i dettagli.

L'identificativo del client MQTT

"Identificativo client" a pagina 532	L'identificativo client è una stringa di 23 byte che identifica un client MQTT . Ogni ID deve essere univoco a un solo client connesso per volta. L'identificativo deve contenere solo caratteri validi in un nome gestore code. All'interno di questi vincoli, è possibile utilizzare qualsiasi stringa di identificazione. È importante disporre di una procedura per l'assegnazione di identificativi client e di un mezzo per la configurazione di un client con l'identificativo scelto.

- Nessuna differenza.

Pubblicazioni

"Pubblicazioni" a pagina 539	sono istanze di `MqttMessage` associate a una stringa di argomento. Il client MQTT

- La funzione di callback non viene richiamata per le pubblicazioni con "fire and forget", `QoS=0`, QoS (quality of service).

Token di consegna

"Token di consegna" a pagina 533	Quando un client pubblica un argomento, viene creato un nuovo token di consegna. Utilizzare il token di consegna per monitorare la consegna di una pubblicazione o per bloccare l'applicazione client fino al completamento della consegna.

- Un token di consegna è un `int`. Ha un typedef di `MQTTClient_deliveryToken`
- La funzione di callback non viene richiamata per le pubblicazioni con "fire and forget", `QoS=0`, QoS (quality of service).

Pubblicazioni conservate

"Pubblicazioni conservate e client MQTT" a pagina 543	Se si crea una sottoscrizione a un argomento che ha una pubblicazione conservata, la pubblicazione conservata più recente sull'argomento viene immediatamente inoltrata all'utente.

- I messaggi conservati vengono salvati nel daemon solo se la persistenza è configurata; consultare Salvataggio delle sottoscrizioni e dei messaggi conservati.

  Per WebSphere MQ, la qualità del servizio determina se un messaggio conservato viene salvato in modo permanente. Se un client è collegato al servizio di telemetria, i messaggi conservati con "fire and forget", la QoS (quality of service) `QoS=0` vengono eliminati, se il gestore code viene arrestato.

Sottoscrizione

“Sottoscrizione” a pagina 544	Creare sottoscrizioni per registrare un interesse negli argomenti di pubblicazione utilizzando un filtro argomenti. Un client può creare più sottoscrizioni o una sottoscrizione contenente un filtro argomenti che utilizza caratteri jolly, per registrare un interesse in più argomenti. Le pubblicazioni sugli argomenti corrispondenti ai filtri vengono inviati al client. Le sottoscrizioni possono rimanere attive mentre un client è disconnesso. Le pubblicazioni vengono inviate al client quando si riconnette.

- Le sottoscrizioni durevoli vengono salvate nel daemon solo se è configurata la persistenza; consultare Salvataggio delle sottoscrizioni e dei messaggi conservati.
- Le pubblicazioni possono essere ricevute in modo sincrono. Richiamare la funzione `MQTTClient_receive` .

Callback e sincronizzazione

“Callback e sincronizzazione nelle applicazioni client MQTT” a pagina 527	Il modello di programmazione client MQTT utilizza ampiamente i thread. I thread disaccoppiano un'applicazione client MQTT , per quanto possibile, dai ritardi nella trasmissione dei messaggi verso e dal server. Le pubblicazioni, i token di consegna e gli eventi di connessione persi vengono consegnati ai metodi in una classe di callback che implementa `MqttCallback`.

- L'operazione di sincronizzazione nel client C è modale. La chiamata di `MQTTClient_setCallback` mette il client in modalità asincrona.
- In modalità sincrona, il client delle applicazioni deve volontariamente cedere il controllo in modo che il client MQTT possa elaborare i riconoscimenti ed emettere i ping MQTT per mantenere attiva la rete. Rendere il controllo richiamando `MQTTClient_receive` o `MQTTClient_yield`.

Stringhe di argomenti e filtri

“Stringhe argomento e filtri argomento nei client MQTT” a pagina 546	Le stringhe argomento e filtri argomento vengono utilizzati per la pubblicazione e la sottoscrizione. La sintassi delle stringhe argomento e dei filtri nei client MQTT è in gran parte la stessa delle stringhe argomento in IBM WebSphere MQ.

- Il daemon WebSphere MQ Telemetry per i dispositivi gestisce il carattere jolly a più livelli *#* in modo diverso da WebSphere MQ v7. */#* deve essere gli ultimi due caratteri nella stringa di filtro affinché *#* si comporti come un carattere jolly. In WebSphere MQ v7, `../#/..` è un utilizzo valido del carattere jolly multilivello. Il daemon WebSphere MQ Telemetry per i dispositivi tratta il carattere jolly multilivello come WebSphere MQ Broker v6.

QOS (quality of service)

“Qualità del servizio fornito da un client MQTT” a pagina 541	Un client MQTT fornisce tre QoS (quality of service) per la distribuzione di pubblicazioni a WebSphere MQ e al client MQTT : "al più una volta", "almeno una volta" e "esattamente una volta". Quando un client MQTT invia una richiesta a WebSphere MQ per creare una sottoscrizione, la richiesta viene inviata con la QoS "almeno una volta".

- Nessuna differenza.

Persistenza messaggio

“Persistenza del messaggio nei client MQTT” a pagina 536	I messaggi di pubblicazione vengono resi persistenti se vengono inviati con una QoS (quality of service) di "almeno una volta" o "esattamente una volta". È possibile implementare il proprio meccanismo di persistenza sul client oppure utilizzare il meccanismo di persistenza predefinito fornito con il client. La persistenza funziona in entrambe le direzioni, per le pubblicazioni inviate al o dal client.

- A causa delle differenze di collegamento del linguaggio, impostare il meccanismo di persistenza del messaggio nel client C nel modo seguente. Richiamare il client MQTT C con una delle tre opzioni impostate come quarto parametro per `MQTTClient_create`:

  `MQTTCLIENT_PERSISTENCE_DEFAULT`
  Una persistenza basata su file, i cui dettagli sono specifici della piattaforma client.

  `MQTTCLIENT_PERSISTENCE_NONE`
  I dati vengono conservati solo in memoria e persi quando il client si arresta. Il daemon WebSphere MQ Telemetry per dispositivi supporta solo questa opzione.

  `MQTTCLIENT_PERSISTENCE_USER`
  È possibile sviluppare funzioni per implementare il proprio meccanismo di persistenza. Passa una struttura, `MQTTClient_persistence` contenente puntatori alle tue funzioni sulla chiamata `MQTTClient_create` . Per i dettagli, leggere le informazioni di riferimento del client MQTT C.

Ripulisci sessioni

“Ripulisci sessioni” a pagina 531	Il client MQTT e il servizio di telemetria (MQXR) mantengono le informazioni sullo stato della sessione. Le informazioni sullo stato vengono utilizzate per assicurare "almeno una volta" e "esattamente una volta" la consegna e "esattamente una volta" la ricezione delle pubblicazioni. Lo stato della sessione include anche le sottoscrizioni create da un client MQTT. È possibile scegliere di eseguire un client MQTT con o senza mantenere le informazioni sullo stato tra le sessioni. Modificare la modalità di pulizia della sessione impostando `MqttConnectOptions.cleanSession` prima della connessione.

- Nessuna differenza.

Ultime indicazioni

“Ultimo testamento e pubblicazione testamentaria” a pagina 535	Se una connessione client MQTT termina in modo imprevisto, è possibile configurare WebSphere MQ Telemetry per inviare una pubblicazione "Last Will and Testament". Predefinire il contenuto della pubblicazione e l'argomento a cui inviarla. L'"ultima volontà e testamento" è una proprietà di connessione. Crearlo prima di collegare il cliente.

- Nessuna differenza.

Gestione degli errori del programma

Queste informazioni spiegano gli errori associati alle chiamate MQI delle applicazioni quando effettua una chiamata o quando il suo messaggio viene consegnato alla destinazione finale.

Quando possibile, il gestore code restituisce eventuali errori non appena viene effettuata una chiamata MQI. Questi sono *errori determinati localmente*.

Quando si inviano messaggi a una coda remota, gli errori potrebbero non essere evidenti quando viene effettuata la chiamata MQI. In questo caso, il gestore code che identifica gli errori li riporta inviando un altro messaggio al programma di origine. Si tratta di *errori determinati in remoto*.

Errori determinati localmente

Informazioni sugli errori determinati localmente che includono: errore su una chiamata MQI, interruzioni del sistema e messaggi contenenti dati non corretti.

Le tre cause più comuni di errori che il gestore code può segnalare immediatamente sono:

- Errore di una chiamata MQI; ad esempio, perché una coda è piena
- Un'interruzione dell'esecuzione di alcune parti del sistema da cui dipende l'applicazione; ad esempio, il gestore code
- Messaggi che contengono dati che non possono essere elaborati correttamente

Se si sta utilizzando la funzione di inserimento asincrono, gli errori non vengono notificati immediatamente. Utilizzare la chiamata MQSTAT per richiamare le informazioni sullo stato delle precedenti operazioni di immissione asincrona.

Errore di una chiamata MQI

Il gestore code può segnalare immediatamente eventuali errori nella codifica di una chiamata MQI. Lo fa utilizzando una serie di codici di ritorno predefiniti. Questi sono divisi in codici di completamento e codici di errore.

Per mostrare se una chiamata ha esito positivo, il gestore code restituisce un *codice di completamento* quando la chiamata viene completata. Esistono tre codici di completamento, che indicano l'esito positivo, il completamento parziale e l'errore della chiamata. Il gestore code restituisce anche un *codice motivo* che indica il motivo per il completamento parziale o l'errore della chiamata.

I codici di completamento e motivo per ogni chiamata vengono elencati con la descrizione di tale chiamata in Codici di ritorno. Per informazioni più dettagliate, incluse le idee per un'azione correttiva, consultare:

- Codici motivo per tutte le altre piattaforme WebSphere MQ

Progetta i tuoi programmi per gestire tutti i codici di ritorno che possono derivare da ogni chiamata.

Interruzioni del sistema

L'applicazione potrebbe non essere a conoscenza di eventuali interruzioni se il gestore code a cui è connesso deve essere ripristinato da un malfunzionamento del sistema. Tuttavia, è necessario progettare l'applicazione per garantire che i dati non vadano persi se si verifica tale interruzione.

I metodi che è possibile utilizzare per assicurarsi che i propri dati rimangano congruenti dipendono dalla piattaforma su cui è in esecuzione il gestore code:

Sistemi UNIX, Linux e Windows
In questi ambienti, è possibile effettuare le chiamate MQPUT e MQGET nel solito modo, ma è necessario dichiarare i punti di sincronizzazione utilizzando le chiamate MQCMIT e MQBACK (consultare “Commit e backout delle unità di lavoro” a pagina 325). Nell'ambiente CICS , i comandi MQCMIT e MQBACK sono disabilitati, poiché è possibile effettuare le chiamate MQPUT e MQGET all'interno delle unità di lavoro gestite da CICS.

Utilizzare i messaggi persistenti per trasportare tutti i dati che non ci si può permettere di perdere. I messaggi persistenti vengono reintegrati nelle code se il gestore code deve eseguire il ripristino da un malfunzionamento. Con WebSphere MQ su sistemi UNIX, Linuxe Windows , una chiamata MQGET o MQPUT all'interno dell'applicazione avrà esito negativo al punto di riempire tutti i file di log, con il messaggio MQRC_RESOURCE_PROBLEM. Per ulteriori informazioni sui file di log sui sistemi AIX, HP-UX, Linux, Solaris e Windows , consultare Amministrazione.

Se il gestore code viene arrestato da un operatore mentre un'applicazione è in esecuzione, di solito viene utilizzata l'opzione di sospensione. Il gestore code entra in uno stato di sospensione in cui le applicazioni possono continuare a lavorare, ma devono terminare non appena possibile. Le applicazioni piccole e rapide possono probabilmente ignorare lo stato di sospensione e continuare fino a quando non terminano normalmente. Le applicazioni in esecuzione più a lungo, o quelle che attendono l'arrivo dei messaggi, devono utilizzare l'opzione *fail if quiescing* quando utilizzano le chiamate MQOPEN, MQPUT, MQPUT1e MQGET. Queste opzioni indicano che le chiamate hanno esito negativo quando il gestore code viene disattivato, ma l'applicazione potrebbe avere ancora tempo per terminare in modo pulito emettendo chiamate che ignorano lo stato di disattivazione. Tali applicazioni potrebbero anche eseguire il commit o il backout delle modifiche apportate e quindi terminare.

Se viene forzato l'arresto del gestore code (ovvero, l'arresto senza disattivazione), le applicazioni riceveranno il codice motivo MQRC_CONNECTION_BROKEN quando effettuano chiamate MQI. Uscire dall'applicazione o, in alternativa, sui sistemi UNIX, Linuxe Windows , emettere una chiamata MQDISC.

Messaggi contenenti dati non corretti

Quando si utilizzano le unità di lavoro nella propria applicazione, se un programma non può elaborare correttamente un messaggio che richiama da una coda, viene eseguito il backout della chiamata MQGET.

Il gestore code mantiene un conteggio (nel campo *BackoutCount* del descrizione del messaggio) del numero di volte che si verifica. Mantiene questo conteggio nel descrittore di ogni messaggio interessato. Questo conteggio può fornire informazioni utili sull'efficienza di una applicazione. I messaggi con conteggi di backout in aumento nel tempo vengono ripetutamente rifiutati; progettare l'applicazione in modo che analizzi i motivi e gestisca tali messaggi di conseguenza.

Su sistemi WebSphere MQ per Windows, UNIX e Linux , il conteggio di backout sopravvive sempre al riavvio del gestore code.

Utilizzo dei messaggi di report per la determinazione dei problemi

Il gestore code remoto non è in grado di riportare errori quali l'errore di inserimento di un messaggio in una coda quando si effettua la chiamata MQI, ma può inviare un messaggio di report per indicare come è stato elaborato il messaggio.

All'interno dell'applicazione è possibile creare messaggi di report (MQPUT) e selezionare l'opzione per riceverli (nel qual caso vengono inviati da un'altra applicazione o da un gestore code).

Creazione di messaggi di report

I messaggi di report consentono a un'applicazione di comunicare a un'altra applicazione che non può gestire il messaggio inviato.

Tuttavia, il campo *Report* deve essere inizialmente analizzato per determinare se l'applicazione che ha inviato il messaggio è interessata ad essere informata di eventuali problemi. Dopo aver determinato che è richiesto un messaggio di report, è necessario decidere:

- Se si desidera includere l'intero messaggio originale, solo i primi 100 byte di dati o nessuno dei messaggi originali.
- Cosa fare con il messaggio originale. È possibile eliminarlo o lasciarlo andare alla coda di messaggi non recapitabili.
- Se anche il contenuto dei campi *MsgId* e *CorrelId* è necessario.

Utilizzare il campo *Feedback* per indicare il motivo della generazione del messaggio di report. Inserire i messaggi di report in una coda di risposta dell'applicazione. Per ulteriori informazioni, consultare Feedback .

Richiesta e ricezione di messaggi di report (MQGET)

Quando si invia un messaggio a un'altra applicazione, non si viene informati di eventuali problemi a meno che non si completi il campo *Report* per indicare il feedback richiesto. Consultare Struttura del campo del report per le opzioni disponibili.

I gestori code inserisano sempre i messaggi di report nella coda di risposta di un'applicazione e si consiglia alle proprie applicazioni di fare lo stesso. Quando si utilizza la funzione del messaggio di report, specificare il nome della propria coda di risposta nel descrittore del messaggio; in caso contrario, la chiamata MQPUT non riesce.

L'applicazione deve contenere procedure che monitorano la coda di risposta ed elaborano i messaggi che arrivano su di essa. Tenere presente che un messaggio di report può contenere tutti i messaggi originali, i primi 100 byte del messaggio originale o nessuno dei messaggi originali.

Il gestore code imposta il campo *Feedback* del messaggio di report per indicare il motivo dell'errore; ad esempio, la coda di destinazione non esiste. I programmi dovrebbero fare lo stesso.

Per ulteriori informazioni sui messaggi di report, consultare “Messaggi di report” a pagina 11.

Errori determinati in remoto

Quando si inviano messaggi a una coda remota, anche quando il gestore code locale ha elaborato la chiamata MQI senza trovare un errore, altri fattori possono influenzare il modo in cui il messaggio viene gestito da un gestore code remoto.

Ad esempio, la coda di destinazione potrebbe essere piena o non esistere. Se il messaggio deve essere gestito da altri gestori code intermedi sull'instradamento alla coda di destinazione, uno di questi potrebbe rilevare un errore.

Problemi nella consegna di un messaggio

Quando una chiamata MQPUT ha esito negativo, è possibile provare a inserire nuovamente il messaggio nella coda, restituirlo al mittente o inserirlo nella coda di messaggi non recapitabili.

Ogni opzione ha i suoi meriti, ma è possibile che non si desideri provare a inserire di nuovo un messaggio se il motivo per cui MQPUT non è riuscito è che la coda di destinazione era piena. In questa istanza, l'inserimento nella coda di messaggi non recapitabili consente di consegnarla successivamente alla coda di destinazione corretta.

Riprova consegna messaggio

Prima che il messaggio venga inserito in una coda di messaggi non instradabili, un gestore code remoto tenta di inserire di nuovo il messaggio nella coda se gli attributi *MsgRetryCount* e *MsgRetryInterval* sono stati impostati per il canale o se esiste un programma di uscita per i tentativi da utilizzare (il cui nome è contenuto nel campo *MsgRetryExitId* dell'attributo del canale).

Se il campo *MsgRetryExitId* è vuoto, vengono utilizzati i valori negli attributi *MsgRetryCount* e *MsgRetryInterval* .

Se il campo *MsgRetryExitId* non è vuoto, viene eseguito il programma di uscita con questo nome. Per ulteriori informazioni sull'utilizzo dei propri programmi di uscita, consultare “Programmi di uscita canale per canali di messaggistica” a pagina 399.

Restituisci messaggio al mittente

Si restituisce un messaggio al mittente richiedendo che venga generato un messaggio di report per includere tutto il messaggio originale.

Consultare “Messaggi di report” a pagina 11 per i dettagli sulle opzioni dei messaggi di report.

Utilizzo della coda dei messaggi non recapitabili (messaggi non recapitati)

Quando un gestore code non è in grado di consegnare un messaggio, tenta di inserire il messaggio nella coda di messaggi non recapitabili. Questa coda deve essere definita quando il gestore code è installato.

I programmi possono utilizzare la coda di messaggi non recapitabili nello stesso modo in cui viene utilizzata dal gestore code. È possibile trovare il nome della coda di messaggi non recapitabili aprendo l'oggetto gestore code (utilizzando la chiamata MQOPEN) e verificando l'attributo *DeadLetterQName* (utilizzando la chiamata MQINQ).

Quando il gestore code inserisce un messaggio su questa coda, aggiunge un'intestazione al messaggio, il cui formato è descritto dalla struttura MQDLH (dead - letter header); consultare MQDLH - Dead - letter header . Questa intestazione include il nome della coda di destinazione e il motivo per cui il messaggio è stato inserito sulla coda di messaggi non recapitabili. È necessario rimuoverlo e risolvere il problema prima di inserire il messaggio nella coda desiderata. Inoltre, il gestore code modifica il campo *Format* del descrittore del messaggio (MQMD) per indicare che il messaggio contiene una struttura MQDLH.

Struttura MQDLH

Si consiglia di aggiungere una struttura MQDLH a tutti i messaggi inseriti nella coda di messaggi non recapitabili; tuttavia, se si intende utilizzare il gestore di messaggi non recapitabili fornito da alcuni prodotti WebSphere MQ , è ***necessario*** aggiungere una struttura MQDLH ai messaggi.

L'aggiunta dell'intestazione a un messaggio potrebbe rendere il messaggio troppo lungo per la coda di messaggi non recapitabili, quindi accertarsi sempre che i messaggi siano più brevi della dimensione massima consentita per la coda di messaggi non recapitabili, almeno del valore della costante MQ_MSG_HEADER_LENGTH. La dimensione massima dei messaggi consentiti in una coda è determinata dal valore dell'attributo *MaxMsgLength* della coda. Per la coda di messaggi non recapitabili, assicurarsi di impostare questo attributo sul valore massimo consentito dal gestore code. Se l'applicazione non è in grado di consegnare un messaggio e il messaggio è troppo lungo per essere inserito nella coda di messaggi non recapitabili, seguire i consigli forniti nella descrizione della struttura MQDLH.

Verificare che la coda di messaggi non recapitabili sia monitorata e che tutti i messaggi in arrivo su di essa vengano elaborati. Il gestore code di messaggi non instradabili viene eseguito come un programma di utilità batch e può essere utilizzato per eseguire varie azioni sui messaggi selezionati sulla coda di messaggi non instradabili. Per ulteriori dettagli, fare riferimento a “Elaborazione coda di messaggi non recapitabili” a pagina 554.

Se la conversione dati è necessaria, il gestore code converte le informazioni di intestazione quando si utilizza l'opzione MQGMO_CONVERT nella chiamata MQGET. Se il processo di inserimento del messaggio è un MCA, l'intestazione è seguita da tutto il testo del messaggio originale.

I messaggi inseriti nella coda di messaggi non recapitabili potrebbero essere troncati se sono troppo lunghi per questa coda. Una possibile indicazione di questa situazione è che i messaggi sulla coda di messaggi non instradabili hanno la stessa lunghezza del valore dell'attributo *MaxMsgLength* della coda.

Elaborazione coda di messaggi non recapitabili

Queste informazioni contengono informazioni generali sull'interfaccia di programmazione quando si utilizza l'elaborazione della coda di messaggi non recapitabili.

L'elaborazione della coda dei messaggi non instradati dipende dai requisiti del sistema locale, ma considerare quanto segue quando si stabilisce la specifica:

- Il messaggio può essere identificato come avente un'intestazione della coda di messaggi non instradabili poiché il valore del campo formato in MQMD è MQFMT_DEAD_LETTER_HEADER.
- Su WebSphere MQ per z/OS utilizzando CICS, se un MCA inserisce questo messaggio nella coda di messaggi non recapitabili, il campo *PutApplType* è MQAT_CICS e il campo *PutApplName* è il *ApplId* del sistema CICS seguito dal nome della transazione dell'MCA.
- Il motivo per cui il messaggio viene instradato alla coda di messaggi non instradati è contenuto nel campo *Reason* dell'intestazione della coda di messaggi non instradati.
- L'intestazione della coda di messaggi non instradabili contiene i dettagli del nome della coda di destinazione e del nome del gestore code.
- L'intestazione della coda di messaggi non instradabili contiene campi che devono essere ripristinati nel descrittore del messaggio prima che il messaggio venga inserito nella coda di destinazione. Sono:

1. *Encoding*
2. *CodedCharSetId*
3. *Format*

- Il descrittore del messaggio è uguale a PUT dall'applicazione originale, ad eccezione dei tre campi mostrati (Codifica, CodedCharSetIde Formato).

L'applicazione della coda di messaggi non recapitabili deve eseguire una o più delle seguenti operazioni:

- Esaminare il campo *Reason* . Un messaggio potrebbe essere stato inserito da un MCA per i seguenti motivi:
  - Il messaggio era più lungo della dimensione massima del messaggio per il canale

    Il motivo è MQRC_MSG_TOO_BIG_FOR_CHALLEGATO
  - Non è stato possibile inserire il messaggio nella coda di destinazione

    Il motivo è un qualsiasi codice motivo MQRC_ * che può essere restituito da un'operazione MQPUT
  - Questa azione è stata richiesta da un'uscita utente

    Il codice di errore è quello fornito dall'uscita utente oppure il codice MQRC_SUPPRESSED_BY_EXIT predefinito
- Provare ad inoltrare il messaggio alla destinazione desiderata, dove ciò è possibile.
- Conservare il messaggio per un certo periodo di tempo prima di scartare quando viene determinato il motivo della deviazione, ma non immediatamente correggibile.
- Fornire istruzioni agli amministratori per correggere i problemi in cui sono stati determinati.
- Eliminare i messaggi danneggiati o non elaborabili.

Esistono due modi per gestire i messaggi recuperati dalla coda di messaggi non recapitabili:

1. Se il messaggio è per una coda locale:
   - Eseguire tutte le traduzioni di codice richieste per estrarre i dati dell'applicazione
   - Eseguire le conversioni di codice su tali dati se si tratta di una funzione locale
   - Inserire il messaggio risultante nella coda locale con tutti i dettagli del descrittore del messaggio ripristinato
2. Se il messaggio è per una coda remota, inserirlo nella coda.

Per informazioni sul modo in cui vengono gestiti i messaggi non recapitati in un ambiente di accodamento distribuito, consultare Cosa succede quando un messaggio non può essere consegnato?.

Programmazione multicast

Utilizzare queste informazioni per informazioni sulle attività di programmazione di WebSphere MQ Multicast, come la connessione a un gestore code e la notifica delle eccezioni.

WebSphere MQ Multicast è stato progettato per essere il più trasparente possibile per l'utente e comunque compatibile con le applicazioni esistenti. Definire un oggetto COMMINFO e impostare i parametri **MCAST** e **COMMINFO** dell'oggetto TOPIC significa che le applicazioni WebSphere MQ esistenti non richiedono una riscrittura sostanziale per utilizzare il multicast. Tuttavia, potrebbero essere presenti alcune limitazioni (consultare “Multicast e Message Queue Interface” a pagina 555 per ulteriori informazioni) e alcuni problemi di sicurezza da considerare (consultare Sicurezza multicast per ulteriori informazioni).

Multicast e Message Queue Interface

Utilizzare queste informazioni per comprendere i principali concetti MQI e il modo in cui sono correlati a WebSphere MQ Multicast.

Le sottoscrizioni multicast non sono durevoli; poiché non vi sono code fisiche coinvolte, non è possibile memorizzare i messaggi non in linea creati dalle sottoscrizioni durevoli.

Dopo che un'applicazione ha effettuato la sottoscrizione a un argomento multicast, le viene restituito un handle di oggetto che può utilizzare o MQGET da, come se fosse un handle per una coda. Ciò significa che sono supportate solo le sottoscrizioni multicast gestite (sottoscrizioni create con MQSO_MANAGED), ossia non è possibile effettuare una sottoscrizione e 'puntare ' i messaggi su una coda. Ciò significa che i messaggi devono essere utilizzati dall'handle dell'oggetto restituito sulla chiamata di sottoscrizione. Sul client, i messaggi vengono memorizzati in un buffer di messaggi fino a quando non vengono utilizzati dal client; per ulteriori informazioni, consultare la stanza MessageBuffer del file di configurazione del client . Se il client non si mantiene al passo con la velocità di pubblicazione, i messaggi vengono eliminati come richiesto, con i messaggi meno recenti eliminati per primi.

Di solito è una decisione di amministrazione se un'applicazione utilizza o meno Multicast, specificata impostando l'attributo MCAST di un oggetto TOPIC. Se un'applicazione di pubblicazione deve garantire che il multicast non venga utilizzato, può utilizzare l'opzione `MQOO_NO_MULTICAST` . Allo stesso modo, un'applicazione di sottoscrizione può garantire che il multicast non venga utilizzato sottoscrivendo con l'opzione `MQSO_NO_MULTICAST` .

WebSphere MQ Multicast supporta l'utilizzo dei selettori di messaggi. Un selettore viene utilizzato da una applicazione per registrare il suo interesse solo in quei messaggi con proprietà che soddisfano la query SQL92 rappresentata dalla stringa di selezione. Per ulteriori informazioni sui selettori dei messaggi, consultare “Selettori” a pagina 22.

La seguente tabella elenca tutti i principali concetti MQI e come sono correlati a Multicast:

Tabella 71. Concetti MQI e modalità di correlazione con multicast

Concetto MQI	**Azione quando si tenta di utilizzare il multicast**	**Codice di errore**
Inserimento di un messaggio di lunghezza zero	Rifiutato	2005 (07D5) (RC2005): MQRC_BUFFER_LENGTH_ERROR
Raggruppamento	Rifiutato	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
Segmentazione	Rifiutato	2443 (098B) (RC2443): MQRC_SEGMENTATION_NOT_ALLOWED
Liste di distribuzione	Rifiutato	2154 (086A) (RC2154): MQRC_RECS_PRESENT_ERROR
MQINQ	Rifiutato per handle di argomenti: MQINQ e MQSET di argomenti non sono supportati.	2038 (07F6) (RC2038): MQRC_NOT_OPEN_FOR_INQUIRE
	Accettato per l'handle gestito. È possibile interrogare solo `Profondità corrente` .	• Se il valore è Profondità corrente, non esiste alcun codice motivo applicabile. • Se il valore è diverso da Profondità corrente, il codice motivo è 2067 (0813) (RC2067): MQRC_SELECTOR_ERROR.
MQSET	Rifiutato per tutti gli handle.	2040 (07F8) (RC2040): MQRC_NOT_OPEN_FOR_SET
Transazioni (XA o meno)	Rifiutato	2072 (0818) (RC2072): MQRC_SYNCPOINT_NOT_AVAILABLE
Ricerca messaggi	Rifiutato	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Blocca messaggi	Rifiutato	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR

Tabella 71. Concetti MQI e modalità di correlazione con multicast (Continua)		
Concetto MQI	**Azione quando si tenta di utilizzare il multicast**	**Codice di errore**
Sfoglia con contrassegno	Rifiutato	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Passa contesto	Rifiutato	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPUT1	Rifiutato. Non è valido provare MQPUT1 per un argomento solo multicast.	2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY
Sottoscrizione durevole	Rifiutato se l'argomento è contrassegnato come "Solo multicast", altrimenti viene effettuata una sottoscrizione non multicast.	2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED
TopicString > 255	Rifiutato. Se la stringa dell'argomento contiene più di 255 caratteri, viene rifiutata nel client.	2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR
Sottoscrizione non gestita effettuata	Rifiutato se l'argomento è contrassegnato come "Solo multicast", altrimenti viene effettuata una sottoscrizione non multicast.	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPMO_NOT_OWN_SUBS	Rifiutato	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR

I seguenti elementi espandono alcuni dei concetti MQI della precedente tabella e forniscono informazioni su alcuni concetti MQI non presenti nella tabella:

Persistenza messaggio
Per sottoscrittori multicast non durevoli, i messaggi persistenti del publisher vengono consegnati in modo irreversibile.

Troncamento dei messaggi
Il troncamento del messaggio è supportato, il che significa che è possibile per un'applicazione:

1. Emettere un MQGET.
2. Richiamare MQRC_TRUNCATED_MSG_FAILED.
3. Assegnare un buffer più grande.

4. Emettere nuovamente MQGET per recuperare il messaggio.

Scadenza sottoscrizione
La scadenza della sottoscrizione non è supportata. Qualsiasi tentativo di impostare una scadenza viene ignorato.

Alta disponibilità per multicast

Utilizzare queste informazioni per comprendere WebSphere MQ l'operazione peer-to-peer continua multicast; sebbene WebSphere MQ si connetta a un gestore code WebSphere MQ , i messaggi non vengono trasmessi attraverso tale gestore code.

Anche se è necessario stabilire una connessione a un gestore code per consentire a MQOPEN o MQSUB l'oggetto argomento multicast, i messaggi stessi non passano attraverso il gestore code. Pertanto, dopo che MQOPEN o MQSUB è stato completato sull'oggetto dell'argomento multicast, è possibile continuare a trasmettere messaggi multicast anche se la connessione al gestore code è stata persa. Esistono due modalità di funzionamento:

Viene stabilita una connessione normale al gestore code
La comunicazione multicast è possibile mentre esiste la connessione al gestore code. Se la connessione non riesce, vengono applicate le normali regole MQI, ad esempio; un MQPUT all'handle dell'oggetto multicast restituisce 2009 (07D9) (RC2009): MQRC_CONNECTION_BROKEN.

Viene effettuata una connessione client al gestore code
La comunicazione multicast è possibile anche durante il ciclo di riconnessione. Ciò significa che anche quando la connessione al gestore code è stata interrotta, l'immissione e l'utilizzo di messaggi multicast non vengono influenzati. Il client tenta di riconnettersi a un gestore code e, se la riconnessione ha esito negativo, l'handle di connessione si interrompe e tutte le chiamate MQI, incluse quelle multicast, non riescono. Per ulteriori informazioni, consultare: Riconnessione client automatizzata

Se un'applicazione emette esplicitamente un MQDISC, tutte le sottoscrizioni multicast e gli handle degli oggetti vengono chiusi.

Operazione peer-to-peer continua multicast

Uno dei vantaggi della comunicazione peer - to - peer tra i client è che i messaggi non devono passare attraverso il gestore code; pertanto, se la connessione al gestore code si interrompe, il trasferimento dei messaggi continua. Le seguenti limitazioni si applicano ai requisiti dei messaggi continui di questa modalità:

- La connessione deve essere effettuata utilizzando una delle opzioni MQCNO_RECONNECT_ * per l'operazione continua. Questo processo indica che sebbene la sessione di comunicazioni potrebbe essere interrotta, l'handle di connessione effettivo non è interrotto e si trova invece nello stato di riconnessione. Se la riconnessione non riesce, l'handle di connessione è ora interrotto e ciò impedisce tutte le ulteriori chiamate MQI.
- In questa modalità sono supportati solo MQPUT, MQGET, MQINQ e Async Consume. Qualsiasi comando MQOPEN, MQCLOSE o MQDISC richiede il completamento della riconnessione al gestore code.
- Lo stato passa all'arresto del gestore code; qualsiasi stato nel gestore code potrebbe quindi essere obsoleto o mancante. Ciò significa che i client potrebbero inviare e ricevere messaggi e che non vi è alcuno stato noto sul gestore code. Per ulteriori informazioni, consultare: Monitoraggio dell'applicazione multicast

Conversione dati in MQI per la messaggistica multicast

Utilizzare queste informazioni per comprendere come funziona la conversione dei dati per la messaggistica WebSphere MQ Multicast.

WebSphere MQ Multicast è un protocollo condiviso, senza connessione, e quindi non è possibile per ogni client effettuare richieste specifiche per la conversione dei dati. Ogni client sottoscritto allo stesso flusso

multicast riceve gli stessi dati binari; pertanto, se è richiesta la conversione di dati WebSphere MQ , la conversione viene eseguita localmente su ciascun client.

I dati vengono convertiti sul client per il traffico multicast WebSphere MQ . Se viene specificata l'opzione **MQGMO_CONVERT** , la conversione dei dati viene eseguita come richiesto. I formati definiti dall'utente richiedono l'uscita di conversione dati installata sul client; consultare “Scrittura delle uscite di conversione dati” a pagina 417 per informazioni su quali librerie si trovano ora nei pacchetti client e server.

Per informazioni sulla gestione della conversione dati, consultare Abilitazione della conversione dati per la messaggistica multicast.

Per ulteriori informazioni sulla conversione dei dati, consultare Conversione dei dati.

Per ulteriori informazioni sulle uscite di conversione dati e `ClientExitPath`, consultare la sezione ClientExitPercorso del file di configurazione client.

Report di eccezioni multicast

Utilizzare queste informazioni per informazioni su WebSphere MQ Multicast event handler and reporting WebSphere MQ Multicast exceptions.

WebSphere MQ Multicast assiste nella determinazione dei problemi richiamando il gestore eventi per notificare eventi multicast riportati utilizzando il meccanismo del gestore eventi standard WebSphere MQ .

Un singolo evento Multicast può determinare la chiamata di più di un evento WebSphere MQ poiché potrebbero essere presenti più handle di connessione `MQHCONN` che utilizzano lo stesso trasmettitore o ricevitore multicast. Tuttavia, ciascuna eccezione multicast fa sì che venga richiamato un solo gestore eventi per connessione WebSphere MQ .

La costante WebSphere MQ `MQCBDO_EVENT_CALL` consente alle applicazioni di registrare un callback per ricevere solo eventi WebSphere MQ e `MQCBDO_MC_EVENT_CALL` consente alle applicazioni di registrare un callback per ricevere solo eventi multicast. Se vengono utilizzate entrambe le costanti, vengono ricevuti entrambi i tipi di evento.

Richiesta di eventi multicast

WebSphere MQ Gli eventi multicast utilizzano la costante `MQCBDO_MC_EVENT_CALL` nel campo `cbd.Options` . Il seguente esempio illustra come richiedere eventi multicast:

```
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);
```

Quando l'opzione `MQCBDO_MC_EVENT_CALL` viene specificata per il campo `cbd.Options` , il programma di gestione eventi viene inviato solo WebSphere MQ eventi multicast invece di eventi di livello di connessione. Per richiedere che entrambi i tipi di eventi vengano inviati al gestore eventi, l'applicazione deve specificare la costante `MQCBDO_EVENT_CALL` nel campo `cbd.Options` e la costante `MQCBDO_MC_EVENT_CALL` come mostrato nel seguente esempio:

```
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);
```

Se nessuna di queste costanti viene utilizzata, solo gli eventi a livello di connessione vengono inviati al gestore eventi.

Per ulteriori informazioni sui valori per il campo `Options` , consultare Opzioni (MQLONG).

Formato evento multicast

WebSphere MQ Le eccezioni Multicast includono alcune informazioni di supporto restituite nel parametro **Buffer** della funzione di callback. Il puntatore **Buffer** punta ad un array di puntatori e il campo `MQCBC.DataLength` specifica la dimensione, in byte, dell'array. Il primo elemento dell'array punta sempre a una breve descrizione di testo dell'evento. Ulteriori parametri potrebbero essere forniti in base al tipo di evento. La seguente tabella elenca le eccezioni:

Tabella 72. Descrizioni codice evento multicast

Codice evento	Descrizione	Dati aggiuntivi
CARICO PACCHETTO MQMCEV	Perdita di pacchetti non recuperabile	Numero di pacchetti persi
MQMCEV_HEARTBEAT_TIMEOUT	Lunga assenza del pacchetto di controllo heartbeat	N.d.
MQMCEV_VERSION_CONFLICT	Ricezione di pacchetti di versioni di protocollo più recenti	N.d.
MQMCEV_AFFIDABILITÀ	Diverse modalità di affidabilità del trasmettitore e del ricevitore	N.d.
TRANS MQMCEV_CLOSED_	La trasmissione dell'argomento è chiusa da 1 origine	N.d.
ERRORE MQMCEV_STREAM_	Errore rilevato sul flusso	N.d.
MQMCEV_NEW_SOURCE	Una nuova origine inizia a trasmettere sull'argomento	Struttura di origine
MQMCEV_RECEIVE_QUEUE_TRIMMED	Pacchetti rimossi da PacketQ a causa della scadenza di tempo o spazio	Numero di pacchetti ritagliati
MQMCEV_PACKET_LOSS_NACK_EXPIRE	Perdita di pacchetti non recuperabile a causa della scadenza NACK	Numero di pacchetti persi
MQMCEV_ACK_RETRIES_SUPERATO	Pacchetti rimossi dalla cronologia dopo il superamento di **max_ack_retries**	Numero di pacchetti rimossi
MQMCEV_STREAM_SUSPEND_NACK	I NACK sono stati sospesi su un flusso accettato da questo argomento	Sospendi ID flusso
		Tempo in millisecondi in cui il flusso è sospeso per
MQMCEV_STREAM_RESUME_NACK	I NACK sono stati ripresi dopo essere stati sospesi su un flusso	ID flusso
MQMCEV_STREAM_ESPULSO	Un flusso accettato da questo argomento è stato rifiutato a causa di una richiesta di espulsione	ID flusso
MQMCEV_PRIMO_MESSAGGIO	Primo messaggio da un'origine	Numero messaggio
MQMCEV_LATE_JOIN_FAILURE	Impossibile avviare la sessione di unione in ritardo	N.d.
MQMCEV_MESSAGE_LOSS	Perdita del messaggio non recuperabile	Numero di messaggi persi

Tabella 72. Descrizioni codice evento multicast (Continua)

Codice evento	Descrizione	Dati aggiuntivi
MQMCEV_SEND_PACKET_FAILURE	Il trasmettitore multicast non è riuscito a inviare un pacchetto multicast	N.d.
MQMCEV_REPAIR_DELAY	Il ricevitore multicast non ha ricevuto un pacchetto di riparazione per un NAK in sospeso	N.d.
MQMCEV_MEMORY_ALERT_ATTIVO	I buffer di ricezione del ricevitore si stanno riempiendo	Percentuale di utilizzo bufferpool
MQMCEV_MEMORY_ALERT_OFF	I buffer di ricezione del ricevitore sono inattivi al livello normale	Percentuale di utilizzo bufferpool
MQMCEV_NACK_ALERT_ATTIVO	La frequenza della richiesta del pacchetto di riparazione del ricevitore ha raggiunto il limite massimo	La frequenza corrente delle richieste di riparazione in pacchetti al secondo
MQMCEV_NACK_ALERT_OFF	La velocità di richiesta del pacchetto di riparazione del destinatario è inferiore al normale	La frequenza corrente delle richieste di riparazione in pacchetti al secondo
MQMCEV_REPAIR_ALERT_ON	La velocità di invio del pacchetto di riparazione del trasmettitore ha raggiunto il limite massimo	N.d.
MQMCEV_REPAIR_ALERT_OFF	La velocità di invio del pacchetto di riparazione del trasmettitore è inferiore al normale	N.d.
MQMCEV_SHM_DEST_UNUSABLE	È stato rilevato che la regione di memoria condivisa utilizzata da una destinazione argomento del trasmettitore è inutilizzabile	N.d.
MQMCEV_SHM_PORT_UNUSABLE	È stato rilevato che la porta di memoria condivisa utilizzata da un'istanza del destinatario è inutilizzabile	N.d.
MQMCEV_CCT_GETTIME_NON RIUSCITO	L'ora di richiamo dall'ora del cluster coordinato non è riuscita	N.d.
MQMCEV_DEST_INTERFACE_FAILURE	L'interfaccia di rete utilizzata da una destinazione argomento del trasmettitore ha avuto esito negativo e un'interfaccia di rete di backup non è disponibile	
MQMCEV_DEST_INTERFACE_FAILOVER	L'interfaccia di rete utilizzata da una destinazione argomento del trasmettitore ha avuto esito negativo ed è stato completato correttamente un failover su un'altra interfaccia	

Tabella 72. Descrizioni codice evento multicast (Continua)		
Codice evento	**Descrizione**	**Dati aggiuntivi**
MQMCEV_PORT_INTERFACE - ERRORE	L'interfaccia di rete utilizzata da un ricevitore rmmPort ha avuto esito negativo e un'interfaccia di rete di backup non è disponibile (o ha avuto esito negativo)	Configurazione di RMM
MQMCEV_PORT_INTERFACE_FAILOVER	L'interfaccia di rete utilizzata da un ricevitore rmmPort ha avuto esito negativo ed è stato completato un failover corretto su un'altra interfaccia	Configurazione di RMM

Usando .Net

WebSphere Le classi di MQ per .NET consentono a un programma scritto nel framework di programmazione .NET di connettersi a WebSphere MQ come client WebSphere MQ MQI o di connettersi direttamente a un WebSphere MQ .

Se si dispone di applicazioni che utilizzano Microsoft .NET Framework e si desidera utilizzare le funzionalità di WebSphere MQ, è necessario utilizzare le classi WebSphere MQ per .NET.

L'interfaccia di WebSphere MQ .NET orientata agli oggetti è diversa dall'interfaccia MQI in quanto utilizza metodi di oggetti piuttosto che utilizzare i verbi MQI.

L'API (application programming interface) WebSphere MQ procedurale si basa su verbi come quelli presenti nel seguente elenco:

```
MQCONN, MQDISC, MQOPEN, MQCLOSE,
MQINQ, MQSET, MQGET, MQPUT, MQSUB
```

Tutti questi verbi assumono, come parametro, un handle all'oggetto WebSphere MQ su cui devono operare. Poiché .NET è orientato agli oggetti, l'interfaccia di programmazione .NET gira questo giro. Il programma è costituito da una serie di oggetti WebSphere MQ , su cui si agisce richiamando i metodi su tali oggetti. È possibile scrivere programmi in qualsiasi linguaggio supportato da .NET.

Quando si utilizza l'interfaccia procedurale, ci si disconnette da un gestore code utilizzando la chiamata MQDISC (*Hconn*, CompCode, Reason), dove *Hconn* è un handle per il gestore code.

Nell'interfaccia .NET, il gestore code è rappresentato da un oggetto della classe MQQueueManager. È possibile disconnettersi dal gestore code richiamando il metodo Disconnect () su tale classe.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.Disconnect();
```

WebSphere MQ classes per .NET è una serie di classi che consentono alle applicazioni .NET di interagire con WebSphere MQ. Essi rappresentano i diversi componenti di WebSphere MQ utilizzati dall'applicazione, quali gestori code, code, canali e messaggi. Per dettagli su queste classi, consultare Le classi e le interfacce WebSphere MQ .NET.

Prima di poter compilare le applicazioni scritte, è necessario che sia installato .NET Framework. Per istruzioni sull'installazione delle classi WebSphere MQ for .NET e .NET Framework, consultare “Installazione di classi WebSphere MQ per .NET” a pagina 564.

Concetti correlati

Panoramica tecnica

“Opzioni di connessione” a pagina 563
Esistono tre modi per connettere le classi WebSphere MQ per .NET a un gestore code. Considerare il tipo di connessione più adatto alle proprie esigenze.

“Utilizzo delle classi WebSphere MQ per .NET” a pagina 574
Questa raccolta di argomenti descrive come configurare il sistema per eseguire i programmi di esempio per verificare l'installazione delle classi WebSphere MQ per .NET e come eseguire i propri programmi.

“Risoluzione dei problemi relativi a WebSphere MQ .NET” a pagina 577
Se un programma non viene completato correttamente, eseguire una delle applicazioni di esempio e seguire i consigli riportati nei messaggi di diagnostica.

“Scrittura e distribuzione di programmi WebSphere MQ .NET” a pagina 578
Per utilizzare le classi WebSphere MQ per .NET per accedere alle code WebSphere MQ , scrivere i programmi in qualsiasi linguaggio supportato da .NET contenente le chiamate che inserono i messaggi e richiamano i messaggi dalle code WebSphere MQ .

“Canale personalizzato IBM WebSphere MQ per Microsoft Windows Communication Foundation (WCF)” a pagina 597
Il canale personalizzato Microsoft Windows Communication Foundation (WCF) per IBM WebSphere MQ invia e riceve messaggi tra i servizi e i client WCF.

“Scelta del linguaggio di programmazione da utilizzare” a pagina 79
Utilizza queste informazioni per scoprire i linguaggi di programmazione e i framework supportati da IBM WebSphere MQe alcune considerazioni per utilizzarli.

“Sviluppo delle applicazioni” a pagina 7
IBM WebSphere MQ fornisce diversi modi in cui è possibile sviluppare applicazioni per inviare e ricevere messaggi necessari per supportare i processi aziendali. È anche possibile sviluppare applicazioni per gestire i gestori code e le risorse correlate.

Introduzione a WebSphere MQ classes per .NET

WebSphere Le classi di MQ per .NET consentono a un programma scritto nel framework di programmazione .NET di connettersi a WebSphere MQ come client WebSphere MQ MQI o di connettersi direttamente a un WebSphere MQ .

Opzioni di connessione

Esistono tre modi per connettere le classi WebSphere MQ per .NET a un gestore code. Considerare il tipo di connessione più adatto alle proprie esigenze.

Connessione bind client

Per utilizzare le classi WebSphere MQ per .NET come client MQI WebSphere MQ , è possibile installarlo con il client WebSphere MQ MQI, sulla macchina server WebSphere MQ o su una macchina separata. Una connessione di bind client può utilizzare transazioni XA o non XA

Connessione bind server

Quando vengono utilizzate in modalità di bind del server, le classi WebSphere MQ per .NET utilizzano l'API del gestore code invece di comunicare attraverso una rete. Ciò fornisce migliori prestazioni per le applicazioni WebSphere MQ rispetto all'uso delle connessioni di rete.

Per utilizzare la connessione dei collegamenti, è necessario installare le classi WebSphere MQ per .NET sul server WebSphere MQ .

Connessione client gestito

Una connessione effettuata in questa modalità si connette come client WebSphere MQ a un server WebSphere MQ in esecuzione su una macchina locale o remota.

Le classi WebSphere MQ per .NET che si collegano in questa modalità rimangono nel codice gestito .NET e non effettuano chiamate ai servizi nativi. Per ulteriori informazioni sul codice gestito, fare riferimento alla documentazione di Microsoft .

Esistono diverse limitazioni all'utilizzo del client gestito. Per ulteriori informazioni, consultare “Connessioni client gestite” a pagina 578.

Installazione di classi WebSphere MQ per .NET

WebSphere MQ classes per .NET, inclusi gli esempi, viene installato con WebSphere MQ. Esiste un prerequisito di Microsoft .NET Framework.

L'ultima versione di WebSphere MQ classes for .NET viene installata, per impostazione predefinita, come parte dell'installazione standard di WebSphere MQ nella funzione *Servizi Web e messaggistica Java e .NET* . Per le istruzioni di installazione, consultare Installazione del server IBM WebSphere MQ su Windows o Installazione di un client IBM WebSphere MQ su sistemi Windows .

In un ambiente di installazione multipla, se in precedenza sono state installate le classi WebSphere MQ per .NET come pacchetto di supporto, non è possibile installare WebSphere MQ a meno che non si disinstalli prima il pacchetto di supporto. La funzione WebSphere MQ classes per .NET installata con WebSphere MQ contiene la stessa funzione del pacchetto di supporto.

Vengono fornite anche applicazioni di esempio, inclusi i file di origine; consultare “Applicazioni di esempio” a pagina 575.

Per eseguire le classi WebSphere MQ per .NET su piattaforme a 32 bit o a 64 bit, è necessario aver installato Microsoft .NET Framework V2.0 o versioni successive.

Nota: Se Microsoft .NET Framework v2.0 o versione successiva non è installato prima di installare WebSphere MQ V7.0.1, l'installazione del prodotto WebSphere MQ continuerà senza errori, ma le classi WebSphere MQ per .NET non saranno disponibili. Se .NET Framework è installato dopo l'installazione di WebSphere MQ 7.0.1, gli assembly di WebSphere .NET devono essere registrati eseguendo lo script *WMQInstallDir*`\bin\amqiRegisterdotNet.cmd` , dove *WMQInstallDir* è la directory in cui è installato WebSphere MQ 7.0.1 . Questo script installa gli assembly richiesti nella GAC (Global Assembly Cache). Una serie di file `amqi*.log` che registrano le azioni eseguite viene creata nella directory %TEMP% .

Per informazioni sull'utilizzo del canale personalizzato WebSphere MQ per Microsoft WCF con .NET 3, consultare: “Canale personalizzato IBM WebSphere MQ per Microsoft Windows Communication Foundation (WCF)” a pagina 597

Transazioni distribuite in .NET

Le transazioni distribuite o le transazioni globali consentono alle applicazioni client di includere diverse fonti di dati su due o più sistemi di rete in una transazione.

Nelle transazioni distribuite, un gestore transazioni coordina e gestisce la transazione tra due o più gestori risorse.

Le transazioni possono essere un processo di commit a fase singola o a due fasi. Il commit a fase singola è un processo in cui solo un gestore risorse partecipa alla transazione e il processo di commit a due fasi è in cui più di un gestore risorse partecipa alla transazione. Nel processo di commit a due fasi, il gestore transazioni invia una chiamata di preparazione per verificare se tutti i gestori risorse sono pronti per il commit. Quando riceve il riconoscimento da tutti i gestori risorse, viene emessa la chiamata di commit. Altrimenti, si verifica un rollback sull'intera transazione. Consultare Gestione delle transazioni e supporto per ulteriori dettagli. I gestori delle risorse devono informare i gestori delle transazioni della partecipazione alla transazione. Quando il gestore risorse informa il gestore transazioni della sua

partecipazione, il gestore risorse riceve i callback dal gestore transazioni quando la transazione sta per eseguire il commit o il rollback.

WebSphere MQ Le classi .NET già supportano transazioni distribuite in connessioni in modalità bind server e non gestite. In queste modalità, WebSphere le classi di MQ .NET delegano tutte le relative chiamate al client di transazioni estese C, che gestisce l'elaborazione delle transazioni per conto di .NET.

WebSphere MQ.NET ora supportano le transazioni distribuite in modalità gestita dove WebSphere MQ .NET Classes utilizza lo spazio dei nomi System.Transactions per il supporto delle transazioni distribuite. L'infrastruttura System.Transactions rende la programmazione transazionale semplice ed efficiente, supportando le transazioni avviate in tutti i gestori risorse incluso WebSphere MQ. L'applicazione WebSphere MQ .NET può inserire e richiamare messaggi utilizzando la programmazione di transazioni implicite .NET o un modello di programmazione di transazioni esplicite. Nelle transazioni implicite, i limiti della transazione vengono creati dal programma applicativo che decide quando eseguire il commit, il rollback (per le transazioni esplicite) o completare la transazione. Nelle transazioni esplicite, è necessario specificare esplicitamente se si desidera eseguire il commit, il rollback e il completamento della transazione.

WebSphere MQ.NET utilizza Microsoft distributed transaction coordinator (MS DTC) come gestore transazioni, che coordina e gestisce la transazione tra più gestori risorse. WebSphere MQ viene utilizzato come gestore risorse.

WebSphere MQ.NET segue il modello DTP (Distributed Transaction Processing) X/Open. Il modello X/Open Distributed Transaction Processing è un modello di elaborazione delle transazioni distribuite proposto da Open Group, un consorzio di fornitori. Questo modello è uno standard tra la maggior parte dei fornitori commerciali nell'elaborazione delle transazioni e nei domini di database. La maggior parte dei prodotti di gestione delle transazioni commerciali supporta il modello X/DTP.

Modalità di transazione

- “Transazioni distribuite in modalità gestita” a pagina 566
- Transazioni distribuite per modalità non gestita

Coordinamento delle transazioni in vari scenari

- Una connessione potrebbe partecipare a diverse transazioni, ma solo una transazione è attiva in qualsiasi momento.
- Durante una transazione, MQQueueManager.La chiamata di disconnessione viene rispettata. In questo caso, viene richiesto il rollback della transazione.
- Durante una transazione, viene rispettata la chiamata MQQueue.Close o MQTopic.Close . In questo caso viene richiesto il rollback della transazione.
- I limiti di transazione vengono creati dal programma applicativo che decide quando eseguire il commit, il rollback (per le transazioni esplicite) o il completamento (per le transazioni implicite) della transazione.
- Se l'applicazione client si interrompe durante una transazione con un errore non previsto prima di emettere una chiamata Put o Get su una chiamata della coda o dell'argomento, viene eseguito il rollback della transazione e viene generata una MQException.
- Se il codice motivo MQCC_FAILED viene restituito durante una chiamata Put o Get su una coda o su una chiamata Topic, viene generata un'eccezione MQException con codice motivo e viene eseguito il rollback della transazione. Se una chiamata di preparazione è stata già emessa dal gestore transazioni, WebSphere MQ .NET restituisce la richiesta di preparazione eseguendo forzatamente il rollback della transazione. Quindi, il gestore transazioni DTC causa un rollback sul lavoro corrente con tutti i gestori risorse nelle transazioni dell'ambiente correnti.
- Durante una transazione che coinvolge più gestori risorse se alcuni motivi ambientali causano il blocco indefinito della chiamata Put o Get, il gestore transazioni attende fino a un tempo stabilito. Dopo che il tempo è terminato, provoca il rollback di tutte le operazioni correnti con tutti i gestori risorse nelle transazioni dell'ambiente corrente. Se questa attesa indefinita si verifica durante la fase di

preparazione, il gestore transazioni potrebbe scadere o emettere una chiamata in dubbio sulla risorsa, nel cui caso viene eseguito il rollback della transazione.

- Le applicazioni che utilizzano le transazioni devono inserire o richiamare i messaggi in SYNC_POINT. Se una chiamata Put o Get del messaggio viene emessa in un contesto transazionale che non si trova in SYNC_POINT, la chiamata ha esito negativo con il codice motivo MQRC_UNIT_OF_WORK_NOT_STARTED.

Differenze comportamentali tra il supporto delle transazioni del client gestito e non gestito utilizzando lo spazio dei nomi Microsoft .NET System.Transactions

Le transazioni nidificate hanno un TransactionScope all'interno di un altro TransactionScope

- WebSphere MQ .NET client completamente gestito supporta TransactionScope nidificato
- WebSphere MQ Il client non gestito .NET non supporta TransactionScope nidificato

Transazioni dipendenti da System.Transactions

- Il client WebSphere MQ .NET completamente gestito supporta la funzione di transazioni dipendenti fornita da System.Transactions.
- WebSphere MQ .NET client non gestito non supporta la funzione di transazioni dipendenti fornita da System.Transactions.

Esempi di Prodotti

Nuovi esempi di prodotto `SimpleXAPute` `SimpleXAGet` sono disponibili in `WebSphere MQ\tools\dotnet\samples\cs\base` . Gli esempi sono applicazioni C#, che dimostrano l'uso di MQPUT e MQGET in Transazioni distribuite utilizzando lo spazio nomi SystemTransactions . Per ulteriori informazioni su questi esempi, consultare “Creazione di semplici messaggi put e get in un TransactionScope” a pagina 569

Transazioni distribuite in modalità gestita

WebSphere MQ Le classi .NET utilizzano lo spazio dei nomi System.Transactions per il supporto delle transazioni distribuite in modalità gestita. Nella modalità gestita, MS DTC coordina e gestisce le transazioni distribuite su tutti i server elencati in una transazione.

WebSphere Le classi MQ .NET forniscono un modello di programmazione esplicito basato sulla classe System.Transactions.Transaction e un modello di programmazione implicito che utilizza la classe System.Transactions.TransactionScope, in cui le transazioni vengono gestite automaticamente dall'infrastruttura.

Transazione implicita

La seguente parte di codice descrive come un'applicazione WebSphere MQ .NET inserisce un messaggio utilizzando la programmazione di transazioni implicite .NET.

```
Using (TransactionScope scope = new TransactionScope ())
{
    Q.Put (putMsg,pmo);
    scope.Complete ();
}

Q.close();
qMgr.Disconect();}
```

Spiegazione del flusso di codice della transazione implicita

Il codice crea *TransactionScope* e inserisce il messaggio nell'ambito. Richiama quindi *Completa* per informare il coordinatore della transazione del completamento della transazione. Il coordinatore della transazione ora emette *prepare* e *commit* per completare la transazione. Se viene rilevato un problema, viene richiamato un *rollback* .

Transazione esplicita

Il seguente codice descrive il modo in cui un'applicazione WebSphere MQ .NET inserisce i messaggi utilizzando il modello di programmazione delle transazioni esplicite .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();
}
```

Spiegazione del flusso di codice della transazione esplicita

La parte di codice crea la transazione utilizzando la classe *CommitableTransaction* . Inserisce un messaggio in tale ambito e richiama esplicitamente *commit* per completare la transazione. In caso di problemi, viene richiamato il comando *rollback* .

Transazioni distribuite in modalità non gestita

WebSphere MQ.NET supportano le connessioni non gestite (client) utilizzando il client di transazione esteso e COM + /MTS come coordinatore della transazione, utilizzando il modello di programmazione di transazione implicito o esplicito. Nella modalità non gestita, le classi WebSphere MQ .NET delegano tutte le relative chiamate al client di transazioni estese C che gestisce l'elaborazione delle transazioni per conto di .NET.

L'elaborazione della transazione è controllata da un gestore transazioni esterno, che coordina l'unità di lavoro globale sotto il controllo dell'API del gestore transazioni. I verbi MQBEGIN, MQCMIT e MQBACK non sono disponibili. WebSphere MQ Le classi .NET espongono questo supporto tramite la modalità di trasporto non gestito (client C). Consultare Configurazione di gestori transazioni compatibili con XA

MTS si è evoluto come un sistema TP (transaction processing) per fornire le medesime funzioni su Windows NT disponibili in CICS, Tuxedo e su altre piattaforme. Quando l'MTS è installato, viene aggiunto un servizio separato a Windows NT denominato Microsoft Distributed Transaction Coordinator (MSDTC). L'MSDTC coordina le transazioni che si estendono a archivi dati o risorse separati. Per funzionare, è necessario che ogni archivio dati implementi il proprio gestore risorse proprietario.

WebSphere MQ diventa compatibile con MSDTC implementando un'interfaccia (interfaccia del gestore risorse proprietaria) in cui gestisce la mappatura delle chiamate XA DTC alle chiamate WebSphere MQ(X/Open). WebSphere MQ svolge il ruolo di gestore risorse.

Quando un componente come COM + richiede l'accesso a un WebSphere MQ, COM di solito verifica con l'oggetto di contesto MTS appropriato se è richiesta una transazione. Se è richiesta una transazione, il COM informa il DTC e avvia automaticamente una transazione WebSphere MQ integrale per questa operazione. Quindi il COM lavora con i dati attraverso il software MQMTS, inserendo e ottenendo i messaggi come richiesto. L'istanza dell'oggetto ottenuta da COM richiama il metodo SetComplete o SetAbort una volta che tutte le azioni sui dati sono state completate. Quando l'applicazione emette SetComplete, la chiamata segnala al DTC che l'applicazione ha completato la transazione e il DTC può procedere con il processo di commit a due fasi. Il DTC emette quindi chiamate a MQMTS che a loro volta emette chiamate a WebSphere MQ per eseguire il commit o il rollback della transazione.

Scrittura di un'applicazione WebSphere MQ .NET utilizzando un client non gestito

Per essere eseguita nel contesto di COM +, una classe .NET deve ereditare da System.EnterpriseServices.ServicedComponent. Le regole e i consigli per creare assiemi che utilizzano componenti assistiti sono i seguenti:

Nota: Le seguenti operazioni sono rilevanti solo se si utilizza la modalità System.EnterpriseServices .

- La classe e il metodo avviati in COM + devono essere entrambi pubblici (nessuna classe interna e nessun metodo protetto o statico).
- Gli attributi della classe e del metodo: l'attributo TransactionOption indica il livello di transazione della classe, ovvero se le transazioni sono disabilitate, supportate o richieste. L'attributo AutoComplete nel metodo ExecuteUOW() indica a COM + di eseguire il commit della transazione se non viene generata alcuna eccezione non gestita.
- Forte - denominazione di un assieme: l'assieme deve avere un nome forte e deve essere registrato nella GAC (Global Assembly Cache). L'assemblea è registrata in COM + esplicitamente o per registrazione pigra dopo che è stata registrata nel GAC.
- Registrazione di un assembly in COM +: preparare l'assembly per essere esposto ai client COM. Quindi, creare una libreria di tipi utilizzando lo strumento di registrazione Assembly, regasm.exe.

  ```
  regasm UnmanagedToManagedXa.dll
  ```

- Registrare l'assieme in GAC `gacutil /i UnmanagedToManagedXa.dll`.
- Registrare l'assembly in COM + utilizzando il programma di installazione dei servizi .NET, regsvcs.exe. Consultare il tipo di libreria creata da regasm.exe:

  ```
  Regsvcs /appname:UnmanagedToManagedXa  /tlb:UnmanagedToManagedXa.tlb
  UnmanagedToManagedXa.dll
  ```

- L'assemblaggio viene distribuito nel GAC, e successivamente viene registrato in COM + tramite registrazione lazy. .NET Framework si occupa della registrazione dopo che il codice è stato eseguito per la prima volta.

Il flusso di codice di esempio utilizzando il modello System.EnterpriseServices e System.Transactions con COM + sono descritti nelle seguenti sezioni:

Esempio di flusso di codice utilizzando il modello 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();
}
```

Esempio di flusso di codice che utilizza System.Transactions per interazioni con 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();
}
```

Creazione di semplici messaggi put e get in un TransactionScope

Le applicazioni C# di esempio del prodotto sono disponibili in WebSphere MQ. Queste semplici applicazioni dimostrano l'inserimento e il richiamo di messaggi in un TransactionScope. Alla fine dell'attività, sarà possibile inserire e richiamare i messaggi da una coda o da un argomento.

Prima di iniziare

Il servizio MSDTC deve essere in esecuzione e abilitato per le transazioni XA.

Informazioni su questa attività

L'esempio è una semplice applicazione, `SimpleXAPut` e `SimpleXAGet`. I programmi `SimpleXAPut` e `SimpleXAGet` sono applicazioni C# disponibili all'interno di WebSphere MQ. `SimpleXAPut` dimostra l'utilizzo di MQPUT, in Transazioni distribuite utilizzando lo spazio dei nomi SystemTransactions . `SimpleXAGet` dimostra di utilizzare MQGET, in Transazioni distribuite utilizzando lo spazio dei nomi SystemTransactions .

`SimpleXAPut` si trova in `WebSphere MQ\tools\dotnet\samples\cs\base`

Procedura

Le applicazioni possono essere eseguite con i parametri della riga comandi da `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]
```

dove i parametri sono:

-destinationURI

Può essere una coda o un argomento. Per una coda, specificare `queue://queueName` e per un argomento specificare `topic://topicName`.

-host

Può essere un nome host come localhost o un indirizzo IP.

-port

La porta su cui è in esecuzione il gestore code.

-channel

Il canale di connessione utilizzato. Il valore predefinito è `SYSTEM.DEF.SVRCONN`

-transaction

Il risultato della transazione, ad esempio commit o rollback.

-mode

La modalità di trasporto, ad esempio gestita o non gestita.

-numberOfMsgs

Il numero di messaggi. Il valore predefinito è 1.

Esempio

```
SimpleXAPut -d topic://T01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

```
SimpleXAGet -d queue://Q01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

Ripristino delle transazioni

Questa sezione descrive il processo di ripristino delle transazioni in WebSphere MQ .NET XA utilizzando la modalità gestita.

Panoramica

Nell'elaborazione della transazione distribuita, le transazioni possono essere completate correttamente. Tuttavia, ci possono essere scenari in cui una transazione potrebbe avere esito negativo per molti motivi. Questi motivi possono includere un errore di sistema, un errore hardware, un errore di rete, dati errati o non validi, errori dell'applicazione o disastri naturali o causati dall'uomo. Non è possibile impedire errori di transazione. Il sistema di transazione distribuito deve essere in grado di gestire questi errori. Deve essere in grado di rilevare e correggere gli errori quando si verificano. Questo processo è noto come ripristino della transazione.

Un aspetto importante di DTP (Distributed Transaction Processing) è il recupero delle transazioni incomplete o in dubbio. È essenziale eseguire il ripristino poiché la parte Unità di lavoro di una particolare transazione viene bloccata fino a quando non viene ripristinata. Microsoft .NET dalla sua libreria di classi

System.Transactions fornisce l'opzione per il ripristino delle transazioni incomplete / in dubbio. Questo supporto di ripristino prevede che il Resource Manager gestisca i log di transazione ed esegua il ripristino quando necessario.

Modello di ripristino

Nel modello di ripristino delle transazioni di Microsoft .NET, il Gestore transazioni (System.Transactionso Microsoft Distributed Transaction coordinator (MS DTC), o entrambi), avvia, coordina e controlla il ripristino delle transazioni. I gestori risorse basati sul protocollo OLE Tx (protocollo XA di Microsoft) forniscono le opzioni per configurare il DTC per guidare, coordinare e controllare il ripristino per loro. A tale scopo, i gestori risorse devono registrare XA_Switch con MS DTC utilizzando l'interfaccia nativa.

XA_Switch fornisce i punti di ingresso delle funzioni XA come xa_start, xa_end e xa_recover nel Resource Manager al Distributed Transaction Coordinator.

Ripristino utilizzando Microsoft DTC (Distributed Transaction coordinator):

Il coordinatore Microsoft Distributed Transaction fornisce due tipi di processi di recupero.

Recupero a freddo

Il ripristino a freddo viene eseguito se il processo del gestore transazioni ha esito negativo mentre è aperta una connessione a un gestore risorse XA. Quando il gestore transazioni viene riavviato, legge i log del gestore transazioni e ristabilisce la connessione al gestore risorse XA, quindi avvia il ripristino.

Ripristino a caldo

Il ripristino a caldo viene eseguito se il gestore transazioni rimane attivo mentre la connessione tra il gestore transazioni e il gestore risorse XA ha esito negativo perché il gestore risorse XA o la rete hanno esito negativo. Dopo l'errore, il gestore transazioni tenta periodicamente di riconnettersi al gestore risorse XA. Quando la connessione viene ristabilita, il gestore transazioni avvia il ripristino XA.

Lo spazio dei nomi System.Transactions fornisce l'implementazione gestita delle transazioni distribuite basate su MS DTC come gestore transazioni. Fornisce funzioni simili a quelle dell'interfaccia nativa di MS DTC, ma in un ambiente completamente gestito. L'unica differenza riguarda il ripristino della transazione. System.Transactions si aspetta che i gestori risorse guidino il recupero da soli e si coordinino con i gestori transazioni (MS DTC). Il Resource Manager deve richiedere il recupero di una particolare transazione incompleta e quindi il Gestore transazioni la accetta e la coordina in base al risultato effettivo di quella particolare transazione.

Processo di recupero transazioni per WebSphere MQ .NET

Questa sezione descrive come è possibile ripristinare le transazioni distribuite con le classi WebSphere MQ .NET.

Panoramica

Per ripristinare una transazione incompleta, sono richieste le informazioni di ripristino. Le informazioni di recupero della transazione devono essere registrate nella memorizzazione dai gestori risorse. Le classi WebSphere MQ .NET seguono un percorso simile. Le informazioni di ripristino della transazione vengono registrate in una coda di sistema denominata SYSTEM.DOTNET.XARECOVERY.QUEUE.

Il ripristino della transazione in WebSphere MQ .NET è un processo in due fasi.

1. Registrazione delle informazioni di ripristino della transazione.
   - Per ogni transazione, durante la fase di preparazione viene aggiunto a SYSTEM.DOTNET.XARECOVERY.QUEUE.
   - Il messaggio viene eliminato se la chiamata di commit ha esito positivo.
2. Ripristino delle transazioni utilizzando un'applicazione di monitoraggio WmqDotnetXAMonitor.
   - WmqDotnetXAMonitor è un'applicazione gestita .NET che elabora i messaggi in SYSTEM.DOTNET.XARECOVERY.QUEUE e recupera le transazioni incomplete

Se l'MCA non è in grado di inserire il messaggio nella coda di destinazione, genera un report di eccezioni contenente il messaggio originale e lo inserisce in una coda di trasmissione da inviare alla coda di risposta specificata nel messaggio originale. (Se la coda di risposta si trova sullo stesso gestore code dell'MCA, il messaggio viene inserito direttamente in quella coda, non in una coda di trasmissione.)

SYSTEM.DOTNET.XARECOVERY.QUEUE

Si tratta di una coda di sistema che contiene le informazioni di ripristino delle transazioni incomplete. Questa coda viene creata quando viene creato un gestore code.

Nota: Non eliminare SYSTEM.DOTNET.XARECOVERY.QUEUE .

Applicazione WMQDotnetXAMonitor

L'applicazione WebSphere MQ .NET XA Monitor monitora un dato gestore code e recupera le transazioni incomplete, se presenti. Le seguenti operazioni sono considerate incomplete e vengono recuperate:

Transazioni incomplete

- Se la transazione è preparata ma COMMIT non è stato completato entro il periodo di timeout.
- Se la transazione è preparata, ma il gestore code WebSphere MQ è stato disattivo.
- Se la transazione è preparata, ma il gestore transazioni è stato disattivo.

  L'applicazione di monitoraggio deve essere eseguita dallo stesso sistema su cui è in esecuzione l'applicazione client WebSphere MQ .NET. Se ci sono applicazioni in esecuzione su più sistemi che si connettono allo stesso gestore code, l'applicazione di monitoraggio deve essere eseguita da tutti i sistemi. Sebbene ogni macchina client disponga di un'applicazione di monitoraggio in esecuzione per ripristinare l'applicazione, ogni monitor dovrebbe essere in grado di identificare il messaggio che corrisponde alla transazione che l'MS DTC locale del monitor corrente stava coordinando in modo da poterla rielencare e completare.

Casi di utilizzo per il ripristino delle transazioni per WebSphere MQ .NET

Di seguito sono riportati i diversi scenari di utilizzo:

- **WebSphere MQ Applicazione che utilizza un singolo DTC e una singola istanza del gestore code:** in questo scenario, quando ci si connette al gestore code ed si esegue l'unità di lavoro (UoW) nella transazione, e se la transazione non riesce e diventa incompleta, l'applicazione di monitoraggio recupera la transazione e la completa.

  In questo scenario, ci sarà una singola istanza dell'applicazione di monitor in esecuzione, poiché un singolo gestore code è associato alle transazioni.
- **Più applicazioni WebSphere MQ utilizzando un singolo DTC e una singola istanza del gestore code:** in questo scenario, esistono più applicazioni WMQ in un singolo DTC e tutte si collegano allo stesso gestore code ed eseguono UoW nelle transazioni.

  Se le transazioni hanno esito negativo e diventano incomplete, l'applicazione di monitoraggio le recupera e completa le transazioni relative a tutte le applicazioni.

  In questo scenario, viene eseguita una singola applicazione di controllo, poiché nelle transazioni viene utilizzato un gestore code.
- **Più applicazioni WebSphere MQ , più DTC, diverse istanze del gestore code:** in questo scenario, vi sono più di un'applicazione WMQ in diversi DTC (ossia, ogni applicazione è in esecuzione su una macchina diversa) e si connette a gestori code differenti.

  Se si verifica un errore e la transazione diventa incompleta, l'applicazione di monitoraggio controlla il TransactionManagerWhereabouts nel messaggio per determinare l'indirizzo DTC. Se il valore di TransactionManagerWhereabouts corrisponde all'indirizzo DTC sotto il quale il monitoraggio è in esecuzione, completa il recupero, altrimenti continua la ricerca fino a quando non viene trovato il messaggio corrispondente al suo DTC.

In questo scenario, ci sarà una sola istanza dell'applicazione di monitoraggio in esecuzione per client (utente o computer) poiché ogni client ha il suo proprio gestore code utilizzato nelle transazioni.

- **Più applicazioni WebSphere MQ , più DTC, più istanze di gestori code:** in questo scenario, esistono più di un'applicazione WMQ in diversi DTC (ossia ogni applicazione è in esecuzione su una macchina differente) e tutte si connettono allo stesso gestore code.

  Se si verifica un errore e la transazione diventa incompleta, l'applicazione di monitoraggio verifica i messaggi TransactionManagerWhereabouts nel messaggio per controllare se l'indirizzo e il valore DTC corrispondono al DTC in cui è in esecuzione il controllo. Se entrambi i valori corrispondono, completa il ripristino, altrimenti continua la ricerca fino a quando non trova il messaggio corrispondente al suo DTC.

  In questo scenario, ci sarà solo una sola istanza dell'applicazione di monitoraggio in esecuzione per client (utente o computer), poiché ogni client ha la propria associazione del gestore code utilizzata nelle transazioni.

- **Più applicazioni WebSphere MQ , DTC singolo, istanze del gestore code differenti:** in questo scenario, esistono più applicazioni WMQ in un singolo DTC (ossia, su un computer, sono in esecuzione più di un'applicazione WMQ) e che si collegano a gestori code differenti.

  Se la transazione ha esito negativo e diventa incompleta, l'applicazione di monitoraggio recupera la transazione.

  In questo scenario, ci saranno tante istanze di applicazione di controllo in esecuzione come gestori code a cui si è connessi, poiché ogni applicazione ha il proprio gestore code utilizzato nelle transazioni e ognuna di esse deve essere ripristinata.

  Nota: Se l'applicazione monitor non è in esecuzione in background, è possibile avviarla.

Utilizzo dell'applicazione WMQDotnetXAMonitor

L'applicazione di monitoraggio XA deve essere eseguita manualmente. Può essere avviato in qualsiasi momento. È possibile avviarlo quando vengono visualizzati i messaggi sul SISTEMA SYSTEM.DOTNET.XARECOVERY.QUEUE oppure è possibile mantenerlo in esecuzione in background prima di eseguire qualsiasi operazione transazionale con le applicazioni scritte utilizzando le classi .NET WebSphere MQ .

Comando per avviare l'applicazione di monitoraggio

```
WmqDotnetXAMonitor.exe -m <QueueManagerName> -n <ConnectionName> -c <Channel> -i
```

Dove

- **n** - Nome connessione in formato host (porta). Il nome connessione può contenere più di un nome connessione. È necessario fornire più nomi di connessione nell'elenco separato da virgole, ad esempio "localhost (1414), localhost (1415), localhost (1416)". L'applicazione di monitoraggio esegue il ripristino per ciascuno dei nomi di connessione specificati nell'elenco separato da virgole.
- **c** - Nome canale.
- **m** - Nome gestore code. Facoltativo
- **i** - Completamento ramo euristico. Facoltativo

L'applicazione di monitoraggio esegue le azioni riportate di seguito:

1. Verifica la profondità della coda di SYSTEM.DOTNET.XARECOVERY.QUEUE ad un intervallo di 100 secondi.
2. Se la profondità della coda è maggiore di zero, il controllo XA esamina la coda per individuare i messaggi e verifica se il messaggio soddisfa i criteri di transazione incompleti.
3. Se uno qualsiasi dei messaggi soddisfa i criteri di transazione incompleti, il monitoraggio lo estrae e richiama le informazioni di ripristino della transazione.
4. Determina quindi se le informazioni di ripristino sono relative al MS DTC locale. Se sì, procede al recupero della transazione. Altrimenti torna indietro per sfogliare il messaggio successivo.
5. Quindi, effettua chiamate al gestore code per ripristinare la transazione incompleta.

Impostazioni del file di configurazione dell'applicazione WmqDotNETXAMonitor
Per monitorare l'applicazione, è possibile fornire gli input anche utilizzando il file di configurazione dell'applicazione. Un file di configurazione dell'applicazione di esempio viene fornito con WebSphere MQ .NET. Questo file può essere modificato in base alle proprie esigenze.

Il file di configurazione dell'applicazione ha la precedenza più alta considerando i valori di input. Se vengono forniti valori di input sia sulla riga comandi che sul file di configurazione dell'applicazione, vengono considerati i valori della configurazione dell'applicazione.

File di configurazione dell'applicazione di esempio.

```
<?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>
```

Log dell'applicazione WmqDotNetXAMonitor
L'applicazione Monitor crea un file di log nella directory dell'applicazione per registrare l'avanzamento del monitoraggio e lo stato di ripristino della transazione. La registrazione inizia con il nome della connessione e i dettagli del canale per mostrare il gestore code corrente per cui è in esecuzione il recupero.

Una volta avviato il ripristino, verrà registrato MessageId del messaggio di ripristino della transazione, TransactionId della transazione incompleta e del risultato effettivo della transazione come per Transaction Manager Coordination.

File di log di esempio:

```
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
```

Utilizzo delle classi WebSphere MQ per .NET

Questa raccolta di argomenti descrive come configurare il sistema per eseguire i programmi di esempio per verificare l'installazione delle classi WebSphere MQ per .NET e come eseguire i propri programmi.

Configurazione del gestore code per l'accettazione delle connessioni client TCP/IP

Per configurare un gestore code per accettare le richieste di connessione in entrata dai client:

1. Definire un canale di connessione server:
    a. Avviare il gestore code.

b. Definire un canale di esempio denominato NET.CHANNEL[3]:

```
DEF CHL('NET.CHANNEL') CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ') +
DESCR('Sample channel for WebSphere MQ classes for .NET')
```

2. Avviare un listener:

```
runmqlsr -t tcp [-m qmnqme] [-p portnum]
```

Nota: Le parentesi quadre indicano parametri facoltativi; *qmname* non è richiesto per il gestore code predefinito e il numero di porta *portnum* non è richiesto se si utilizza il valore predefinito (1414).

Applicazioni di esempio

Per eseguire le proprie applicazioni .NET, utilizzare le istruzioni per i programmi di verifica, sostituendo il nome dell'applicazione con le applicazioni di esempio.

Sono fornite cinque applicazioni di esempio:

- Un'applicazione di inserimento messaggi
- Un'applicazione di richiamo messaggi
- Un'applicazione 'hello world'
- Un'applicazione di pubblicazione / sottoscrizione
- Un'applicazione che utilizza le proprietà del messaggio

Tutte queste applicazioni di esempio sono fornite nel linguaggio C# e alcune sono fornite anche in C++ e Visual Basic. È possibile scrivere applicazioni in qualsiasi linguaggio supportato da .NET.

Programma "Put message" SPUT (nmqsput.cs, mmqsput.cpp, vmqsput.vb)

Questo programma mostra come inserire un messaggio in una coda denominata. Il programma ha tre parametri:

- Il nome di una coda (obbligatorio), ad esempio SYSTEM.DEFAULT.LOCAL.QUEUE
- Il nome di un gestore code (facoltativo)
- La definizione di un canale (facoltativo), ad esempio, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Se non viene fornito alcun nome gestore code, il gestore code assume il valore predefinito del gestore code locale predefinito. Se un canale è definito, ha lo stesso formato della variabile di ambiente MQSERVER.

Programma "Get message" SGET (nmqsget.cs, mmqsget.cpp, vmqsget.vb)

Questo programma mostra come richiamare un messaggio da una coda denominata. Il programma ha tre parametri:

- Il nome di una coda (obbligatorio), ad esempio SYSTEM.DEFAULT.LOCAL.QUEUE
- Il nome di un gestore code (facoltativo)
- La definizione di un canale (facoltativo), ad esempio, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Se non viene fornito alcun nome gestore code, il gestore code assume il valore predefinito del gestore code locale predefinito. Se un canale è definito, ha lo stesso formato della variabile di ambiente MQSERVER.

Programma "Hello World" (nmqwrld.cs, mmqwrld.cpp, vmqwrld.vb)

Questo programma mostra come inserire e richiamare un messaggio. Il programma ha tre parametri:

- Il nome di una coda (facoltativo), ad esempio, SYSTEM.DEFAULT.LOCAL.QUEUE o SYSTEM.DEFAULT.MODEL.QUEUE
- Il nome di un gestore code (facoltativo)

[3] In questo esempio, non vengono considerate implicazioni di sicurezza. Per un sistema di produzione, utilizzare SSL o un'uscita di sicurezza. Consultare Sicurezza per ulteriori informazioni.

- Una definizione di canale (facoltativa), ad esempio SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Se non viene fornito alcun nome coda, il nome assume il valore predefinito SYSTEM.DEFAULT.LOCAL.QUEUE. Se non viene fornito alcun nome gestore code, il gestore code assume il valore predefinito del gestore code locale predefinito.

Programma "Publish/subscribe" (MQPubSubSample.cs)
Questo programma mostra come utilizzare la pubblicazione / sottoscrizione di WebSphere MQ . Viene fornito solo in C#. Il programma ha due parametri:

- Il nome di un gestore code (facoltativo)
- Una definizione di canale (facoltativo)

Programma "Message properties" (MQMessagePropertiesSample.cs)
Questo programma mostra come utilizzare le proprietà del messaggio. Viene fornito solo in C#. Il programma ha due parametri:

- Il nome di un gestore code (facoltativo)
- Una definizione di canale (facoltativo)

È possibile verificare l'installazione compilando ed eseguendo queste applicazioni.

Le applicazioni di esempio vengono installate nelle seguenti ubicazioni, in base alla lingua in cui sono scritte. *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato 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

C++ gestito

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

Per creare le applicazioni di esempio, è stato fornito un file batch per ogni lingua.

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\bldcssamp.bat

Il file bldcssamp.bat contiene una riga per ciascun esempio, che è tutto ciò che è necessario per creare questo programma di esempio:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib:MQ_INSTALLATION_PATH\bin
/out:nmqwrld.exe nmqwrld.cs
```

C++ gestito

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\bldmcpsamp.bat

Il file bldmcpsamp.bat contiene una riga per ciascun esempio, che è tutto ciò che è necessario per creare questo programma di esempio:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Se si desidera compilare queste applicazioni su Microsoft Visual Studio 2003/.NET SDKv1.1, sostituire il comando di compilazione:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

con

```
cl /clr MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Visual Basic

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\bldvbsamp.bat

Il file bldvbsamp.bat contiene una riga per ciascun esempio, che è tutto ciò che è necessario per creare questo programma di esempio:

```
vbc /r:System.dll /r:MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:vmqwrld.exe vmqwrld.vb
```

Risoluzione dei problemi relativi a WebSphere MQ .NET

Se un programma non viene completato correttamente, eseguire una delle applicazioni di esempio e seguire i consigli riportati nei messaggi di diagnostica.

Queste applicazioni di esempio sono descritte in "Utilizzo delle classi WebSphere MQ per .NET" a pagina 574.

Se il problema persiste e è necessario contattare il team di assistenza IBM , potrebbe essere richiesto di attivare la funzionalità di traccia.

Traccia dell'applicazione di esempio

Per istruzioni sull'utilizzo della funzionalità di traccia, fare riferimento a "Traccia dei programmi WebSphere MQ .NET" a pagina 597.

Messaggi di errore

È possibile che venga visualizzato il seguente messaggio di errore comune:

Un'eccezione non gestita di tipo 'System.IO.FileNotFoundException' nel modulo sconosciuto

Se questo errore si verifica per amqmdnet.dll o amqmdxcs.dll, assicurarsi che entrambi siano registrati nella 'Global Assembly Cache' o creare un file di configurazione che punti agli assembly amqmdnet.dll e amqmdxcs.dll . È possibile esaminare e modificare il contenuto della cache di assemblaggio utilizzando mscorcfg.msc, che viene fornito come parte del framework .NET.

Se .NET Framework non era disponibile quando WebSphere MQ è stato installato, le classi potrebbero non essere registrate nella cache di assemblaggio globale. È possibile eseguire di nuovo manualmente il processo di registrazione utilizzando il comando

```
amqidnet -c MQ_INSTALLATION_PATH\bin\amqidotn.txt -l logfile.txt
```

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Le informazioni su questa installazione vengono scritte nel file di log specificato (`logfile.txt` in questo esempio).

Scrittura e distribuzione di programmi WebSphere MQ .NET

Per utilizzare le classi WebSphere MQ per .NET per accedere alle code WebSphere MQ , scrivere i programmi in qualsiasi linguaggio supportato da .NET contenente le chiamate che inserono i messaggi e richiamano i messaggi dalle code WebSphere MQ .

La documentazione WebSphere MQ contiene informazioni solo sui linguaggi C#, C++ e Visual Basic.

Questa raccolta di argomenti fornisce informazioni utili per la scrittura di applicazioni per interagire con i sistemi WebSphere MQ . Per dettagli sulle singole classi, consultare Le classi e le interfacce WebSphere MQ .NET.

Differenze di connessione

Il modo in cui si programma per WebSphere MQ .NET ha alcune dipendenze sulle modalità di connessione che si desidera utilizzare.

Connessioni client gestite

Quando le classi WebSphere MQ per .NET vengono utilizzate come client gestito, esistono diverse differenze rispetto a un client MQI standard di WebSphere MQ .

Le seguenti funzioni non sono disponibili per un client gestito:

- Compressione canale
- Supporto SSL
- Concatenamento uscita canale

Se si tenta di utilizzare queste funzioni con un client gestito, verrà restituita un'eccezione MQException. Se l'errore viene rilevato alla fine del client di una connessione, utilizzerà il codice di errore MQRC_ENVIRONMENT_ERROR. Se viene rilevato all'estremità del server, verrà utilizzato il codice di errore restituito dal server.

Le uscite del canale scritte per un client non gestito non funzionano. È necessario scrivere nuove uscite specificamente per il client gestito. Verificare che non vi siano uscite canale non valide specificate nella tabella di definizione del canale client (CCDT).

Il nome di un'uscita canale gestito può avere una lunghezza massima di 999 caratteri. Tuttavia, se si utilizza la CCDT per specificare il nome dell'uscita del canale, è limitato a 128 caratteri.

La comunicazione è supportata solo su TCP/IP.

Quando si arresta il gestore code utilizzando il comando **endmqm** , la chiusura di un canale di connessione server a un client gestito .NET può richiedere più tempo rispetto ai canali di connessione server ad altri client.

Se *NMQ_MQ_LIB* è stato impostato su `gestito` per utilizzare la diagnostica dei problemi di WebSphere MQ , nessuno dei parametri -i, -p, -s, -b o -c del comando **strmqtrc** è supportato.

Un'applicazione .NET gestita che utilizza transazioni XA non funzionerà con un gestore code z/OS . Un client gestito. Net che sta tentando di connettersi a un gestore code z/OS ha esito negativo con un errore, MQRC_UOW_ENLISTMENT_ERROR (mqrc=2354), nella chiamata MQOPEN. Tuttavia, un'applicazione .NET gestita che utilizza transazioni XA funzionerà con il gestore code distribuito.

Definizione del tipo di connessione da utilizzare

Il tipo di connessione è determinato dall'impostazione del nome della connessione, del nome canale, del valore di personalizzazione NMQ_MQ_LIB e della proprietà MQC.TRANSPORT_PROPERTY.

È possibile specificare il nome della connessione nel modo seguente:

- Esplicitamente su un costruttore MQQueueManager :

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- Impostando le proprietà MQC.HOST_NAME_PROPERTY e, facoltativamente, MQC.PORT_PROPERTY in una voce hashtable su un costruttore MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Come valori MQEnvironment espliciti

```
MQEnvironment.Hostname
```

`MQEnvironment.Port`(facoltativo).

- Impostando le proprietà MQC.HOST_NAME_PROPERTY e, facoltativamente, MQC.PORT_PROPERTY nell'hashtable MQEnvironment.properties .

È possibile specificare il nome del canale come segue:

- Esplicitamente su un costruttore MQQueueManager :

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- Impostando la proprietà MQC.CHANNEL_PROPERTY in una voce hashtable su un costruttore di MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Come valore MQEnvironment esplicito

```
MQEnvironment.Channel
```

- Impostando la proprietà MQC.CHANNEL_PROPERTY nell'hashtable MQEnvironment.properties .

È possibile specificare la proprietà di trasporto come segue:

- Impostando la proprietà MQC.TRANSPORT_PROPERTY in una voce hashtable su un costruttore MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Impostando la proprietà MQC.TRANSPORT_PROPERTY nell'hashtable MQEnvironment.properties .

Selezionare il tipo di connessione richiesto utilizzando uno dei valori riportati di seguito:

MQC.TRANSPORT_MQSERIES_BINDINGS - connettersi come server
MQC.TRANSPORT_MQSERIES_CLIENT - connettersi come client non XA
MQC.TRANSPORT_MQSERIES_XACLIENT - connettersi come client XA
MQC.TRANSPORT_MQSERIES_MANAGED - connettersi come client gestito non XA

È possibile impostare il valore di personalizzazione NMQ_MQ_LIB per scegliere esplicitamente il tipo di connessione come mostrato nella seguente tabella

Valore NMQ_MQ_LIB	**Tipo di connessione**
mqic.dll	Connetti come client non XA
mqicxa.dll	Connetti come client XA

Valore NMQ_MQ_LIB	Tipo di connessione
mqm.dll	Connetti come server o come client non XA
Gestito	Connetti come client gestito non XA
Nota: I valori di mqic32.dll e mqic32xa.dll sono accettati come sinonimi di mqic.dll e mqicxa.dll per la compatibilità con le release precedenti. Tuttavia, mqm.dll e mqm.pdb sono solo parte del pacchetto client dalla versione 7.1 in poi.	

Se si sceglie un tipo di connessione non disponibile nel proprio ambiente, ad esempio si specifica mqic32xa.dll e non si dispone del supporto XA, WebSphere MQ .NET genera un'eccezione.

L'impostazione di NMQ_MQ_LIB su "gestito" fa sì che il client utilizzi le funzioni WebSphere MQ per la diagnostica dei problemi, la conversione dei dati .NET e altre funzioni WebSphere MQ di basso livello.

Tutti gli altri valori per NMQ_MQ_LIB fanno sì che il processo .NET utilizzi funzioni WebSphere MQ non gestite per le verifiche diagnostiche dei problemi e la conversione dei dati e altre funzioni WebSphere MQ di basso livello non gestite (supponendo che sul sistema sia installato un client o un server MQI WebSphere MQ).

WebSphere MQ .NET sceglie il tipo di connessione come segue:

1. Se MQC.TRANSPORT_PROPERTY è specificato, si connette in base al valore di MQC.TRANSPORT_PROPERTY.

   Notare, tuttavia, che l'impostazione di MQC.TRANSPORT_PROPERTY in MQC.TRANSPORT_MQSERIES_MANAGED non garantisce che il processo client venga eseguito gestito. Anche con questa impostazione, il cliente non viene gestito nei seguenti casi:

   - Se un altro thread nel processo si è collegato a MQC.TRANSPORT_PROPERTY impostato su un valore diverso da MQC.TRANSPORT_MQSERIES_MANAGED.
   - Se NMQ_MQ_LIB non è impostato su "gestito", le verifiche diagnostiche dei problemi, la conversione dei dati e altre funzioni di basso livello non sono completamente gestite (supponendo che un client o un server WebSphere MQ MQI sia installato sul sistema).
2. Se un nome di connessione è stato specificato senza un nome di canale o un nome di canale è stato specificato senza un nome di connessione, viene generato un errore.
3. Se sono stati specificati sia un nome connessione che un nome canale:
   - Se NMQ_MQ_LIB è impostato su mqic32xa.dll, si collega come un client XA.
   - Se NMQ_MQ_LIB è impostato su gestito, si connette come client gestito.
   - Altrimenti si connette come un client non XA.
4. Se viene specificato NMQ_MQ_LIB, si connette in base al valore di NMQ_MQ_LIB.
5. Se è installato un server WebSphere MQ , si connette come server.
6. Se è installato un client WebSphere MQ MQI, si connette come un client non XA.
7. Altrimenti, si connette come client gestito.

File di configurazione per le classi WebSphere MQ per .NET

Un'applicazione client .NET può utilizzare un file di configurazione WebSphere MQ MQI e, se si utilizza il tipo di connessione gestita, un file di configurazione dell'applicazione .NET. Le impostazioni nel file di configurazione dell'applicazione hanno priorità.

File di configurazione client

Un'applicazione client WebSphere MQ può utilizzare un file di configurazione client nello stesso modo di qualsiasi altro client WebSphere MQ MQI. Questo file è in genere denominato mqclient.ini, ma è possibile specificare un nome file diverso. Per ulteriori informazioni sul file di configurazione client, consultare

Configurazione di un client mediante un file di configurazione WebSphere MQ File di configurazione client MQI.

Solo i seguenti attributi in un WebSphere MQ file di configurazione del client MQI sono rilevanti per WebSphere MQ classes per .NET. Se si specificano altri attributi, non ha alcun effetto.

stanza	**Attributo**
Canali	CCSID
Canali	Directory ChannelDefinition
Canali	File ChannelDefinition
Canali	Parametri ServerConnection
ClientExitClientExit	ExitsDefaultPath
ClientExitClientExit	ExitsDefaultPath64
MessageBuffer	MaximumSize
MessageBuffer	PurgeTime
MessageBuffer	UpdatePercentage
TCP	ClntRcvBufSize
TCP	ClntSndBufSize
TCP	IPAddressVersion
TCP	KeepAlive

È possibile sovrascrivere uno qualsiasi di questi attributi utilizzando la variabile di ambiente appropriata.

File di configurazione dell'applicazione

Se si è in esecuzione con il tipo di connessione gestita, è anche possibile sovrascrivere il file di configurazione del client WebSphere MQ e le variabili di ambiente equivalenti utilizzando il file di configurazione dell'applicazione .NET.

Le impostazioni del file di configurazione dell'applicazione .NET vengono utilizzate solo durante l'esecuzione con tipo di connessione gestita e vengono ignorate per altri tipi di connessione.

Il file di configurazione dell'applicazione .NET e il relativo formato sono definiti da Microsoft per un utilizzo generale all'interno di .NET Framework, ma i particolari nomi di sezione, chiavi e valori menzionati in questa documentazione sono specifici di Websphere MQ.

Il formato del file di configurazione dell'applicazione .NET è un numero di *sezioni*. Ogni sezione contiene una o più *chiavi*e ogni chiave ha un *valore*associato. Il seguente esempio mostra le sezioni, chiavi e valori utilizzati in un file di configurazione dell'applicazione .NET per controllare la proprietà TCP/IP KeepAlive :

```
<configuration>
  <configSections>
    <section name="TCP" type="System.Configuration.NameValueSectionHandler"/>
  </configSections>
  <TCP>
    <add key="KeepAlive" value="true"></add>
  </TCP>
<configuration>
```

Le parole chiave utilizzate nei nomi e nelle chiavi della sezione del file di configurazione dell'applicazione .NET corrispondono esattamente alle parole chiave per le Stanze e gli Attributi definiti nel file di configurazione client.

Per ulteriori informazioni, consultare la documentazione Microsoft .

Frammento di codice di esempio

Il seguente frammento di codice C# illustra un'applicazione che esegue tre operazioni:

1. Connetterti a un gestore code
2. Inserire un messaggio in SYSTEM.DEFAULT.LOCAL.QUEUE
3. Richiama il messaggio

Mostra anche come modificare il tipo di connessione.

```
// ============================================================================
// 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
```

Operazioni sui gestori code

Questa sezione descrive come connettersi e disconnettersi da un gestore code utilizzando le classi WebSphere MQ per .NET.

Impostazione dell'ambiente WebSphere MQ

Prima di utilizzare la connessione client per connettersi a un gestore code, è necessario configurare l'ambiente WebSphere MQ .

Nota: Questo passo non Š necessario quando si utilizzano le classi WebSphere MQ per .NET in modalit ... di bind del server.

L'interfaccia di programmazione .NET consente di utilizzare il valore di personalizzazione di NMQ_MQ_LIB ma include anche una classe MQEnvironment. Questa classe consente di specificare i dettagli da utilizzare durante il tentativo di collegamento, come quelli nel seguente elenco:

- Nome canale
- Nome host
- Numero di porta
- Uscite canale
- Parametri SSL
- ID utente e password

Per informazioni complete sulla classe MQEnvironment, consultare `MQEnvironment` .NET class

Per specificare il nome canale e il nome host, utilizzare il codice seguente:

```
MQEnvironment.Hostname = "host.domain.com";
MQEnvironment.Channel  = "client.channel";
```

Per impostazione predefinita, i client tentano di connettersi a un listener WebSphere MQ sulla porta 1414. Per specificare una porta differente, utilizzare il codice:

```
MQEnvironment.Port = nnnn;
```

Connessione a un gestore code

Sei ora pronto a connetterti a un gestore code creando una nuova istanza della classe MQQueueManager :

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Per disconnettersi da un gestore code, richiamare il metodo `Disconnect` sul gestore code:

```
queueManager.Disconnect();
```

È necessario disporre dell'autorizzazione di interrogazione (`inq`) sul gestore code durante il tentativo di connessione al gestore code. Senza richiedere l'autorizzazione, il tentativo di connessione non riesce.

Se si richiama il metodo `Disconnect` , tutte le code e i processi aperti a cui è stato effettuato l'accesso tramite tale gestore code vengono chiusi. Tuttavia, è buona pratica di programmazione chiudere esplicitamente queste risorse quando si finisce di utilizzarle. Per chiudere le risorse, utilizzare il metodo `Close` sull'oggetto associato a ciascuna risorsa.

I metodi di `Commit` e `Backout` su un gestore code sostituiscono le chiamate MQCMIT e MQBACK utilizzate con l'interfaccia procedurale.

Accesso a code e argomenti

È possibile accedere alle code e agli argomenti utilizzando i metodi di MQQueueManager o i costruttori appropriati.

Per accedere alle code, utilizzare i metodi della classe MQQueueManager . MQOD (object descriptor structure) è compresso nei parametri di questi metodi. Ad esempio, per aprire una coda su un gestore code rappresentato da un oggetto MQQueueManager denominato queueManager, utilizzare il seguente codice:

```
MQQueue queue = queueManager.AccessQueue("qName",
                                         MQC.MQOO_OUTPUT,
                                         "qMgrName",
```

```
                                        "dynamicQName",
                                        "altUserId");
```

Il parametro *options* è uguale al parametro Options nella chiamata MQOPEN.

Il metodo AccessQueue restituisce un nuovo oggetto della classe MQQueue.

Una volta terminato di utilizzare la coda, utilizzare il metodo Close () per chiuderla, come nel seguente esempio:

```
queue.Close();
```

Con WebSphere MQ .NET, è possibile anche creare una coda utilizzando il costruttore MQQueue. I parametri sono esattamente gli stessi del metodo accessQueue , con l'aggiunta di un parametro del gestore code che specifica l'oggetto MQQueueManager istanziato da utilizzare. Ad esempio:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            MQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserId");
```

La creazione di un oggetto coda in questo modo consente di scrivere le proprie sottoclassi di MQQueue.

Allo stesso modo, è anche possibile accedere agli argomenti utilizzando i metodi della classe MQQueueManager . Utilizzare un metodo AccessTopic() per aprire un argomento. Viene restituito un nuovo oggetto della classe MQTopic. Una volta terminato di utilizzare l'argomento, utilizzare il metodo Close () di MQTopic per chiuderlo.

È anche possibile creare un argomento utilizzando un costruttore MQTopic. Esistono diversi costruttori per gli argomenti; per ulteriori informazioni, consultare MQTopic Classe .NET.

Gestione dei messaggi

I messaggi vengono gestiti utilizzando i metodi delle classi coda o argomento. Per creare un nuovo messaggio, creare un nuovo MQMessageobject.

Inserire i messaggi nelle code o negli argomenti utilizzando il metodo Put () della classe MQQueue o MQTopic. Richiamare i messaggi dalle code o dagli argomenti utilizzando il metodo Get () della classe MQQueue o MQTopic. A differenza dell'interfaccia procedurale, dove MQPUT e MQGET immettono e ottengono array di byte, le classi WebSphere MQ per .NET immettono e ottengono istanze della classe MQMessage. La classe MQMessage incapsula il buffer di dati che contiene i dati del messaggio effettivi, insieme a tutti i parametri MQMD (message descriptor) che descrivono quel messaggio.

Per creare un nuovo messaggio, creare una nuova istanza della classe MQMessage e utilizzare i metodi WriteXXX per inserire i dati nel buffer di messaggio.

Quando viene creata la nuova istanza di messaggio, tutti i parametri MQMD vengono impostati automaticamente sui loro valori predefiniti, come definito in Valori iniziali e dichiarazioni della lingua per MQMD . Il metodo Put () di MQQueue utilizza anche un'istanza della classe MQPutMessageOptions come parametro. Questa classe rappresenta la struttura MQPMO. Il seguente esempio crea un messaggio e lo inserisce in una coda:

```
// 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);
```

Il metodo Get () di MQQueue restituisce una nuova istanza di MQMessage, che rappresenta il messaggio appena preso dalla coda. Inoltre, utilizza un'istanza della classe di opzioni MQGetMessagecome parametro. Questa classe rappresenta la struttura MQGMO.

Non è necessario specificare una dimensione massima del messaggio, poiché il metodo Get () regola automaticamente la dimensione del relativo buffer interno per adattarla al messaggio in arrivo. Utilizzare i metodi ReadXXX della classe MQMessage per accedere ai dati nel messaggio restituito.

Il seguente esempio mostra come ottenere un messaggio da una coda:

```
// 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();
```

È possibile modificare il formato numerico utilizzato dai metodi di lettura e scrittura impostando la variabile membro *codifica* .

È possibile modificare la serie di caratteri da utilizzare per la lettura e la scrittura delle stringhe impostando la variabile membro *characterSet* .

Per ulteriori dettagli, consultare MQMessage .NET class .

Nota: Il metodo WriteUTF() di MQMessage codifica automaticamente la lunghezza della stringa e i byte Unicode che contiene. Quando il messaggio verrà letto da un altro programma .NET (utilizzando ReadUTF()), questo è il modo più semplice per inviare informazioni sulla stringa.

Gestione delle proprietà dei messaggi

Le proprietà del messaggio consentono di selezionare i messaggi o di richiamare le informazioni su un messaggio senza accedervi. La classe MQMessage contiene i metodi per richiamare e impostare le proprietà.

È possibile utilizzare le proprietà del messaggio per consentire a un'applicazione di selezionare i messaggi da elaborare o per richiamare le informazioni relative a un messaggio senza accedere alle intestazioni MQMD o MQRFH2 . Inoltre, facilitano la comunicazione tra applicazioni WebSphere MQ e JMS. Per ulteriori informazioni sulle proprietà dei messaggi in WebSphere MQ, consultare Proprietà del messaggio.

La classe MQMessage fornisce una quantità di metodi per richiamare e impostare le proprietà, in base al tipo di dati della proprietà. I metodi get hanno i nomi del formato Get * Property e i metodi set hanno i nomi del formato Set * Property, dove l'asterisco (*) rappresenta una delle seguenti stringhe:

- Booleano
- Byte
- Byte
- Doppia
- Mobile
- Int
- Int2
- Int4
- Int8
- Lungo
- Oggetto
- Breve

- Stringa

Ad esempio, per ottenere la proprietà WebSphere MQ myproperty (una stringa di caratteri), utilizzare la chiamata `message.GetStringProperty('myproperty')`. È possibile, facoltativamente, passare un descrittore di proprietà, che verrà completato da WebSphere MQ .

Gestione degli errori

Gestire gli errori derivanti dalle classi WebSphere MQ for .NET utilizzando i blocchi `try` e `catch` .

I metodi nell'interfaccia .NET non restituiscono un codice motivo e di completamento. Al contrario, generano un'eccezione ogni volta che il codice di completamento e il codice motivo risultanti da una chiamata WebSphere MQ non sono entrambi zero. Ciò semplifica la logica del programma in modo da non dover controllare i codici di ritorno dopo ogni chiamata a WebSphere MQ. È possibile decidere in quali punti del programma si desidera affrontare la possibilità di errore. In questi punti, puoi racchiudere il tuo codice con blocchi `try` e `catch` , come nel seguente esempio:

```
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 );
}
```

Richiamo e impostazione dei valori di attributo

Le classi MQManagedObject, MQQueue e MQQueueManager contengono metodi che consentono di richiamare e impostare i loro valori di attributo. Tenere presente che per MQQueue, i metodi funzionano solo se si specifica l'interrogazione appropriata e si impostano gli indicatori quando si apre la coda.

Per gli attributi comuni, le classi MQQueueManager e MQQueue ereditano da una classe denominata MQManagedObject. Questa classe definisce le interfacce Inquire () e Set ().

Quando si crea un nuovo oggetto gestore code utilizzando l'operatore *nuovo* , viene aperto automaticamente per l'interrogazione. Quando si utilizza il metodo AccessQueue() per accedere a un oggetto coda, tale oggetto *non* viene aperto automaticamente per le operazioni inquire o set, ciò potrebbe causare problemi con alcuni tipi di code remote. Per utilizzare i metodi Inquire e Set e per impostare le proprietà su una coda, è necessario specificare gli indicatori inquire e set appropriati nel parametro openOptions del metodo AccessQueue().

I metodi inquire e set utilizzano tre parametri:

- array selettori
- array intAttrs
- Array charAttrs

Non sono necessari i parametri SelectorCount, IntAttrCount e CharAttrLength rilevati in MQINQ, poiché la lunghezza di un array è sempre nota. Il seguente esempio mostra come eseguire un'interrogazione su una coda:

```
//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);
```

Tutti gli attributi di questi oggetti possono essere interrogati. Un sottoinsieme di attributi viene esposto come proprietà di un oggetto. Per un elenco di attributi oggetto, consultare Attributi degli oggetti. Per le proprietà dell'oggetto, consultare la descrizione della classe appropriata.

Programmi a più sottoprocessi

L'ambiente di runtime .NET è intrinsecamente multithread. WebSphere MQ classes per .NET consente a un oggetto gestore code di essere condiviso tra più thread, ma garantisce che tutti gli accessi al gestore code di destinazione siano sincronizzati.

Considerare un programma semplice che si connette a un gestore code e apre una coda all'avvio. Il programma visualizza un singolo pulsante sullo schermo. Quando un utente fa clic su quel pulsante, il programma richiama un messaggio dalla coda. In questa situazione, l'inizializzazione dell'applicazione si verifica in un thread e il codice che viene eseguito in risposta alla pressione del pulsante viene eseguito in un thread separato (il thread dell'interfaccia utente).

L'implementazione di WebSphere MQ .NET garantisce che, per un particolare collegamento (istanza oggettoMQQueueManager), tutti gli accessi al gestore code WebSphere MQ di destinazione siano sincronizzati. Il comportamento predefinito è che un thread che desidera emettere una chiamata a un gestore code viene bloccato fino a quando non vengono completate tutte le chiamate in corso per tale connessione. Se si richiede l'accesso simultaneo allo stesso gestore code da più thread all'interno del programma, creare un nuovo oggetto MQQueueManager per ogni thread che richiede l'accesso simultaneo. (Ciò equivale all'emissione di una chiamata MQCONN separata per ciascun thread.)

Se le opzioni di connessione predefinite vengono sovrascritte da MQC.MQCNO_HANDLE_SHARE_NONE o MQC.MQCNO_SHARE_NO_BLOCK il gestore code non è più sincronizzato.

Utilizzo di una tabella di definizione di canale client con .NET

È possibile utilizzare una CCDT (Client Channel Definition Table) con le classi .NET per WebSphere MQ. L'ubicazione della CCDT viene specificata in modi diversi, a seconda che si stia utilizzando una connessione gestita o non gestita.

Tipo di connessione client non gestito XA o non XA

Con un tipo di connessione non gestito, è possibile specificare l'ubicazione della CCDT in due modi:

- Utilizzo delle variabili di ambiente MQCHLLIB per specificare la directory in cui si trova la tabella e MQCHLTAB per specificare il nome file della tabella.
- Utilizzo del file di configurazione client. Nella stanza CHANNELS, utilizzare gli attributi ChannelDefinitionDirectory per specificare la directory in cui si trova la tabella e ChannelDefinitionFile per specificare il nome file.

Se l'ubicazione è specificata sia nel file di configurazione client che utilizzando le variabili di ambiente, le variabili di ambiente hanno la priorità. È possibile utilizzare questa funzione per specificare un'ubicazione standard nel file di configurazione del client e sovrascriverla utilizzando le variabili di ambiente quando necessario.

Tipo di connessione client gestito

Con un tipo di collegamento gestito, è possibile specificare l'ubicazione della CCDT in tre modi:

- Utilizzo del file di configurazione dell'applicazione .NET. Nella sezione CHANNELS, utilizzare le chiavi ChannelDefinitionDirectory per specificare la directory in cui si trova la tabella e ChannelDefinitionFile per specificare il nome file.
- Utilizzo delle variabili di ambiente MQCHLLIB per specificare la directory in cui si trova la tabella e MQCHLTAB per specificare il nome file della tabella.

- Utilizzo del file di configurazione client. Nella stanza CHANNELS, utilizzare gli attributi ChannelDefinitionDirectory per specificare la directory in cui si trova la tabella e ChannelDefinitionFile per specificare il nome file.

Se la posizione è specificata in più di uno di questi modi, le variabili di ambiente hanno la priorità sul file di configurazione del client e il file di configurazione dell'applicazione .NET ha la priorità su entrambi gli altri metodi. È possibile utilizzare questa funzione per specificare un'ubicazione standard nel file di configurazione del client e sovrascriverla utilizzando le variabili di ambiente o il file di configurazione dell'applicazione quando necessario.

Come un'applicazione .NET determina quale definizione di canale utilizzare

Nell'ambiente client WebSphere MQ .NET, la definizione di canale da utilizzare può essere specificata in diversi modi. Possono esistere più specifiche della definizione del canale. Un'applicazione deriva la definizione del canale da una o più origini.

Se esiste più di una definizione di canale, quella utilizzata viene selezionata nel seguente ordine di priorità:

1. Proprietà specificate nel costruttore MQQueueManager , esplicitamente o includendo *MQC.CHANNEL_PROPERTY* nella tabella hash delle proprietà
2. Una proprietà *MQC.CHANNEL_PROPERTY* in MQEnvironment.properties hashtable
3. La proprietà *Canale* in MQEnvironment
4. Il file di configurazione dell'applicazione .NET, nome sezione CHANNELS, chiave ServerConnectionParms (si applica solo alle connessioni gestite)
5. La variabile di ambiente *MQSERVER*
6. Il file di configurazione del client, stanza CHANNELS, attributo ServerConnectionParms
7. La tabella CCDT (client channel definition table). L'ubicazione di CCDT è specificata nel file di configurazione dell'applicazione .NET (si applica solo alle connessioni gestite)
8. La tabella CCDT (client channel definition table). L'ubicazione della CCDT viene specificata utilizzando le variabili di ambiente *MQCHLIB* e *MQCHLTAB*
9. La tabella CCDT (client channel definition table). L'ubicazione di CCDT è specificata utilizzando il file di configurazione client

Per gli elementi 1-3, la definizione di canale viene creata campo per campo dai valori forniti dall'applicazione. Questi valori possono essere forniti utilizzando interfacce differenti e possono esistere più valori per ciascuno di essi. I valori dei campi vengono aggiunti alla definizione di canale seguendo l'ordine di priorità fornito:

1. Il valore di *connName* sul costruttore MQQueueManager
2. Valori delle proprietà dalla tabella hash MQQueueManager.properties
3. Valori delle proprietà dalla tabella hash MQEnvironment.properties
4. Valori impostati come campi MQEnvironment (ad esempio, MQEnvironment.Hostname, MQEnvironment.Port)

Per gli elementi 4-6, l'intera definizione di canale viene fornita come valore. I campi non specificati nella definizione di canale assumono i valori predefiniti del sistema. Nessun valore proveniente da altri metodi di definizione dei canali e dei relativi campi viene unito a tali specifiche.

Per gli articoli 7-9, l'intera definizione di canale viene presa dalla CCDT. I campi non specificati esplicitamente quando il canale è stato definito assumono i valori predefiniti del sistema. Nessun valore proveniente da altri metodi di definizione dei canali e dei relativi campi viene unito a tali specifiche.

Utilizzo delle uscite di canale in IBM WebSphere MQ .NET

Se si utilizzano i collegamenti client, è possibile utilizzare le uscite canale come per qualsiasi altra connessione client. Se si utilizzano i collegamenti gestiti, è necessario scrivere un programma di uscita che implementi un'interfaccia appropriata.

Bind client

Se si utilizzano i collegamenti client, è possibile utilizzare le uscite canale come descritto in Uscite canale. Non è possibile utilizzare le uscite del canale scritte per i bind gestiti.

Bind gestiti

Se si utilizza una connessione gestita, per implementare un'uscita, definire una nuova classe .NET che implementi l'interfaccia appropriata. Nel package WebSphere MQ sono definite tre interfacce di uscita:

- MQSendExit
- MQReceiveExit
- MQSecurityExit

Nota: Le uscite utente scritte utilizzando queste interfacce non sono supportate come uscite canale nell'ambiente non gestito.

Il seguente esempio definisce una classe che implementa tutti e tre:

```
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
  }

}
```

Ad ogni uscita viene passata una MQChannelExit e un'istanza oggetto MQChannelDefinition . Questi oggetti rappresentano le strutture MQCXP e MQCD definite nell'interfaccia procedurale.

I dati che devono essere inviati da un'uscita di invio e i dati ricevuti in un'uscita di sicurezza o di ricezione vengono specificati utilizzando i parametri dell'uscita.

All'immissione, i dati all'offset *dataOffset* con lunghezza *dataLength* nell'array di byte *dataBuffer* sono i dati che verranno inviati da un'uscita di invio e i dati ricevuti in un'uscita di sicurezza o di ricezione. Il parametro *dataMaxLength* fornisce la lunghezza massima (da *dataOffset*) disponibile per l'uscita in *dataBuffer*. Nota: per un'uscita di sicurezza, è possibile che dataBuffer sia null, se questa è la prima volta che l'uscita viene richiamata o se il partner ha deciso di non inviare dati.

Al ritorno, il valore di *dataOffset* e *dataLength* deve essere impostato in modo da puntare all'offset e alla lunghezza all'interno della matrice di byte restituita che le classi .NET devono utilizzare. Per un'uscita di invio, indica i dati che devono essere inviati e, per un'uscita di sicurezza o di ricezione, i dati che devono essere interpretati. L'uscita dovrebbe normalmente restituire una schiera di byte; le eccezioni sono un'uscita di sicurezza che potrebbe scegliere di non inviare dati e qualsiasi uscita richiamata con i motivi INIT o TERM. La forma più semplice di uscita che può essere scritta è quella che non fa altro che restituire dataBuffer:

Il corpo di uscita più semplice possibile è:

```
{
  return dataBuffer;
}
```

Classe MQChannelDefinition

V 7.5.0.6 Da Version 7.5.0, Fix Pack 6, l'ID utente e la password specificati con l'applicazione client .NET gestita sono impostati nella classe IBM WebSphere MQ .NET MQChannelDefinition passata all'uscita di sicurezza client. L'uscita di sicurezza copia l'id utente e password in MQCD MQCD.RemoteUserIdentifier e MQCD.RemotePassword RemotePassword (consultare “Scrittura di un'uscita di sicurezza” a pagina 408).

Specifica delle uscite canale (client gestito)

Se si specifica un nome canale e un nome connessione quando si crea l'oggetto MQQueueManager (nell'MQEnvironment o nel costruttore MQQueueManager), è possibile specificare le uscite del canale in due modi.

In ordine di precedenza, sono:

1. Passaggio delle proprietà della hashtable MQC.SECURITY_EXIT_PROPERTY, MQC.SEND_EXIT_PROPERTY o MQC.RECEIVE_EXIT_PROPERTY sul costruttore MQQueueManager .
2. Impostazione delle proprietà MQEnvironment SecurityExit, SendExit o ReceiveExit .

Se non si specifica un nome canale e un nome connessione, le uscite del canale da utilizzare provengono dalla definizione del canale prelevata da una CCDT (client channel definition table). Non è possibile sovrascrivere i valori memorizzati nella definizione del canale. Consultare Tabella di definizione del canale client e “Utilizzo di una tabella di definizione di canale client con .NET” a pagina 588 per ulteriori informazioni sulle tabelle di definizione del canale.

In ogni caso, la specifica assume la forma di una stringa con il seguente formato:

```
Assembly_name(Class_name)
```

Class_name è un nome completo, inclusa una specifica dello spazio dei nomi, di una classe .NET che implementa IBM.WMQ.MQSecurityExit, IBM.WMQ.MQSendExit o IBM.WMQ.MQReceiveExit (come appropriato). *Assembly_name* è l'ubicazione completa, inclusa l'estensione file, dell'assieme che ospita la classe. La lunghezza della stringa è limitata a 999 caratteri se si utilizzano le proprietà di MQEnvironment o MQQueueManager. Tuttavia, se il nome dell'uscita del canale è specificato in CCDT, è limitato a 128 caratteri. Quando necessario, il codice client .NET carica e crea un'istanza della classe specificata analizzando la specifica della stringa.

Specifica dei dati utente dell'uscita canale (client gestito)

Le uscite del canale possono avere dati utente associati. Se si specifica un nome canale e un nome connessione quando si crea l'oggetto MQQueueManager (nell'MQEnvironment o nel costruttore MQQueueManager), è possibile specificare i dati utente in due modi.

In ordine di precedenza, sono:

1. Passaggio delle proprietà della tabella hash MQC.SECURITY_USERDATA_PROPERTY, MQC.SEND_USERDATA_PROPERTY o MQC.RECEIVE_USERDATA_PROPERTY sul costruttore MQQueueManager .
2. Impostazione delle proprietà MQEnvironment SecurityUserData, SendUserData o ReceiveUserData.

Se non si specifica un nome canale e un nome connessione, i valori dei dati utente di uscita da utilizzare provengono dalla definizione di canale selezionata dalla CCDT (client channel definition table). Non è possibile sovrascrivere i valori memorizzati nella definizione del canale. Consultare Tabella di definizione del canale client e “Utilizzo di una tabella di definizione di canale client con .NET” a pagina 588 per ulteriori informazioni sulle tabelle di definizione del canale.

In ogni caso, la specifica è una stringa, limitata a 32 caratteri.

Riconnessione client automatica in .NET

È possibile riconnettere automaticamente il proprio client a un gestore code durante un'interruzione di connessione non prevista.

Un client può essere inaspettatamente disconnesso da un gestore code se, ad esempio, il gestore code si arresta o se si verifica un malfunzionamento della rete o del server.

Senza la riconnessione automatica del client, viene generato un errore quando la connessione non riesce. È possibile utilizzare il codice di errore per ristabilire la connessione.

Un client che utilizza la funzione di riconnessione client automatica viene definito un client ricollegabile. Per creare un client ricollegabile, specificare alcune opzioni denominate opzioni di riconnessione durante la connessione al gestore code.

Se l'applicazione client è un client WebSphere MQ .NET, può scegliere di ottenere la riconnessione automatica del client specificando un valore appropriato per CONNECT_OPTIONS_PROPERTY quando si utilizza la classe `MQQueueManager` per creare un gestore code. Consultare Opzioni di riconnessione per dettagli sui valori di CONNECT_OPTIONS_PROPERTY.

È possibile selezionare se l'applicazione client si connette e si riconnette sempre a un gestore code con lo stesso nome, allo stesso gestore code o a qualsiasi serie di gestori code definiti con lo stesso `QMNAME` nella tabella di connessione client (per i dettagli, consultare Gruppi di gestori code in CCDT).

Supporto SSL (Secure Sockets Layer)

La seguente sezione non si applica al client gestito.

WebSphere Le classi di MQ per applicazioni client .NET supportano la codifica SSL (Secure Sockets Layer). SSL fornisce la codifica di comunicazione, l'autenticazione e l'integrità del messaggio. Generalmente viene utilizzato per proteggere le comunicazioni tra due peer su Internet o all'interno di una intranet.

Abilitazione di SSL

SSL è supportato solo per le connessioni client. Per abilitare SSL, è necessario specificare la CipherSpec da utilizzare durante la comunicazione con il gestore code e deve corrispondere alla CipherSpec impostata sul canale di destinazione.

Per abilitare SSL, specificare CipherSpec utilizzando la variabile del membro statico SSLCipherSpec di MQEnvironment. Il seguente esempio si collega a un canale SVRCONN denominato SECURE.SVRCONN.CHANNEL, che è stato impostato per richiedere SSL con un CipherSpec 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");
```

Consultare Specifica di CipherSpecs per un elenco di CipherSpecs.

La proprietà SSLCipherSpec può essere impostata anche utilizzando MQC.SSL_CIPHER_SPEC_PROPERTY nella tabella hash delle proprietà di connessione.

Per connettersi correttamente utilizzando SSL, il keystore client deve essere configurato con la catena di certificati root dell'autorità di certificazione da cui è possibile autenticare il certificato presentato dal gestore code. Allo stesso modo, se SSLClientAuth sul canale SVRCONN è stato impostato su MQSSL_CLIENT_AUTH_REQUIRED, il keystore del client deve contenere un certificato personale di identificazione ritenuto attendibile dal gestore code.

Utilizzo del DN (Distinguished Name) del gestore code

Il gestore code si identifica utilizzando un certificato SSL, che contiene un *DN (Distinguished Name)* .

Un'applicazione client WebSphere MQ .NET può utilizzare questo DN per assicurarsi che stia comunicando con il gestore code corretto. Un modello DN viene specificato utilizzando la variabile nome sslPeerdi MQEnvironment. Ad esempio, l'impostazione:

```
MQEnvironment.SSLPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

consente la riuscita della connessione solo se il gestore code presenta un certificato con un nome comune che inizia con QMGR., e almeno due nomi di unità organizzative, il cui primo deve essere IBM e il secondo WEBSPHERE.

La proprietà SSLPeerName può essere impostata anche utilizzando la MQC.SSL_PEER_NAME_PROPERTY nella tabella hash delle proprietà di connessione. Per ulteriori informazioni sui DN (Distinguished Name) e le regole per l'impostazione dei nomi peer, consultare Sicurezza.

Se è impostato SSLPeerName , le connessioni hanno esito positivo solo se sono impostate su un modello valido e il gestore code presenta un certificato corrispondente.

Gestione degli errori quando si utilizza SSL

I seguenti codici di errore possono essere emessi dalle classi WebSphere MQ per .NET durante la connessione a un gestore code mediante SSL:

MQRC_SSL_NOT_ALLOWED
La proprietà SSLCipherSpec è stata impostata, ma è stata utilizzata la connessione dei bind. Solo la connessione client supporta SSL.

MQRC_SSL_PEER_NAME_MISMATCH
Il modello di DN specificato nella proprietà SSLPeerName non corrisponde al DN presentato dal gestore code.

ERRORE MQRC_SSL_PEER_NAME_ERROR
Il modello DN specificato nella proprietà SSLPeerName non era valido.

Utilizzo di .NET Monitor

Per informazioni importanti, consultare Funzioni che possono essere utilizzate solo con l'installazione primaria su Windows .

.NET Monitor è un'applicazione simile a un controllo dei trigger WebSphere MQ . È possibile creare componenti .NET che vengono istanziati ogni volta che un messaggio viene ricevuto su una coda monitorata e che elaborano tale messaggio. .NET Monitor viene avviato dal comando `runmqdnm` e arrestato dal comando `endmqdnm` . Per dettagli su questi comandi, consultare runmqdnm e endmqdnm.

Per utilizzare .NET Monitor, scrivere un componente che implementi l'interfaccia IMQObjectTrigger , definita in amqmdnm.dll.

I componenti possono essere transazionali o non transazionali. Un componente transazionale deve ereditare da System.EnterpriseServices.ServicedComponent ed essere registrato come RequiresTransaction o SupportsTransaction. Non deve essere registrato come RequiresNew poiché .NET Monitor ha già avviato una transazione.

Il componente riceve oggetti MQQueueManager, MQQueue e MQMessage da `runmqdnm`. Può anche ricevere una stringa del parametro utente, se ne è stata specificata una, utilizzando l'opzione della riga comandi *-u* , quando è stato avviato `runmqdnm` . Tenere presente che il componente riceve il contenuto di un messaggio arrivato sulla coda monitorata in un oggetto MQMessage. Non deve connettersi al gestore code, aprire la coda o richiamare il messaggio stesso. Il componente deve quindi elaborare il messaggio come appropriato e restituire il controllo a .NET Monitor.

Se il componente è stato scritto come componente transazionale, viene registrato per eseguire il commit o il rollback della transazione utilizzando le funzioni fornite da System.EnterpriseServices.ServicedComponent.

Poiché il componente riceve gli oggetti MQQueueManager e MQQueue e il messaggio, dispone di informazioni di contesto complete per quel messaggio e può, ad esempio, aprire un'altra coda sullo stesso gestore code senza dover connettersi separatamente a WebSphere MQ.

Frammenti di codice di esempio

Questo argomento contiene due esempi di componenti che ottengono un messaggio da .NET Monitor e lo stampano, uno utilizzando l'elaborazione transazionale e l'altro non transazionale. Un terzo esempio mostra le routine di utilità comuni, applicabili a entrambi i primi due esempi. Tutti gli esempi sono in C#.

Esempio 1: elaborazione transazionale

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

Esempio 2: elaborazione non transazionale

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

Esempio 3: routine comuni

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

    }
}
```

Compilazione dei programmi WebSphere MQ .NET

Comandi di esempio per compilare applicazioni .NET scritte in vari linguaggi.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Per creare un'applicazione C# utilizzando le classi WebSphere MQ per .NET, utilizzare il comando seguente:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib:MQ_INSTALLATION_PATH\bin /out:MyProg.exe MyProg.cs
```

Per creare un'applicazione Visual Basic utilizzando le classi WebSphere MQ per .NET, utilizzare il seguente comando:

```
vbc /r:System.dll /r:MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:MyProg.exe MyProg.vb
```

Per creare un'applicazione C++ gestita utilizzando WebSphere MQ classes for .NET, utilizza il seguente comando:

```
cl /clr MQ_INSTALLATION_PATH\bin Myprog.cpp
```

Per altre lingue, consultare la documentazione fornita dal fornitore della lingua.

Traccia dei programmi WebSphere MQ .NET

In WebSphere MQ .NET, si avvia e si controlla la funzionalità di traccia come nei programmi WebSphere MQ utilizzando MQI.

Tuttavia, i parametri -i e -p del comando strmqtrc, che consentono di specificare gli identificatori del processo e del sottoprocesso e i processi denominati, non hanno alcun effetto.

Normalmente è necessario utilizzare la funzione di traccia solo su richiesta del servizio IBM .

Per informazioni sui comandi di traccia, vedere Utilizzo della traccia su Windows .

Canale personalizzato IBM WebSphere MQ per Microsoft Windows Communication Foundation (WCF)

Il canale personalizzato Microsoft Windows Communication Foundation (WCF) per IBM WebSphere MQ invia e riceve messaggi tra i servizi e i client WCF.

Concetti correlati

“Introduzione all'utilizzo del canale personalizzato WebSphere MQ per WCF con .NET 3” a pagina 597
Panoramica delle informazioni disponibili per i programmatori utilizzando il canale personalizzato WebSphere MQ per Windows Communication Foundation (WCF) con .NET 3.

“Utilizzo dei canali personalizzati WebSphere MQ per WCF” a pagina 601
Panoramica delle informazioni disponibili per i programmatori che utilizzano i canali personalizzati WebSphere MQ V7 per Windows Communication Foundation (WCF).

“Utilizzo degli esempi WCF” a pagina 618
Gli esempi WCF (Windows Communication Foundation) forniscono alcuni semplici esempi di come può essere utilizzato il canale personalizzato WebSphere MQ .

“Determinazione dei problemi sul canale personalizzato WCF per WebSphere MQ” a pagina 624
È possibile utilizzare la traccia WebSphere MQ per raccogliere informazioni dettagliate sulle varie parti del codice WebSphere MQ . Quando si utilizza Windows Communication Foundation (WCF), viene generato un output di traccia separato per la traccia del canale personalizzato WCF integrata con la traccia dell'infrastruttura WCF Microsoft .

Introduzione all'utilizzo del canale personalizzato WebSphere MQ per WCF con .NET 3

Panoramica delle informazioni disponibili per i programmatori utilizzando il canale personalizzato WebSphere MQ per Windows Communication Foundation (WCF) con .NET 3.

Cos' è il canale personalizzato WebSphere MQ per WCF?

Il canale personalizzato per WebSphere MQ è un canale di trasporto che utilizza il modello di programmazione unificato Microsoft Windows Communication Foundation (WCF).

Il framework Microsoft Windows Communication Foundation, introdotto in Microsoft .NET 3, consente lo sviluppo di applicazioni e servizi .NET indipendentemente dal trasporto e dai protocolli utilizzati per collegarli, consentendo l'utilizzo di configurazioni o trasporti alternativi in base all'ambiente in cui viene distribuito il servizio o l'applicazione.

Le connessioni vengono gestite in fase di runtime da WCF creando uno stack del canale contenente la combinazione richiesta di:

- Elementi protocollo: una serie facoltativa di elementi in cui nessuno, uno o più possono essere aggiunti per supportare protocolli come gli standard WS - *.
- Codificatore del messaggio: un elemento obbligatorio nello stack che controlla la serializzazione del messaggio nel suo formato wire.
- Canale di trasporto: un elemento obbligatorio nello stack responsabile del trasferimento del messaggio serializzato al relativo endpoint.

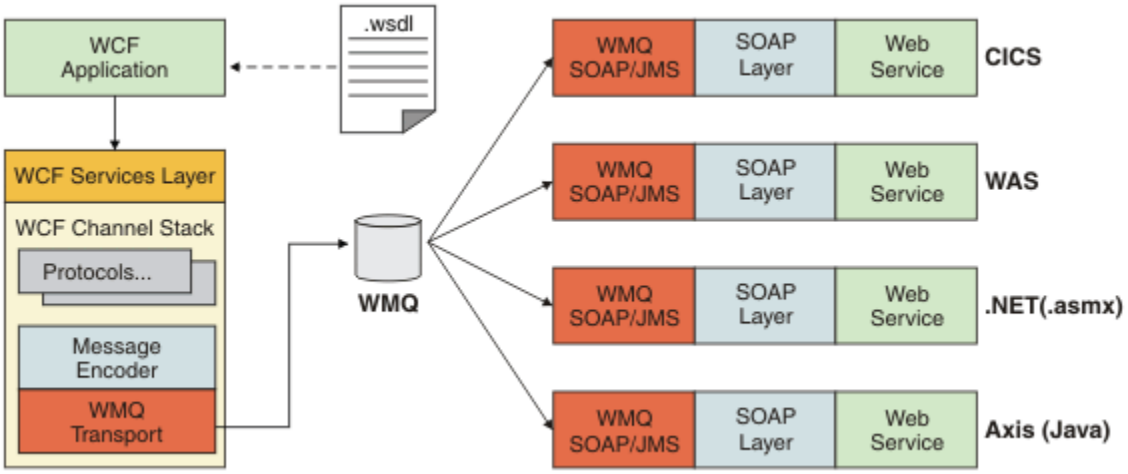
Il canale personalizzato per WebSphere MQ è un canale di trasporto e, come tale, deve essere accoppiato con un codificatore di messaggi e protocolli facoltativi come richiesto dall'applicazione che utilizza un bind personalizzato WCF. In questo modo, le applicazioni che sono state sviluppate per utilizzare WCF possono utilizzare il canale personalizzato per WebSphere MQ per inviare e ricevere i dati nello stesso modo in cui utilizzano i trasporti integrati forniti da Microsoft, consentendo una semplice integrazione con le funzioni di messaggistica asincrone, scalabili e affidabili di WebSphere MQ. Per un elenco completo delle funzioni supportate, consultare “Funzionalità e caratteristiche del canale WCF Custom” a pagina 601.

Quando e perché utilizzare il canale personalizzato WebSphere MQ per WCF?

Il canale personalizzato WebSphere MQ può essere utilizzato per inviare e ricevere messaggi tra i client WCF e i servizi allo stesso modo dei trasporti integrati forniti da Microsoft, consentendo alle applicazioni di accedere alle funzioni di WebSphere MQ all'interno del modello di programmazione unificato WCF.

Uno scenario di modello di utilizzo tipico del canale personalizzato WebSphere MQ per WCF è come interfaccia per i servizi Web ospitati su WebSphere MQ (SOAP/JMS)

I messaggi vengono trasmessi utilizzando il formato di messaggio SOAP su JMS di WebSphere MQ, consentendo ai client e ai servizi WCF di richiamare o essere richiamati anche da altre applicazioni WebSphere MQ o da ambienti host compatibili con questo formato, inclusi i servizi Web e i client in esecuzione in WebSphere Application Server, CICS, Axis v1 (Java), e .asmx (.NET), come mostrato nel seguente diagramma:



Per i dettagli su SOAP su JMS, consultare: “Trasporto WebSphere MQ per SOAP” a pagina 949

Un esempio di scenario tipico dal diagramma è:

1. Un servizio Web ospitato in WebSphere Application Server ed esposto su WebSphere MQ utilizzando il supporto per SOAP su JMS all'interno di WebSphere Application Server
2. Il documento WSDL che descrive il servizio può quindi essere utilizzato dallo strumento WCF per generare un proxy client e una configurazione che creerebbe uno stack di canali WCF appropriato, incluso il canale personalizzato.
3. L'applicazione client può quindi utilizzare il proxy per avviare il servizio Web nello stesso modo di qualsiasi altro servizio Web.

Il canale viene generalmente utilizzato con un codificatore di messaggi WCF text/SOAP, ma il canale può essere accoppiato con altri codificatori di messaggi WCF, se necessario. L'utilizzo di codificatori alternativi può anche fornire un'integrazione limitata con le applicazioni WebSphere MQ native che non supportano SOAP su JMS, ma questo non è il ruolo principale del canale.

I vantaggi principali dell'utilizzo del canali personalizzato in un ambiente WCF sono:

- Richiamo asincrono: il supporto attiva e dimentica le operazioni client in cui il client è disaccoppiato dalla disponibilità del servizio e delle funzionalità, come il reinstradamento delle risposte e il multi - hop.
- Caratteristiche di scalabilità affidabili: la messaggistica basata sulla coda consente di aggiungere in modo prevedibile la capacità a un sistema.
- Qualità del servizio: i messaggi sono tangibili e tracciabili e possono essere facilmente gestiti e amministrati.

Requisiti software e istruzioni di installazione per WebSphere MQ canale personalizzato per WCF

Questo argomento descrive i requisiti software e le informazioni di installazione per il canale personalizzato WebSphere MQ per WCF.

Il canale personalizzato WebSphere MQ per WCF può connettersi solo a WebSphere MQ V7 o a gestori code superiori.

Requisiti software per il canale personalizzato WCF per WebSphere MQ

Queste informazioni elencano i requisiti software per il canale personalizzato WCF per WebSphere MQ.

Ambiente di runtime

- Microsoft .NET Framework v3.0 o versione successiva deve essere installato sulla macchina host.
- *Java e .NET Messaging and Web Services* viene installato per impostazione predefinita come parte del programma di installazione WebSphere MQ 7.0.1 . Installa gli assembly .NET necessari per il canale personalizzato nella Global Assembly Cache.

Nota: Se Microsoft .NET Framework v2.0 o versione successiva non è installato prima di installare WebSphere MQ V7.0.1, l'installazione del prodotto WebSphere MQ continua senza errori, ma il canale personalizzato WebSphere MQ non è disponibile. Se .NET Framework viene installato dopo l'installazione di WebSphere MQ 7.0.1, allora il canale personalizzato WebSphere MQ deve essere attivato eseguendo lo script *WMQInstallDir*\bin\amqiRegisterdotNet.cmd , dove *WMQInstallDir* è la directory in cui è installato WebSphere MQ 7.0.1 . Questo script installa gli assembly richiesti nella GAC (Global Assembly Cache). Una serie di file amqi*.log che registrano le azioni eseguite viene creata nella directory %TEMP% . Non è necessario eseguire nuovamente lo script amqiRegisterdotNet.cmd se .NET è aggiornato a v3.0 o versione successiva da una versione precedente, ad esempio da .NET v2.0.

Ambiente di sviluppo

- Microsoft Visual Studio 2008 o Windows Software Development Kit per .NET 3.0 o versioni successive.
- Microsoft .NET Framework V3.5 o superiore deve essere installato sulla macchina host per creare i file della soluzione di esempio.

Nota: Se Microsoft .NET Framework v2.0 o versione successiva non è installato prima di installare WebSphere MQ V7.0.1, l'installazione del prodotto WebSphere MQ continua senza errori, ma il canale personalizzato WebSphere MQ non è disponibile. Se .NET Framework viene installato dopo l'installazione di WebSphere MQ 7.0.1, allora il canale personalizzato WebSphere MQ deve essere attivato eseguendo lo script *WMQInstallDir*\bin\amqiRegisterdotNet.cmd , dove *WMQInstallDir* è la directory in cui è installato WebSphere MQ 7.0.1 . Questo script installa gli assembly richiesti nella GAC (Global Assembly Cache). Una serie di file amqi*.log che registrano le azioni eseguite viene creata nella

directory %TEMP% . Non è necessario eseguire nuovamente lo script `amqiRegisterdotNet.cmd` se .NET è aggiornato a v3.0 o versione successiva da una versione precedente, ad esempio da .NET v2.0.

Canale personalizzato WebSphere MQ per WCF: cosa è installato?

Il canale personalizzato per WebSphere MQ è un canale di trasporto che utilizza il modello di programmazione unificato WCF (Microsoft Windows Communication Foundation). Il canale personalizzato viene installato per impostazione predefinita come parte dell'installazione di WebSphere MQ 7.0.1 .

Canale personalizzato WebSphere MQ per WCF

Il canale personalizzato di WebSphere MQ per WCF è installato per impostazione predefinita come parte dell'installazione di WebSphere MQ 7.0.1 ; il canale personalizzato e le relative dipendenze sono contenuti all'interno del componente `Java and .NET Messaging and Web Services` , che è installato per impostazione predefinita. Durante l'aggiornamento a WebSphere MQ 7.0.1 da una versione precedente, l'aggiornamento installerà il canale personalizzato WebSphere MQ per WCF per impostazione predefinita se il componente `Java and .NET Messaging and Web Services` è stato precedentemente installato in un'installazione precedente.

Il componente `Java and .NET Messaging and Web Services` contiene il file `IBM.XMS.WCF.dll` e il file `IBM.XMS.WCF.dll` è l'assemblaggio del canale personalizzato principale, che contiene le classi di interfaccia WCF. Questo file è installato nella cache GAC (Global Assembly Cache) ed è disponibile anche nella seguente directory: *MQ_INSTALLATION_PATH*`\bin` dove *MQ_INSTALLATION_PATH* è la directory in cui è installato WebSphere MQ 7.0.1 .

Le classi chiave richieste per l'utilizzo del canale personalizzato si trovano nello spazio dei nomi *: IBM.XMS.WCF* e:

Nome bind trasporto	IBM.XMS.WCF.SoapJmsIbmTransportBindingElement
Programma di importazione binding di trasporto	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementImporter
Configurazione binding di trasporto	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementConfig

Esempi di canale personalizzato WebSphere MQ

Gli esempi forniscono alcuni semplici esempi di come è possibile utilizzare il canale personalizzato WebSphere MQ per WCF. Gli esempi e i relativi file associati si trovano nella directory *MQ_INSTALLATION_PATH*`\tools\wcf\samples\` , dove *MQ_INSTALLATION_PATH* è la directory di installazione per WebSphere MQ. Per ulteriori informazioni sugli esempi di canali personalizzati di WebSphere MQ , consultare: “Utilizzo degli esempi WCF” a pagina 618

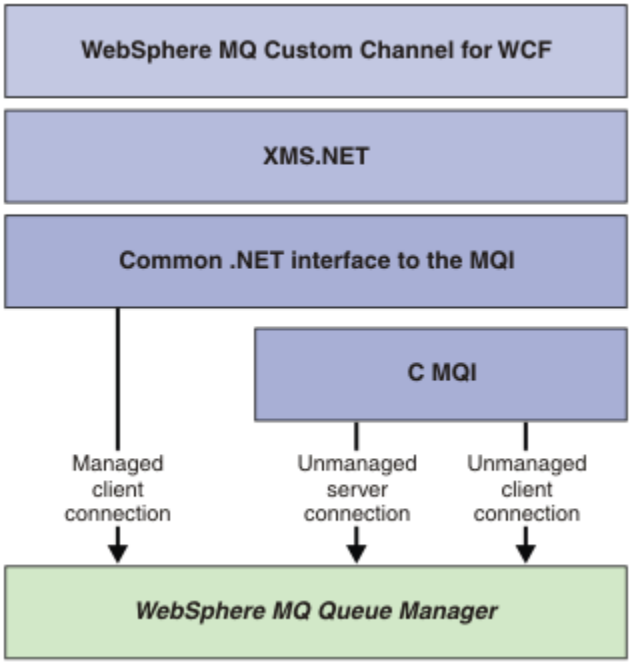
svcutil.exe.config

Il `svcutil.exe.config` è un esempio delle impostazioni di configurazione richieste per consentire allo strumento di generazione proxy del client Microsoft WCF `svcutil` di riconoscere il canale personalizzato. Il file `svcutil.exe.config` si trova nella directory *MQ_INSTALLATION_PATH*`\tools\wcf\docs\examples\` , dove *MQ_INSTALLATION_PATH* è la directory di installazione per WebSphere MQ. Per ulteriori informazioni sull'utilizzo di `svcutil.exe.config`, consultare: “Generazione di un proxy client WCF e di file di configurazione dell'applicazione utilizzando lo strumento svcutil con metadati da un servizio in esecuzione” a pagina 615.

Architettura WCF

Il canale personalizzato WebSphere MQ per WCF è integrato nell'API `IBM Message Service Client for .NET (XMS .NET)` .

L'architettura WCF è quella mostrata nel diagramma seguente:



Tutti i componenti richiesti vengono installati per impostazione predefinita con l'installazione di WebSphere MQ V7.0.1 .

Le tre connessioni sono: connessioni client gestite, connessioni server non gestite e connessioni client non gestite. Per ulteriori informazioni su queste connessioni, consultare “Opzioni di connessione WCF” a pagina 605.

Utilizzo dei canali personalizzati WebSphere MQ per WCF

Panoramica delle informazioni disponibili per i programmatori che utilizzano i canali personalizzati WebSphere MQ V7 per Windows Communication Foundation (WCF).

La Microsoft Windows Communication Foundation supporta i servizi Web e il supporto della messaggistica in Microsoft .NET Framework 3. WebSphere MQ V7 può ora essere utilizzato come canale personalizzato all'interno di WCF in .NET Framework 3 allo stesso modo dei canali integrati offerti da Microsoft.

I messaggi trasportati attraverso il canale personalizzato vengono formattati in base all'implementazione SOAP su JMS di WebSphere MQ V7. Le applicazioni possono quindi comunicare con i servizi ospitati da WCF o dall'infrastruttura del servizio WebSphere SOAP over JMS. Per i dettagli su SOAP su JMS, consultare: “Trasporto WebSphere MQ per SOAP” a pagina 949

Funzionalità e caratteristiche del canale WCF Custom

Utilizzare i seguenti argomenti per informazioni relative alle funzioni e alle funzioni del canale personalizzato WCF.

Forme canale personalizzate WCF

Panoramica delle forme canale personalizzate che WebSphere MQ può essere utilizzato all'interno dei canali personalizzati Microsoft Windows Communication Foundation (WCF).

Il canale personalizzato WebSphere MQ per WCF supporta due forme di canale:

- Unidirezionale
- Richiesta-Risposta

WCF seleziona automaticamente la forma del canale in base al contratto di servizio che si sta ospitando.

I contratti che includono metodi che utilizzano solo il parametro **IsOneWay** vengono serviti dalla forma del canale unidirezionale, ad esempio:

```
[OperationContract(IsOneWay = true)]
void printString(String text);
```

I contratti che includono una combinazione di metodi unidirezionali e di richiesta - risposta, o tutti i metodi di richiesta - risposta, vengono serviti dalla forma del canale richiesta - risposta. Ad esempio:

```
[OperationContract]
int subtract(int a, int b);

[OperationContract(IsOneWay = true)]
void printString(string text);
```

Nota: Quando si combinano metodi unidirezionali e di richiesta - risposta nello stesso contratto, è necessario assicurarsi che il comportamento sia quello previsto, in particolare quando si lavora in un ambiente misto perché i metodi unidirezionali attendono che ricevano una risposta null dal servizio.

Canale unidirezionale

Il canale personalizzato a una via di WebSphere MQ per WCF viene utilizzato, ad esempio, per inviare messaggi da un client WCF utilizzando una forma di canale a una via. Il canale può inviare messaggi solo in una direzione, ad esempio, da un gestore code client a un servizio WCF.

Canale di richiesta - risposta

Il canale personalizzato di richiesta - risposta WebSphere MQ per WCF viene utilizzato, ad esempio, per inviare i messaggi in due direzioni in modo asincrono; la stessa istanza client deve essere utilizzata per la messaggistica asincrona. Il canale può inviare messaggi in una direzione, ad esempio, da un gestore code client a una coda su un servizio WCF e quindi inviare un messaggio di risposta da WCF a una coda sul gestore code client.

Valori e nomi parametro URI WCF

connectionFactory
Il parametro connectionFactory è richiesto. Per la sintassi di questo parametro, consultare Sintassi URI e parametri per la distribuzione del servizio Web .

initialContextFactory
Il parametro initialContextFactory è obbligatorio e deve essere impostato su "com.ibm.mq.jms.Nojndi" per la compatibilit ... con WebSphere Application Server e altri prodotti (consultare “Distribuzione di un servizio in WebSphere Application Server per utilizzare WebSphere Transport for SOAP” a pagina 1007).

Distribuzione garantita canale personalizzato WCF

La consegna garantita garantisce che una richiesta di servizio o risposta è azionata e non persa.

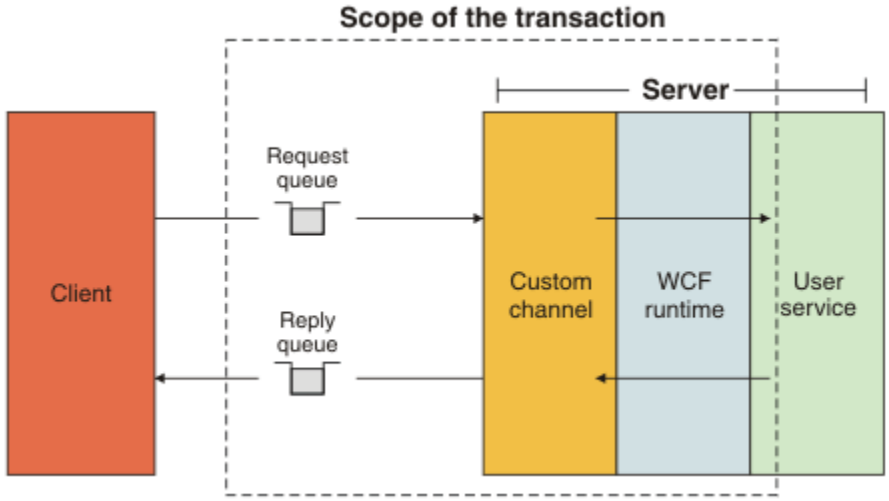
Viene ricevuto un messaggio di richiesta e qualsiasi messaggio di risposta viene inviato in un punto di sincronizzazione della transazione locale, di cui è possibile eseguire il rollback in caso di errore di runtime. Esempi di questi errori sono: un'eccezione non gestita generata dal servizio, un errore di invio del messaggio al servizio o un errore di consegna del messaggio di risposta.

`AssuredDelivery` è l'attributo di consegna garantita che può essere specificato su un contratto di servizio per garantire che tutti i messaggi di richiesta ricevuti da un servizio e qualsiasi messaggio di risposta inviato da un servizio, non vadano persi in caso di errore di runtime.

Per garantire che i messaggi vengano conservati anche in caso di errore di sistema o di interruzione dell'alimentazione, i messaggi devono essere inviati come persistenti. Per utilizzare i messaggi persistenti, l'applicazione client deve avere questa opzione specificata sull'URI dell'endpoint. Per

ulteriori informazioni sull'impostazione delle proprietà URI, consultare: Parametri e sintassi URI per la distribuzione del servizio Web.

Le transazioni distribuite non sono supportate e l'ambito della transazione non si estende oltre l'elaborazione del messaggio di richiesta e risposta eseguita da WebSphere MQ. Qualsiasi lavoro eseguito all'interno del servizio potrebbe essere rieseguito come risultato di un errore che causa la ricezione del messaggio. Il seguente diagramma mostra l'ambito della transazione:



La consegna garantita viene abilitata applicando l'attributo `AssuredDelivery` alla classe di servizio come mostrato nel seguente esempio:

```
[AssuredDelivery]
class TestCalculatorService : IWMQSampleCalculatorContract
{
    public int add(int a, int b)
    {
        int ans = a + b;
        return ans;
    }
}
```

Quando si utilizza l'attributo `AssuredDelivery` , è necessario essere consapevoli dei punti seguenti:

- Quando un canale determina che è probabile che un errore si ripeta se un messaggio è stato sottoposto a rollback e ricevuto di nuovo, il messaggio viene trattato come un messaggio non elaborabile e non viene restituito alla coda di richieste per la rielaborazione. Ad esempio: se il messaggio ricevuto non è formattato correttamente o non può essere inviato ad un servizio. Le eccezioni non gestite generate da un'operazione di servizio vengono sempre reinviate fino a quando il messaggio non viene riconsegnato il numero massimo di volte specificato dalla proprietà della soglia di backout della coda di richieste. Per ulteriori informazioni, consultare: “Messaggi non elaborabili del canale personalizzato WCF” a pagina 604
- Il canale esegue la lettura, l'elaborazione e la risposta di ogni messaggio di richiesta come un'operazione atomica utilizzando un singolo thread di esecuzione per applicare l'integrità transazionale. Per abilitare l'esecuzione simultanea delle operazioni di servizio, il canale consente a WCF di creare più istanze del canale. Il numero di istanze del canale disponibili per l'elaborazione delle richieste è controllato dalla proprietà di bind `MaxConcurrentCalls`. Per ulteriori informazioni, consultare “Opzioni di configurazione del bind WCF” a pagina 611
- La funzione di distribuzione garantita utilizza sia i punti di estendibilità IOperationInvoker che IErrorHandler WCF. Se questi punti di estendibilità vengono utilizzati esternamente da un'applicazione, l'applicazione deve garantire che vengano richiamati tutti i punti di estendibilità precedentemente registrati. La mancata esecuzione di questa operazione per IErrorHandler può causare errori non notificati. Se non si riesce a eseguire questa operazione per IOperationInvoker , WCF potrebbe non rispondere più.

Sicurezza del canale personalizzato WCF

Il canale personalizzato WebSphere MQ per WCF supporta l'utilizzo di SSL solo per connessioni client non gestite al gestore code.

SSL può essere specificato in due modi:

- Specificare SSL direttamente sul SOAP sull'URI JMS. Per una descrizione completa delle opzioni SSL, consultare SSL e il trasporto WebSphere MQ per SOAP
- Specificare SSL utilizzando una voce nella CCDT (client channel definition table). Per ulteriori informazioni su CCDT, consultare Tabella di definizione del canale client

Tabelle di definizione del canale client WCF (CCDT)

Il canale personalizzato di WebSphere MQ per WCF supporta l'utilizzo di CCDT (client channel definition tables) per configurare le informazioni di connessione per le connessioni client.

I CCDT sono controllati tramite queste due variabili di ambiente:

- *MQCHLLIB* specifica la directory in cui si trova la tabella.
- *MQCHLTAB* specifica il nome file della tabella.

Non Š possibile specificare la tabella di definizione del canale direttamente in SOAP sull'URI JMS. Se queste variabili di ambiente sono definite, hanno la priorità sui dettagli di connessione client specificati nell'URI.

Per ulteriori informazioni sulle tabelle di definizione del canale client, consultare: Tabella di definizione del canale client .

Concetti correlati

Tabella definizione canale client

Messaggi non elaborabili del canale personalizzato WCF

Quando un servizio non riesce a elaborare un messaggio di richiesta o a consegnare un messaggio di risposta a una coda di risposta, il messaggio viene trattato come un messaggio non elaborabile.

Messaggi di richiesta poison

Se un messaggio di richiesta non può essere elaborato, viene trattato come un messaggio non elaborabile. Questa azione impedisce al servizio di ricevere nuovamente lo stesso messaggio non elaborabile. Affinché un messaggio di richiesta non elaborabile venga trattato come un messaggio non elaborabile, una delle seguenti situazioni deve essere true:

- Il conteggio di backout dei messaggi ha superato la soglia di backout specificata sulla coda di richieste, che si verifica solo se è stata specificata la consegna garantita per il servizio. Per ulteriori informazioni sulla consegna sicura, consultare: “Distribuzione garantita canale personalizzato WCF” a pagina 602
- Il messaggio non è stato formattato correttamente e non può essere interpretato come un messaggio SOAP su JMS.

Messaggi di risposta non elaborabili

Se un servizio non riesce a consegnare un messaggio di risposta alla coda di risposta, il messaggio di risposta viene trattato come un messaggio non elaborabile. Per i messaggi di risposta, questa azione consente ai messaggi di risposta di essere richiamati in un secondo momento per facilitare la determinazione dei problemi.

Gestione messaggi non elaborabili

L'azione intrapresa per un messaggio non elaborabile dipende dalla configurazione del gestore code e dai valori impostati nelle opzioni di report del messaggio. Per SOAP su JMS, le seguenti opzioni di report sono impostate sui messaggi di richiesta per impostazione predefinita e non sono configurabili:

- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA

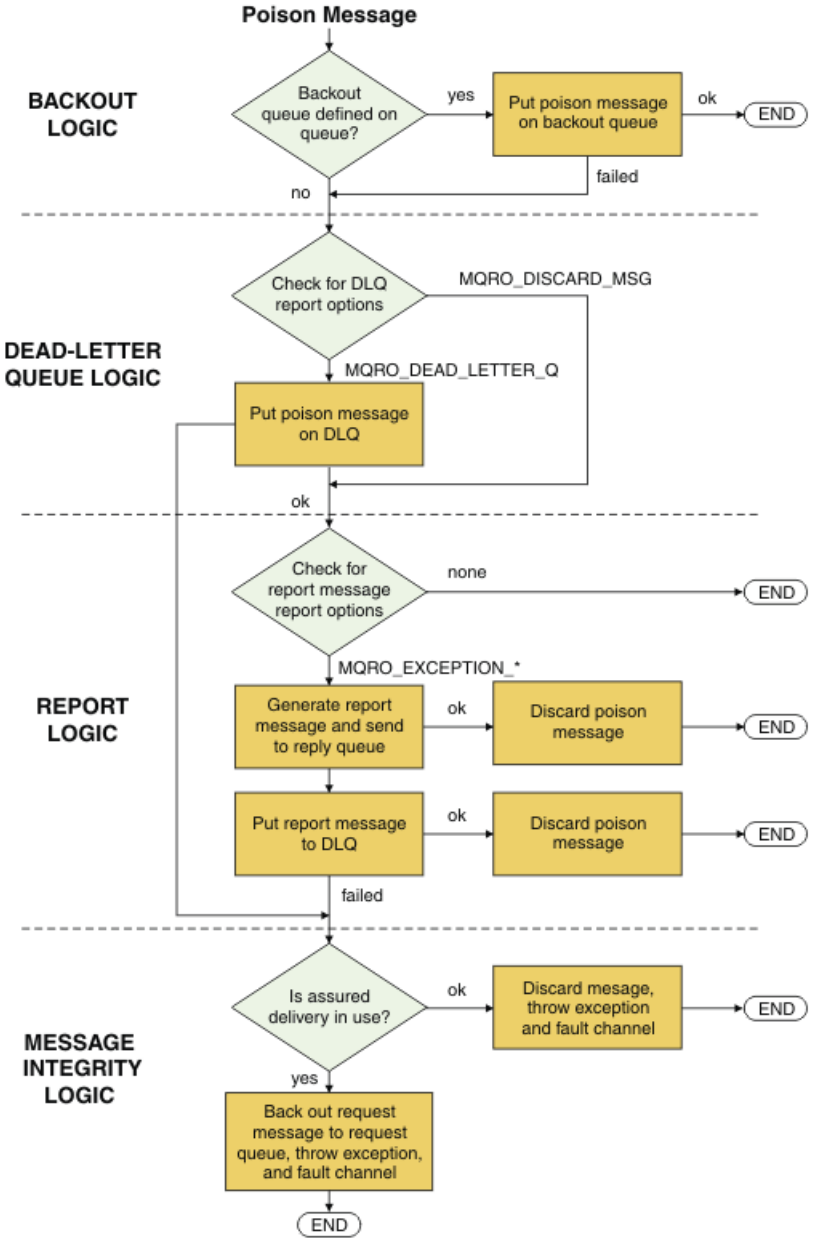
- MQRO_DISCARD_MSG

Per SOAP su JMS, la seguente opzione di report è impostata sui messaggi di risposta per impostazione predefinita e non può essere configurata:

- MQRO_DEAD_LETTER_Q

Se i messaggi provengono da un'origine non WCF, fare riferimento alla documentazione per tale origine.

Il seguente diagramma mostra le azioni possibili e le operazioni eseguite se la gestione di un messaggio non elaborabile ha esito negativo:



Opzioni di connessione WCF

Esistono tre modi per collegare un canale personalizzato WebSphere MQ per WCF a un gestore code. Considerare il tipo di connessione più adatto alle proprie esigenze.

Per ulteriori informazioni sulle opzioni di connessione, consultare: “Differenze di connessione” a pagina 578

Per ulteriori informazioni sull'architettura WCF, consultare: “Architettura WCF” a pagina 600

Connessione client non gestito

Una connessione effettuata in questa modalità si connette come client WebSphere MQ a un server WebSphere MQ in esecuzione sulla macchina locale o su una macchina remota.

Per utilizzare il canale personalizzato di WebSphere MQ per WCF come client WebSphere MQ , è possibile installarlo con il client WebSphere MQ MQI, sul server WebSphere MQ o su una macchina separata.

Connessione server non gestita

Quando viene utilizzato in modalità di bind del server, il canale personalizzato WebSphere MQ per WCF utilizza l'API del gestore code, piuttosto che comunicare attraverso una rete. L'utilizzo delle connessioni di bind fornisce prestazioni migliori per le applicazioni WebSphere MQ rispetto all'utilizzo delle connessioni di rete.

Per utilizzare la connessione dei collegamenti, è necessario installare il canale personalizzato WebSphere MQ per WCF sul server WebSphere MQ .

Connessione client gestito

Una connessione effettuata in questa modalità si connette come client WebSphere MQ a un server WebSphere MQ in esecuzione sulla macchina locale o su una macchina remota.

Le classi del canale personalizzato WebSphere MQ per .NET 3 che si collegano in questa modalità rimangono nel codice gestito .NET e non effettuano chiamate ai servizi nativi. Per ulteriori informazioni sul codice gestito, consultare la documentazione Microsoft .

Esistono diverse limitazioni all'utilizzo del client gestito. Per ulteriori informazioni su queste limitazioni, consultare “Connessioni client gestite” a pagina 578.

Creazione e configurazione del canale personalizzato WebSphere MQ per WCF

I canali personalizzati WebSphere MQ V7 per WCF funzionano allo stesso modo dei canali WCF di trasporto offerti da Microsoft. Il canale personalizzato di WebSphere MQ per WCF può essere creato in uno dei due modi.

Informazioni su questa attività

Il canale personalizzato WebSphere MQ si integra con WCF come un canale di trasporto WCF e, come tale, deve essere associato a un codificatore di messaggi e a canali di protocollo facoltativi, in modo da poter creare uno stack di canale completo che può essere utilizzato da una applicazione. Sono necessari due elementi per creare correttamente uno stack del canale completo:

1. Una definizione di bind: specifica quali elementi sono richiesti per creare lo stack del canale delle applicazioni, incluso il canale di trasporto, il codificatore dei messaggi e i protocolli, oltre a tutte le impostazioni di configurazione generali. Per il canale personalizzato, la definizione di bind deve essere creata sotto forma di un bind personalizzato WCF.
2. Una definizione di endpoint: collega il contratto di servizi con la definizione di bind e fornisce anche l'URI di connessione effettivo che descrive dove l'applicazione può connettersi. Per il canale personalizzato, l'URI è nel formato di un URI SOAP su JMS.

Queste definizioni possono essere create in due modi diversi:

- Amministrativamente; le definizioni vengono create fornendo i dettagli in un file di configurazione dell'applicazione (ad esempio: `app.config`).
- In modo programmatico; le definizioni vengono create direttamente dal codice dell'applicazione.

La decisione su quale metodo utilizzare per creare le definizioni deve essere basata sui requisiti della domanda come segue:

- Il metodo di gestione per la configurazione fornisce la flessibilità per modificare i dettagli del servizio e del client dopo la distribuzione senza ricreare l'applicazione.
- Il metodo programmatico per la configurazione fornisce una maggiore protezione dagli errori di configurazione e la possibilità di creare dinamicamente una configurazione in fase di runtime.

Creazione amministrativa di un canale personalizzato WCF fornendo informazioni sul bind e sull'endpoint in un file di configurazione dell'applicazione

Il canale personalizzato WebSphere MQ per WCF è un canale WCF a livello di trasporto. Un endpoint e un bind devono essere definiti per utilizzare il canale personalizzato e queste definizioni possono essere eseguite fornendo le informazioni sul bind e sull'endpoint in un file di configurazione dell'applicazione.

Per configurare e utilizzare un canale personalizzato WebSphere MQ per WCF, che è un canale WCF a livello di trasporto, è necessario definire un bind e una definizione endpoint. Il bind contiene le informazioni di configurazione per il canale e la definizione endpoint contiene i dettagli di connessione. Queste definizioni possono essere create in due modi:

- Programmaticamente direttamente dal codice dell'applicazione, come descritto di seguito: "Creazione di un canale personalizzato WCF mediante fornitura programmatica di informazioni di binding ed endpoint" a pagina 609
- Amministrativamente, fornendo i dettagli in un file di configurazione dell'applicazione, come descritto nella seguente procedura.

Il file di configurazione dell'applicazione del servizio o del client è comunemente denominato *yourappname*`.exe.config` dove *yourappname* è il nome della tua applicazione. Il file di configurazione dell'applicazione viene modificato più facilmente utilizzando lo strumento dell'editor di configurazione del servizio Microsoft denominato `SvcConfigEditor.exe` nel seguente modo:

- Avviare lo strumento dell'editor di configurazione `SvcConfigEditor.exe` . Il percorso di installazione predefinito per lo strumento è: *`Drive:`*`\Program Files\Microsoft SDKs\Windows\v6.0\Bin\SvcConfigEditor.exe` dove *Unità:* è il nome dell'unità di installazione.

Passo 1: aggiungi un'estensione dell'elemento di bind per abilitare WCF a individuare il canale personalizzato

1. Fare clic con il tasto destro del mouse su **Avanzate** > **Estensione** > **elemento di collegamento** per aprire il menu e selezionare **Nuovo**
2. Completare i campi come mostrato in questa tabella:

Tabella 73. Nuovi campi dell'elemento di collegamento

Campo	**Valore**
Nome	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`
Tipo	Passare a `IBM.XMS.WCF.dll` nella GAC (Global Assembly Cache) e selezionare `IBM.XMS.WCFSoapJmsIbmTransportBindingElementConfig`

Passo 2: crea una definizione di bind personalizzato che accoppia il canale personalizzato con un codificatore di messaggi WCF

1. Fare clic con il tasto destro del mouse su **Bind** per aprire il menu e selezionare **Nuova configurazione bind**
2. Completare i campi come mostrato in questa tabella:

Tabella 74. Nuovi campi di configurazione del collegamento	
Campo	**Valore**
Nome	`CustomBinding_WMQ`
BindingElement 1	`textMessageEncoding (MessageVersion: Soap11)`
BindingElement 2	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`

Passo 3: Specificare le proprietà di binding

1. Selezionare *IBM.XMS.WCF.SoapJmsIbmTransportChannel* dal collegamento creato in "Passo 2: crea una definizione di bind personalizzato che accoppia il canale personalizzato con un codificatore di messaggi WCF" a pagina 607
2. Apportare le modifiche richieste ai valori predefiniti delle proprietà come descritto in: "Opzioni di configurazione del bind WCF" a pagina 611

Passo 4: crea una definizione di endpoint

Crea una definizione di endpoint che fa riferimento al bind personalizzato che hai creato in "Passo 2: crea una definizione di bind personalizzato che accoppia il canale personalizzato con un codificatore di messaggi WCF" a pagina 607 e fornisce i dettagli di connessione del servizio. Il modo in cui queste informazioni vengono specificate dipende dal fatto che la definizione sia per un'applicazione client o per un'applicazione di servizio.

Per un'applicazione client, aggiungere una definizione di endpoint alla sezione client nel modo seguente:

1. Fare clic col pulsante destro del mouse su **Client** > **Endpoint** per aprire il menu e selezionare **Nuovo endpoint client**
2. Completare i campi come mostrato in questa tabella:

Tabella 75. Nuovi campi endpoint client	
Campo	**Valore**
Nome	`Endpoint_WMQ`
Indirizzo	*L'URI SOAP/JMS che descrive i dettagli di connessione WMQ richiesti per accedere al servizio.* Per ulteriori dettagli, consultare: "WebSphere MQ per formato indirizzo URI endpoint WCF" a pagina 610
Binding	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Contratto	*Il nome dell'interfaccia del contratto di servizio*

Per un'applicazione di servizio, aggiungere una definizione di servizio alla sezione dei servizi nel modo seguente:

1. Fare clic con il tasto destro del mouse su **Servizi** per aprire il menu e selezionare **Nuovo servizio**, quindi selezionare la classe di servizio da ospitare.
2. Aggiungere una definizione di endpoint alla sezione **Endpoint** per il nuovo servizio e completare i campi come mostrato in questa tabella:

Tabella 76. Nuovi campi endpoint servizio	
Campo	**Valore**
Nome	`Endpoint_WMQ`

Tabella 76. Nuovi campi endpoint servizio (Continua)	
Campo	**Valore**
Indirizzo	*L'URI SOAP/JMS che descrive i dettagli di connessione WMQ richiesti per accedere al servizio.* Per ulteriori dettagli, consultare: "WebSphere MQ per formato indirizzo URI endpoint WCF" a pagina 610
Binding	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Contratto	*Il nome della classe di implementazione del servizio*

Creazione di un canale personalizzato WCF mediante fornitura programmatica di informazioni di binding ed endpoint

Il canale personalizzato WebSphere MQ per WCF è un canale WCF a livello di trasporto. È necessario definire un endpoint e un bind per utilizzare il canale personalizzato e queste definizioni possono essere eseguite in modo programmatico direttamente dal codice dell'applicazione.

Per configurare e utilizzare un canale personalizzato WebSphere MQ per WCF, che è un canale WCF a livello di trasporto, è necessario definire un bind e una definizione endpoint. Il bind contiene le informazioni di configurazione per il canale e la definizione endpoint contiene i dettagli di connessione. Per ulteriori informazioni, consultare: "Utilizzo degli esempi WCF" a pagina 618

Queste definizioni possono essere create in due modi:

- In modo amministrativo, fornendo i dettagli in un file di configurazione dell'applicazione, come descritto di seguito: "Creazione amministrativa di un canale personalizzato WCF fornendo informazioni sul bind e sull'endpoint in un file di configurazione dell'applicazione" a pagina 607
- Programmaticamente direttamente dal codice dell'applicazione, come descritto nel seguente esempio.

Passo 1: crea un'istanza dell'elemento di collegamento del trasporto del canale

Aggiungere il codice seguente alla propria applicazione:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
```

Passo 2: impostare le proprietà di bind

Impostare le proprietà di bind richieste, ad esempio, aggiungendo il seguente codice alla tua applicazione per impostare `ClientConnectionMode`.

```
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.AS_URI;
```

Passo 3: crea un bind personalizzato che abbina il canale di trasporto con un codificatore di messaggi

Crea un bind personalizzato aggiungendo il seguente codice alla tua applicazione:

```
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
```

Fase 4: creazione dell'URI SOAP/JMS

L'URI SOAP/JMS che descrive i dettagli di connessione WebSphere MQ richiesti per accedere al servizio, deve essere fornito come indirizzo endpoint. Ciò dipende dal fatto che il canale venga utilizzato o meno per un'applicazione di servizio o per un'applicazione client.

Per le applicazioni client, l'URI SOAP/JMS deve essere creato come `EndpointAddress` come segue:

```
EndpointAddress address = new EndpointAddress("jms:/queue?
destination=SampleQ@QM1&connectionFactory=connectQueueManager(QM1)&initialContextFactory=com.ibm
.mq.jms.Nojndi");
```

Per le applicazioni di servizio, l'URI SOAP/JMS deve essere creato come URI come segue:

```
Uri address = new Uri("jms:/queue?
destination=SampleQ@QM1&connectionFactory=connectQueueManager(QM1)&initialContextFactory=com.ibm
.mq.jms.Nojndi");
```

Per ulteriori informazioni sull'indirizzo dell'endpoint, consultare: “WebSphere MQ per formato indirizzo URI endpoint WCF” a pagina 610

WebSphere MQ per formato indirizzo URI endpoint WCF

Un URI (Universal Resource Identifier) fornisce i dettagli di ubicazione e connessione per specificare un servizio web. Questo formato URI consente un grado di controllo completo su opzioni e parametri specifici di SOAP/ WebSphere MQquando si accede ai servizi di destinazione.

Un servizio Web viene specificato utilizzando un URI (Universal Resource Identifier). Questa sezione specifica il formato URI supportato nel trasporto WebSphere MQ per SOAP. Questo formato URI consente un grado di controllo completo su opzioni e parametri specifici di SOAP/WebSphere MQquando si accede ai servizi di destinazione. Questo formato è compatibile con WAS (WebSphere Application Server) e con CICS che facilita l'integrazione di WebSphere MQ con entrambi i prodotti.

La sintassi URI è la seguente:

```
jms:/queue?name=value&name=value...
```

dove `name` è il nome di un parametro e *valore* è il valore appropriato e l'elemento `name=value` può essere ripetuto un qualsiasi numero di volte con la seconda e le successive ricorrenze precedute da una e commerciale (&).

Per ulteriori informazioni sull'impostazione delle proprietà URI, consultare: Sintassi URI e parametri per la distribuzione del servizio web

I nomi dei parametri sono sensibili al maiuscolo / minuscolo, come i nomi degli oggetti WebSphere MQ . Se un parametro viene specificato più di una volta, la ricorrenza finale del parametro diventa effettiva, il che significa che le applicazioni client possono sovrascrivere i valori del parametro accodando all'URI. Se vengono inclusi ulteriori parametri non riconosciuti, vengono ignorati.

Se si memorizza un URI in una stringa XML, è necessario rappresentare il carattere e commerciale come "&". Allo stesso modo, se un URI è codificato in uno script, fare attenzione a eseguire l'escape di caratteri come **&** che altrimenti sarebbero interpretati dalla shell.

Questo è un esempio di un URI semplice per un servizio Axis:

```
jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Di seguito è riportato un esempio di URI semplice per un servizio .NET:

```
jms:/queue?destination=myQ&connectionFactory=()&targetService=MyService.asmx
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Vengono forniti solo i parametri richiesti (`targetService` è obbligatorio solo per i servizi .NET) e a `connectionFactory` non viene fornita alcuna opzione.

In questo esempio di asse, `connectionFactory` contiene una serie di opzioni:

```
jms:/queue?destination=myQ@myRQM&connectionFactory=connectQueueManager(myconnQM)
```

```
binding(client)clientChannel(myChannel)clientConnection(myConnection)
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

In questo esempio di asse, è stata specificata anche l'opzione `sslPeerName` di `connectionFactory` . Il valore di sslPeerName stesso contiene coppie nome - valore e significativi spazi incorporati:

```
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
```

Opzioni di configurazione del bind WCF

Questo argomento descrive come applicare le opzioni di configurazione alle informazioni di collegamento dei canali personalizzati ed elenca le opzioni disponibili.

Le opzioni di configurazione del collegamento possono essere impostate in due diversi modi:

1. Amministrativamente: le impostazioni delle proprietà di collegamento devono essere specificate nella sezione di trasporto della definizione di collegamento personalizzata nel file di configurazione delle applicazioni, ad esempio: `app.config`
2. Programmaticamente: il codice applicazione deve essere modificato per specificare la proprietà durante l'inizializzazione del bind personalizzato.

Impostazione delle proprietà di collegamento in modo amministrativo

Le impostazioni della proprietà di bind possono essere specificate anche nel file config dell'applicazione, ad esempio: `app.config`. Il file di configurazione viene generato da **`svcutil`**, ad esempio:

```
<customBinding>

...

     <IBM.XMS.WCF.SoapJmsIbmTransportChannel maxBufferPoolSize="524288"
      maxMessageSize="4000000" clientConnectionMode="0" maxConcurrentCalls="16"/>

...

</customBinding>
```

Impostazione programmatica delle proprietà di collegamento

Per aggiungere una proprietà di collegamento WCF per specificare la modalità di connessione client, è necessario modificare il codice servizio per specificare la proprietà durante l'inizializzazione del collegamento personalizzato.

Utilizzare il seguente esempio per specificare la modalità di connessione client non gestita:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.CLIENT_UNMANAGED;

Binding sampleBinding = new CustomBinding(new TextMessageEncodingBindingElement(),
                                                    transportBindingElement);
```

Proprietà bind WCF

Nome della proprietà	Applicazione client o servizio	Valore amministrativo	Valore programmatico	Descrizione
maxBufferPoolSize	Entrambi	Numero intero con segno da 0 a 64 bit	Numero intero con segno da 0 a 64 bit	Specifica la dimensione massima della memoria che può essere utilizzata per memorizzare i buffer di messaggi WCF per una istanza del canale.
maxMessageSize	Entrambi	Numero intero con segno da 1 a 32 bit	Numero intero con segno da 1 a 32 bit	Specifica la memoria massima che può essere utilizzata per un singolo messaggio WCF.
Modalità clientConnection	Entrambi	0 (Valore predefinito) 1	AS_URI (valore predefinito) CLIENT_NON gestito	Specifica la modalità di connessione client del canale di trasporto. 0 indica che la modalità di collegamento client è quella specificata nell'URI. Utilizzato solo se viene utilizzata la connessione client. Specifica che la modalità di connessione client è quella specificata nell'URI. 0 è il valore predefinito se non è impostata alcuna modalità di connessione client. 1 indica che la modalità di connessione client è un client non gestito. Utilizzato solo se viene utilizzata la connessione client.
Chiamate MaxConcurrent	Client	L'intervallo è 0-2 147 483 647 16 è il valore predefinito	L'intervallo è 0-2 147 483 647 16 è il valore predefinito	Questa proprietà definisce il numero massimo di operazioni simultanee che possono essere eseguite su un singolo proxy client in qualsiasi momento. Se vengono avviate più operazioni, queste vengono accodate fino al completamento o al timeout di un'operazione in corso. Questa impostazione può essere utilizzata per controllare il numero massimo di thread e risorse che possono essere utilizzati da un singolo proxy. 0 rimuove questo limite, consentendo il tentativo simultaneo di tutte le operazioni.

Nome della proprietà	Applicazione client o servizio	Valore amministrativo	Valore programmatico	Descrizione
Chiamate MaxConcurrent	Servizio	L'intervallo è 1-2 147 483 647 16 è il valore predefinito	L'intervallo è 1-2 147 483 647 16 è il valore predefinito	Questa proprietà viene utilizzata solo se la funzione di consegna garantita è abilitata (per ulteriori informazioni sulla consegna garantita, consultare “Distribuzione garantita canale personalizzato WCF” a pagina 602). Specifica il massimo numero di operazioni simultanee che possono essere in corso contemporaneamente per l'endpoint specificato. È necessario prestare attenzione quando si modifica questa impostazione. Ogni operazione simultanea richiede ulteriori risorse, in particolare una nuova istanza del canale personalizzato e i thread associati dal pool di thread per eseguire le richieste. L'allocazione eccessiva può essere controproducente e influire gravemente sulle prestazioni. È necessario eseguire la configurazione appropriata del pool di thread per supportare questa proprietà.

Servizi di costruzione e hosting per WCF

Panoramica dei servizi Microsoft Windows Communication Foundation (WCF) che illustra come creare e configurare i servizi WCF.

Il canale personalizzato IBM WebSphere MQ per WCF e i servizi WCF che lo utilizzano possono essere ospitati dai seguenti metodi:

- Self - hosting
- Servizio Windows

Il canale personalizzato IBM WebSphere MQ per WCF non può essere ospitato in Windows Process Activation Service.

I seguenti argomenti forniscono alcuni semplici esempi di auto - hosting per dimostrare le fasi coinvolte. La documentazione in linea di Microsoft WCF, che contiene ulteriori informazioni e i dettagli più recenti, è disponibile sul sito Web di MSDN Microsoft all'indirizzo https://msdn.microsoft.com.

Creazione di applicazioni di servizio WCF utilizzando il metodo 1: hosting autonomo mediante un file di configurazione dell'applicazione

Dopo aver creato un file di configurazione dell'applicazione, apri un'istanza del servizio e aggiungi il codice specificato alla tua applicazione.

Prima di iniziare

Creare o modificare un file di configurazione dell'applicazione per il servizio, come descritto in: “Creazione amministrativa di un canale personalizzato WCF fornendo informazioni sul bind e sull'endpoint in un file di configurazione dell'applicazione” a pagina 607

Informazioni su questa attività

1. Istanziare e aprire un'istanza del servizio nell'host del servizio. Il tipo di servizio deve essere uguale al tipo di servizio specificato nel file di configurazione del servizio.
2. Aggiungere il codice seguente alla propria applicazione:

```
ServiceHost service = new ServiceHost(typeof(MyService));
service.Open();
...
service.Close();
```

Creazione di applicazioni di servizio WCF utilizzando il metodo 2: Self - hosting programmaticamente direttamente dall'applicazione

Aggiungere le proprietà di bind, creare l'host di servizio con un'istanza della classe di servizio richiesta e aprire il servizio.

Prima di iniziare

1. Aggiungere un riferimento al canale personalizzato `IBM.XMS.WCF.dll` al progetto. `IBM.XMS.WCF.dll` si trova in *`WMQInstallDir`*`\bin` , dove *WMQInstallDir* è la directory in cui è installato WebSphere MQ 7.
2. Aggiungere un'istruzione *using* al namespace `IBM.XMS.WCF` , ad esempio: `using IBM.XMS.WCF`
3. Creare un'istanza dell'elemento bind dei canali e dell'endpoint come descritto in “Creazione di un canale personalizzato WCF mediante fornitura programmatica di informazioni di binding ed endpoint” a pagina 609

Informazioni su questa attività

Se sono richieste delle modifiche alle proprietà di bind del canale, completare la seguente procedura:

1. Aggiungere le proprietà di collegamento a `transportBindingElement` come mostrato nel seguente esempio:

```
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. Creare l'host di servizio con un'istanza della classe di servizio richiesta:

```
ServiceHost service = new ServiceHost(typeof(MyService));
```

3. Apri il servizio:

```
service.AddServiceEndpoint(typeof(IMyServiceContract), binding, address);
service.Open();
    ...
service.Close();
```

Esposizione di metadati utilizzando un endpoint HTTP

Istruzioni per l'esposizione dei metadati di un servizio configurato per utilizzare il canale personalizzato WebSphere MQ per WCF.

Informazioni su questa attività

Se i metadati dei servizi devono essere esposti (in modo che gli strumenti come `svcutil` possano accedervi direttamente dal servizio in esecuzione piuttosto che da un file WSDL non in linea, ad esempio), è necessario esporre i metadati dei servizi con un endpoint HTTP. I seguenti passi possono essere utilizzati per aggiungere questo endpoint aggiuntivo.

1. Aggiungere l'indirizzo di base in cui i metadati devono essere esposti a ServiceHost, ad esempio:

```
ServiceHost service = new ServiceHost(typeof(TestService),
                      new Uri("http://localhost:8000/MyService"));
```

2. Aggiungere il seguente codice a ServiceHost prima che il servizio venga aperto:

```
ServiceMetadataBehavior metadataBehavior = new ServiceMetadataBehavior();
  metadataBehavior.HttpGetEnabled = true;
  service.Description.Behaviors.Add(metadataBehavior);
  service.AddServiceEndpoint(typeof(IMetadataExchange),
     MetadataExchangeBindings.CreateMexHttpBinding(), "mex");
```

Risultati

I metadati sono ora disponibili al seguente indirizzo: `http://localhost:8000/MyService`

Creazione di applicazioni client per WCF

Panoramica sulla generazione e sulla creazione di applicazioni client Microsoft Windows Communication Foundation (WCF).

Un'applicazione client può essere creata per un servizio WCF; le applicazioni client vengono generalmente generate utilizzando lo Strumento di utilità metadati Microsoft ServiceModel (Svcutil.exe) per creare la configurazione richiesta e i file proxy che possono essere utilizzati direttamente dall'applicazione.

Generazione di un proxy client WCF e di file di configurazione dell'applicazione utilizzando lo strumento svcutil con metadati da un servizio in esecuzione

Istruzioni per l'utilizzo dello strumento svcutil.exe di Microsoft per generare un client per un servizio configurato per utilizzare il canale personalizzato WebSphere MQ per WCF.

Prima di iniziare

Esistono tre prerequisiti per l'utilizzo dello strumento svcutil per creare i file di configurazione e proxy richiesti che possono essere utilizzati direttamente dall'applicazione:

- Il servizio WCF deve essere in esecuzione prima dell'avvio dello strumento svcutil.
- Il servizio WCF deve esporre i propri metadati utilizzando una porta HTTP oltre ai riferimenti endpoint del canale personalizzato WebSphere MQ per generare un client direttamente da un servizio in esecuzione.
- Il canale personalizzato deve essere registrato nei dati di configurazione per svcutil.

Informazioni su questa attività

La seguente procedura spiega come generare un client per un servizio che è configurato per utilizzare il canale personalizzato WebSphere MQ , ma espone anche i relativi metadati al runtime tramite una porta HTTP separata:

1. Avviare il Servizio WCF (il Servizio deve essere in esecuzione prima dell'avvio dello strumento svcutil).
2. Aggiungere i dettagli dal file di configurazione `svcutil.exe` dalla root dell'installazione, nel file di configurazione svcutil attivo, in genere `C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\svcutil.exe.config` in modo che svcutil riconosca il canale personalizzato WebSphere MQ .
3. Eseguire svcutil da un prompt dei comandi, ad esempio:

```
svcutil /language:C# /r:<installlocation>\bin\IBM.XMS.WCF.dll
                /config:app.config http://localhost:8000/IBM.XMS.WCF/samples
```

4. Copiare i file `app.config` e `YourService.cs` generati nel progetto client Microsoft Visual Studio.

Operazioni successive

Se i metadati dei servizi non possono essere richiamati direttamente, svcutil può essere utilizzato per generare i file del client da wsdl. Per ulteriori informazioni, consultare: "Generazione di un proxy del client WCF e di file di configurazione dell'applicazione utilizzando lo strumento svcutil con WSDL" a pagina 616

Generazione di un proxy del client WCF e di file di configurazione dell'applicazione utilizzando lo strumento svcutil con WSDL

Istruzioni per la generazione di client WCF da WSDL se i metadati del servizio non sono disponibili.

Se i metadati del servizio non possono essere richiamati direttamente per generare un client dai metadati di un servizio in esecuzione, è possibile utilizzare svcutil per generare i file client da WSDL. È necessario apportare le seguenti modifiche al WSDL per specificare che deve essere utilizzato il canale personalizzato WebSphere MQ :

1. Aggiungere le seguenti definizioni di spazio dei nomi e informazioni sulla normativa:

```
<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. Modificare la sezione dei collegamenti per fare riferimento alla nuova sezione della politica e rimuovere qualsiasi definizione `transport` dall'elemento di collegamento sottostante:

```
<wsdl:definitions ...>

        <wsdl:binding ...>
                <wsp:PolicyReference URI="#CustomerBinding_IWMQSampleContract_policy" />
                <[soap]:binding ... transport="" />
                ...
        </wsdl:binding>
</wsdl:definitions>
```

3. Eseguire svcutil da un prompt dei comandi, ad esempio:

```
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
```

   Dove *MQ_INSTALLATION_PATH* è la directory di installazione di WebSphere MQ.

Creazione di applicazioni del client WCF utilizzando un proxy client con un file di configurazione dell'applicazione

Prima di iniziare

Creare o modificare un file di configurazione dell'applicazione per il client, come descritto in: "Creazione amministrativa di un canale personalizzato WCF fornendo informazioni sul bind e sull'endpoint in un file di configurazione dell'applicazione" a pagina 607

Informazioni su questa attività

Istanziare e aprire un'istanza del proxy client. Il parametro passato al proxy generato deve corrispondere al nome endpoint specificato nel file di configurazione client, ad esempio `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();
            }
```

Creazione di applicazioni client WCF utilizzando un proxy client con configurazione programmatica

Prima di iniziare

1. Aggiungere un riferimento al canale personalizzato `IBM.XMS.WCF.dll` al progetto. `IBM.XMS.WCF.dll` si trova nella directory *WMQInstallDir*`\bin` in cui *WMQInstallDir* è la directory in cui è installato WebSphere MQ 7.
2. Aggiungere un'istruzione *using* al namespace `IBM.XMS.WCF` , ad esempio: `using IBM.XMS.WCF`
3. Creare un'istanza dell'elemento di collegamento e dell'endpoint del canale come descritto in “Creazione di un canale personalizzato WCF mediante fornitura programmatica di informazioni di binding ed endpoint” a pagina 609

Informazioni su questa attività

Se sono richieste delle modifiche alle proprietà di bind del canale, completare la seguente procedura:

1. Aggiungere le proprietà di collegamento a `transportBindingElement` come mostrato nella seguente figura:

```
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. Creare il proxy client come mostrato nella seguente figura, dove *collegamento* e *indirizzo endpoint* sono il collegamento e l'indirizzo endpoint configurati nel passo “1” a pagina 617 e passati:

```
 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();
            }
```

Utilizzo degli esempi WCF

Gli esempi WCF (Windows Communication Foundation) forniscono alcuni semplici esempi di come può essere utilizzato il canale personalizzato WebSphere MQ .

Per creare i progetti di esempio, è necessario Microsoft .NET 3.5 SDK o Microsoft Visual Studio 2008.

Esempio WCF del server e del client a una via semplice

Questo esempio illustra il canale personalizzato WebSphere MQ utilizzato per avviare un servizio WCF (Communication foundation) Windows da un client WCF utilizzando una forma di canale unidirezionale.

Informazioni su questa attività

Il servizio implementa un singolo metodo che restituisce una stringa alla console. Il client è stato generato utilizzando lo strumento `svcutil` per richiamare i metadati del servizio da un endpoint HTTP esposto separatamente, come descritto in "Generazione di un proxy client WCF e di file di configurazione dell'applicazione utilizzando lo strumento svcutil con metadati da un servizio in esecuzione" a pagina 615

L'esempio è stato configurato con nomi risorsa specifici come descritto nella seguente procedura. Se è necessario modificare i nomi delle risorse, è necessario modificare anche il valore corrispondente sull'applicazione client nel file *MQ_INSTALLATION_PATH*`\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\app.config` e sull'applicazione del servizio nel file *MQ_INSTALLATION_PATH*`\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\TestServices.cs` , dove *MQ_INSTALLATION_PATH* è la directory di installazione per IBM WebSphere MQ. Per ulteriori informazioni sulla formattazione dell'URI dell'endpoint JMS, consultare *WebSphere MQ Transport for SOAP* nella documentazione del prodotto WebSphere MQ . Se è necessario modificare l'origine e la soluzione di esempio, è necessario un IDE, ad esempio Microsoft Visual Studio 8 o superiore.

Procedura

1. Creare un gestore code denominato *QM1*
2. Creare una destinazione code denominata *SampleQ*
3. Avviare il servizio in modo che il listener sia in attesa di messaggi: eseguire il file *MQ_INSTALLATION_PATH*`\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\bin\Release\TestService.exe` , dove *MQ_INSTALLATION_PATH* è la directory di installazione per IBM WebSphere MQ.
4. Eseguire il client una sola volta: eseguire il file *MQ_INSTALLATION_PATH*`\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\bin\Release\TestClient.exe` , dove *MQ_INSTALLATION_PATH* è la directory di installazione per IBM WebSphere MQ.

   L'applicazione client esegue un loop cinque volte inviando cinque messaggi a *SampleQ*

Risultati

L'applicazione del servizio riceve i messaggi da *SampleQ* e visualizza `Hello World` sullo schermo cinque volte.

Operazioni successive

Esempio WCF del server e del client di richiesta - risposta semplice

Questo esempio illustra il canale personalizzato WebSphere MQ utilizzato per avviare un servizio WCF (Communication foundation) Windows da un client WCF utilizzando una forma di canale richiesta - risposta.

Informazioni su questa attività

Questo servizio fornisce alcuni semplici metodi di calcolo per aggiungere e sottrarre due numeri, quindi restituire il risultato. Il client è stato generato utilizzando lo strumento `svcutil` per richiamare i metadati del servizio da un endpoint HTTP esposto separatamente, come descritto in "Generazione di un proxy client WCF e di file di configurazione dell'applicazione utilizzando lo strumento svcutil con metadati da un servizio in esecuzione" a pagina 615

L'esempio è stato configurato con nomi risorsa specifici come descritto nella seguente procedura. Se è necessario modificare i nomi delle risorse, è anche necessario modificare il valore corrispondente sull'applicazione client nel file *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\client\app.config` e sull'applicazione di servizio nel file *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\service\RequestReplyService.cs` , dove *MQ_INSTALLATION_PATH* è la directory di installazione per WebSphere MQ. Per ulteriori informazioni sulla formattazione dell'URI dell'endpoint JMS, consultare *WebSphere MQ Transport for SOAP* nella documentazione del prodotto WebSphere MQ . Se è necessario modificare l'origine e la soluzione di esempio, è necessario un IDE, ad esempio Microsoft Visual Studio 8 o superiore.

Procedura

1. Creare un gestore code denominato *QM1*
2. Creare una destinazione code denominata *SampleQ*
3. Creare una destinazione code denominata *SampleReplyQ*
4. Avviare il servizio in modo che il listener sia in attesa di messaggi: eseguire il file *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\service\bin\Release\SimpleRequestReply_Service.exe` , dove *MQ_INSTALLATION_PATH* è la directory di installazione per WebSphere MQ.
5. Eseguire il client una sola volta: eseguire il file *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\client\bin\Release\SimpleRequestReply_Client.exe` , dove *MQ_INSTALLATION_PATH* è la directory di installazione per WebSphere MQ.

Risultati

Quando il client è stato eseguito, viene avviato il seguente processo e si ripete quattro volte in modo che un totale di cinque messaggi venga inviato in ogni modo:

1. Il client inserisce un messaggio di richiesta su *SampleQ* e attende una risposta.
2. Il servizio riceve il messaggio di richiesta da *SampleQ*.
3. Il servizio aggiunge e sottrae alcuni valori utilizzando il contenuto del messaggio.
4. Il servizio inserisce i risultati in un messaggio su *SampleReplyQ*e attende che il client inserisci un nuovo messaggio.
5. Il client riceve il messaggio da *SampleReplyQ* e visualizza i risultati sullo schermo.

Operazioni successive

Client WCF per un servizio .NET ospitato dall'esempio WebSphere MQ

Le applicazioni client di esempio e le applicazioni proxy del servizio di esempio sono fornite sia per .NET che per Java. Gli esempi si basano su un servizio Stock Quote che acquisisce una richiesta per una quotazione in borsa e quindi fornisce la quotazione in borsa.

Prima di iniziare

L'esempio richiede che l'ambiente host del servizio .NET SOAP su JMS sia installato e configurato correttamente in WebSphere MQ ed è accessibile da un gestore code locale. Per informazioni

sull'installazione e la configurazione dell'ambiente, consultare: “Installazione del trasporto Web WebSphere MQ per SOAP” a pagina 959

Quando l'ambiente host del servizio .NET SOAP over JMS è installato e configurato correttamente in WebSphere MQ ed è accessibile da un gestore code locale, è necessario completare ulteriori fasi di configurazione.

1. Impostare la variabile di ambiente WMQSOAP_HOME sulla directory di installazione di WebSphere MQ , ad esempio: `C:\Program Files\IBM\WebSphere MQ`
2. Verificare che il compilatore Java `javac` sia disponibile e su PATH.
3. Copiare il file `axis.jar` dalla directory `prereqs/axis` del CD di installazione di WebSphere nella directory di produzione WebSphere MQ , ad esempio: `C:\Program Files\IBM\WebSphere MQ\java\lib\soap`
4. Aggiungere a PATH: *MQ_INSTALLATION_PATH*`\Java\lib` dove *MQ_INSTALLATION_PATH* rappresenta la directory in cui è installato WebSphere MQ , ad esempio: `C:\Program Files\IBM\WebSphere MQ`
5. Verificare che l'ubicazione di .NET sia specificata correttamente in *MQ_INSTALLATION_PATH*`\bin\amqwcallWSDL.cmd` dove *MQ_INSTALLATION_PATH* rappresenta la directory in cui è installato WebSphere MQ , ad esempio: `C:\Program Files\IBM\WebSphere MQ`. L'ubicazione di .NET può essere specificata ad esempio: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Una volta completati i passi precedenti, verificare ed eseguire il servizio:

1. Passare alla directory di lavoro SOAP su JMS.
2. Immettere uno dei seguenti comandi per eseguire il test di verifica e lasciare in esecuzione il listener di servizio:
   - Per .NET: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt dotnet hold` dove *MQ_INSTALLATION_PATH* rappresenta la directory in cui è installato WebSphere MQ .
   - Per AXIS:*MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt Dotnet2AxisClient hold` dove *MQ_INSTALLATION_PATH* indica la directory in cui è installato WebSphere MQ .

L'argomento `hold` mantiene i listener in esecuzione una volta completato il test.

Se vengono riportati degli errori durante questa configurazione, è possibile rimuovere tutte le modifiche in modo che la procedura possa essere riavviata nel seguente modo:

1. Eliminare il SOAP generato sulla directory JMS.
2. Eliminare il gestore code.

Informazioni su questa attività

Questo esempio illustra una connessione da un client WCF al servizio di esempio .NET SOAP over JMS fornito in WebSphere MQ utilizzando una forma di canale unidirezionale. Il servizio implementa un semplice esempio StockQuote , che restituisce una stringa di testo alla console.

Il client è stato generato utilizzando WSDL per generare i file client come descritto in “Generazione di un proxy del client WCF e di file di configurazione dell'applicazione utilizzando lo strumento svcutil con WSDL” a pagina 616

L'esempio è stato configurato con nomi risorsa specifici come descritto nella seguente procedura. Se è necessario modificare i nomi delle risorse, è necessario modificare anche il valore corrispondente sull'applicazione client nel file *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WMQNET\default\client\app.config` e sull'applicazione del servizio nel file *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WMQNET\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl` , dove *MQ_INSTALLATION_PATH* rappresenta la directory di installazione per WebSphere MQ. Per ulteriori informazioni sulla formattazione dell'URI dell'endpoint JMS, consultare *WebSphere MQ Transport for SOAP* nella documentazione del prodotto WebSphere MQ .

Procedura

Eseguire il client una sola volta: eseguire il file *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQNET\default\client\bin\Release\TestClient.exe , dove *MQ_INSTALLATION_PATH* rappresenta la directory di installazione per WebSphereWebSphere MQ.

L'applicazione client esegue un loop cinque volte inviando cinque messaggi alla coda di esempio.

Risultati

L'applicazione di servizio richiama i messaggi dalla coda di esempio e visualizza `Hello World` cinque volte sullo schermo.

Client WCF per un servizio Java Axis ospitato dall'esempio WebSphere MQ

Le applicazioni client di esempio e le applicazioni proxy del servizio di esempio vengono fornite sia per Java che per .NET. Gli esempi si basano su un servizio Stock Quote che acquisisce una richiesta per una quotazione in borsa e quindi fornisce la quotazione in borsa.

Prima di iniziare

Questo esempio richiede che l'ambiente di hosting del servizio .NET SOAP over JMS sia installato e configurato in WebSphere MQ ed è accessibile da un gestore code locale. Per informazioni sull'installazione e la configurazione dell'ambiente, consultare: "Installazione del trasporto Web WebSphere MQ per SOAP" a pagina 959

Quando l'ambiente host del servizio .NET SOAP over JMS è installato e configurato correttamente in WebSphere MQ ed è accessibile da un gestore code locale, è necessario completare ulteriori fasi di configurazione.

1. Impostare la variabile di ambiente WMQSOAP_HOME sulla directory di installazione di WebSphere MQ , ad esempio: `C:\Program Files\IBM\WebSphere MQ`
2. Verificare che il compilatore Java `javac` sia disponibile e su PATH.
3. Copiare il file `axis.jar` dalla directory `prereqs/axis` del CD di installazione di WebSphere nella directory di installazione di WebSphere MQ .
4. Aggiungere a PATH: *MQ_INSTALLATION_PATH*\Java\lib dove *MQ_INSTALLATION_PATH* rappresenta la directory in cui è installato WebSphere MQ , ad esempio: `C:\Program Files\IBM\WebSphere MQ`
5. Verificare che l'ubicazione di .NET sia specificata correttamente in *MQ_INSTALLATION_PATH*\bin\amqwcallWSDL.cmd dove *MQ_INSTALLATION_PATH* rappresenta la directory in cui è installato WebSphere MQ , ad esempio: `C:\Program Files\IBM\WebSphere MQ`. L'ubicazione di .NET può essere specificata ad esempio: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Una volta completati i passi precedenti, verificare ed eseguire il servizio:

1. Passare alla directory di lavoro SOAP su JMS.
2. Immettere uno dei seguenti comandi per eseguire il test di verifica e lasciare in esecuzione il listener di servizio:
   - Per .NET: *MQ_INSTALLATION_PATH*\Tools\soap\samples\runivt dotnet hold dove *MQ_INSTALLATION_PATH* rappresenta la directory in cui è installato WebSphere MQ .
   - Per AXIS:*MQ_INSTALLATION_PATH*\Tools\soap\samples\runivt Dotnet2AxisClient hold dove *MQ_INSTALLATION_PATH* indica la directory in cui è installato WebSphere MQ .

L'argomento `hold` mantiene i listener in esecuzione una volta completato il test.

Se durante questa configurazione vengono riportati degli errori, è possibile rimuovere tutte le modifiche in modo che la procedura venga riavviata nel modo seguente:

1. Eliminare il SOAP generato sulla directory JMS.

2. Eliminare il gestore code.

Informazioni su questa attività

L'esempio illustra una connessione da un client WCF al servizio di esempio Axis Java SOAP over JMS fornito in WebSphere MQ utilizzando una forma di canale unidirezionale. Il servizio implementa un esempio StockQuote semplice, che emette una stringa di testo in un file salvato nella directory corrente.

Il client è stato generato utilizzando WSDL per generare i file client come descritto in "Generazione di un proxy del client WCF e di file di configurazione dell'applicazione utilizzando lo strumento svcutil con WSDL" a pagina 616

L'esempio è stato configurato con nomi di risorsa specifici, come descritto in questo paragrafo. Se è necessario modificare i nomi delle risorse, è necessario modificare anche il valore corrispondente nell'applicazione client nel file *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WMQAxis\default\client\app.config` e nell'applicazione del servizio nel file *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WMQAxis\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl` , dove *MQ_INSTALLATION_PATH* rappresenta la directory di installazione per WebSphere MQ.

Procedura

Eseguire il client una sola volta: eseguire il file *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WMQAxis\default\client\bin\Release\TestClient.exe` , dove *MQ_INSTALLATION_PATH* rappresenta la directory di installazione per WebSphereWebSphere MQ.

L'applicazione client esegue un loop cinque volte inviando cinque messaggi alla coda di esempio.

Risultati

L'applicazione del servizio riceve i messaggi dalla coda di esempio e aggiunge `Hello World` cinque volte a un file nella directory corrente.

Riferimenti correlati

"Gestione di nomi di elementi di risposta SOAP differenti" a pagina 630
WCF prevede che il nome di un valore restituito sia in un formato specifico per impostazione predefinita, ma un servizio potrebbe non restituire un elemento con il relativo nome nel formato previsto.

Client WCF per il servizio Java ospitato dall'esempio WebSphere Application Server

Le applicazioni client di esempio e le applicazioni proxy del servizio di esempio vengono fornite per WAS (WebSphere Application Server) 6. Viene fornito anche un servizio richiesta - risposta.

Prima di iniziare

Questo esempio richiede che venga utilizzata la seguente configurazione di WebSphere MQ :

Tabella 77. WebSphere MQ configurazione richiesta

Oggetto	**Nome richiesto**
Gestore code	QM1
Coda locale	HelloWorld
Coda locale	Risposta HelloWorld

Questo esempio richiede inoltre che un ambiente host WebSphere Application Server V6 sia installato e configurato correttamente. WebSphere Application Server V6 utilizza una connessione in modalità bind per connettersi a WebSphere MQ per impostazione predefinita. Pertanto, WebSphere Application Server V6 deve essere installato sulla stessa macchina del gestore code.

Una volta configurato l'ambiente WAS, è necessario completare le seguenti operazioni di configurazione aggiuntive:

1. Creare i seguenti oggetti JNDI nel repository JNDI di WebSphere Application Server:
   a. Una destinazione coda JMS denominata `HelloWorld`
      - Impostare il nome JNDI su `jms/HelloWorld`
      - Impostare il nome della coda su `HelloWorld`
   b. Un factory di connessione code JMS denominato `HelloWorldQCF`
      - Impostare il nome JNDI su `jms/HelloWorldQCF`
      - Impostare il nome del gestore code su `QM1`
   c. Un factory di connessione code JMS denominato `WebServicesReplyQCF`
      - Impostare il nome JNDI su `jms/WebServicesReplyQCF`
      - Impostare il nome del gestore code su `QM1`
2. Creare una porta del listener dei messaggi denominata `HelloWorldPort` in WebSphere Application Server con la seguente configurazione:
   - Impostare il nome JNDI del factory di connessione su `jms/HelloWorldQCF`
   - Impostare il nome JNDI di destinazione su `jms/HelloWorld`
3. Installare l'applicazione `HelloWorldEJBEAR.ear` del servizio Web su WebSphere Application Server nel modo seguente:
   a. Fare clic su **Applicazioni** > **Nuova applicazione** > **Nuova applicazione enterprise**.
   b. Passare a *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` , dove *MQ_INSTALLATION_PATH* è la directory di installazione di WebSphere MQ.
   c. Non modificare alcuna opzione predefinita nella procedura guidata e riavviare il server delle applicazioni dopo che l'applicazione è stata installata.

Una volta completata la configurazione di WAS, verificare il servizio eseguendolo una volta:

1. Passare alla directory di lavoro Soap over JMS.
2. Immettere questo comando per eseguire l'esempio: *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\TestClient.exe` , dove *MQ_INSTALLATION_PATH* è la directory di installazione di WebSphere MQ.

Informazioni su questa attività

L'esempio illustra una connessione da un client WCF al servizio di esempio WebSphere Application Server SOAP over JMS fornito negli esempi WCF inclusi negli esempi WebSphere MQ V7, utilizzando una forma di canale richiesta - risposta. Il flusso di messaggi tra WCF e WebSphere Application Server utilizzando le code WebSphere MQ . Il servizio implementa il metodo `HelloWorld(...)` , che prende una stringa e restituisce un messaggio di saluto al client.

Il client è stato generato utilizzando lo strumento svcutil per richiamare i metadati del servizio da un endpoint HTTP esposto separatamente come descritto in “Generazione di un proxy client WCF e di file di configurazione dell'applicazione utilizzando lo strumento svcutil con metadati da un servizio in esecuzione” a pagina 615

L'esempio è stato configurato con nomi risorsa specifici come descritto nella seguente procedura. Se è necessario modificare i nomi delle risorse, è necessario modificare anche il valore corrispondente sull'applicazione client nel file *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\default\client\app.config` e sull'applicazione di servizio in *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` , dove *MQ_INSTALLATION_PATH* è la directory di installazione di WebSphere MQ. Per ulteriori informazioni sulla formattazione dell'URI dell'endpoint JMS, consultare Parametri e sintassi URI per la distribuzione del servizio Web .

Il servizio e client si basano sul servizio e sul client descritti nell'articolo IBM Developer *Building a JMS Web service using SOAP over JMS and WebSphere Studio*. Per ulteriori informazioni sullo sviluppo di servizi Web SOAP su JMS compatibili con il canale personalizzato WebSphere MQ WCF, consultare l'articolo pertinente all'indirizzo: https://www.ibm.com/developerworks/websphere/library/techarticles/0402_du/0402_du.html.

Procedura

Eseguire il client una sola volta: eseguire il file *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\default\client\bin\Release\TestClient.exe` , dove *MQ_INSTALLATION_PATH* è la directory di installazione per WebSphere MQ.

L'applicazione client avvia entrambi i metodi di servizio contemporaneamente, inviando due messaggi alla coda di esempio.

Risultati

L'applicazione del servizio ottiene i messaggi dalla coda di esempio e fornisce una risposta alla chiamata del metodo `HelloWorld(...)` che l'applicazione del client emette sulla console.

Determinazione dei problemi sul canale personalizzato WCF per WebSphere MQ

È possibile utilizzare la traccia WebSphere MQ per raccogliere informazioni dettagliate sulle varie parti del codice WebSphere MQ . Quando si utilizza Windows Communication Foundation (WCF), viene generato un output di traccia separato per la traccia del canale personalizzato WCF integrata con la traccia dell'infrastruttura WCF Microsoft .

L'abilitazione completa della traccia per il canale personalizzato WCF produce due file di output:

1. La traccia del canale personalizzato WCF integrata con la traccia dell'infrastruttura WCF Microsoft .
2. La traccia del canale personalizzato WCF integrata con XMS .NET.

Avendo due output di traccia, i problemi possono essere tracciati in ogni interfaccia utilizzando gli strumenti appropriati, ad esempio:

- Determinazione dei problemi WCF utilizzando gli strumenti Microsoft appropriati.
- WebSphere MQ Problemi del client MQI utilizzando il formato di traccia XMS .

Per semplificare l'abilitazione della traccia, lo stack di traccia .NET 3 TraceSource e XMS .NET sono entrambi controllati utilizzando una singola interfaccia, come descritto in: “Nomi file di traccia e configurazione WCF” a pagina 625.

Gerarchia di eccezioni del canale personalizzato WCF

I tipi di eccezione generati dal canale personalizzato sono congruenti con WCF e generalmente sono una TimeoutException o una CommunicationException (o una sottoclasse di CommunicationException).

Ulteriori dettagli della condizione di errore, se disponibili, vengono forniti utilizzando eccezioni interne o collegate. Le seguenti eccezioni sono esempi tipici e ogni livello nell'architettura del canale fornisce un'altra eccezione collegata, ad esempio; CommunicationsException ha una XMSException collegata, che ha una MQException collegata:

1. System.serviceModel.CommunicationsExceptions
2. IBM.XMS.XMSException
3. IBM.WMQ.MQException

Le informazioni chiave vengono acquisite e fornite nella raccolta dati della CommunicationException più elevata nella gerarchia. Questa acquisizione e fornitura di dati evita la necessità per le applicazioni di collegarsi a ciascun livello nell'architettura del canale per interrogare le eccezioni collegate e tutte le informazioni aggiuntive che potrebbero contenere. Sono definiti i seguenti nomi chiave:

- IBM.XMS.WCF.ErrorCode: il codice del messaggio di errore dell'eccezione del canale personalizzato corrente.
- IBM.XMS.ErrorCode: il messaggio di errore della prima eccezione XMS nello stack.
- IBM.WMQ.ReasonCode: il codice motivo WebSphere MQ sottostante.
- IBM.WMQ.CompletionCode: il codice di completamento WebSphere MQ sottostante.

Nomi file di traccia e configurazione WCF

Quando la traccia è completamente abilitata, produce due file di output, uno per la diagnosi dei problemi WCF e un file dettagliato per il materiale di diagnostica della traccia interna. Per semplificare l'abilitazione della traccia, gli stack di traccia .NET 3 TraceSource e XMS .NET utilizzano un'interfaccia singola.

Sono disponibili due metodi di traccia differenti per il canale personalizzato WCF, i due metodi di traccia vengono attivati indipendentemente o insieme. Ogni metodo produce il proprio file di traccia, quindi quando entrambi i metodi di traccia sono stati attivati, vengono generati due file di output di traccia.

Per mantenere la configurazione e l'abilitazione il più semplici possibile, viene utilizzata la stessa interfaccia per controllare entrambi i metodi di traccia. Il file `app.config` deve essere modificato per includere la relativa configurazione di traccia come descritto nella seguente sezione. Gli utenti possono quindi aggiungere le proprie sezioni equivalenti per combinare l'emissione con la traccia dalla propria applicazione.

La traccia del canale personalizzato WCF non è abilitata per default. È necessario prima creare un listener di traccia, quindi impostare il livello di traccia richiesto per l'origine di traccia selezionata nel file `app.config` .

Configurazione del canale personalizzato WCF con la traccia dell'infrastruttura WCF

Aggiungere la seguente sezione di codice nella sezione `<system.diagnostics><sources>` nel file `app.config` :

```
<source name="IBM.XMS.WCF" switchValue="Verbose,ActivityTracing">
    <listeners>
        <remove name="Default"/>
        <add name="NewListener"/>
    </listeners>
</source>
```

La parte di codice precedente crea la traccia del canale utilizzando .NET 3 TraceSource. Tutte le chiamate dei file di configurazione associati ai file eseguibili sono controllate da questa parte di codice.

Configurazione del canale personalizzato WCF con la traccia .NET XMS

La configurazione della traccia XMS .NET richiede l'aggiunta di una sezione di codice alla sezione `<system.diagnostics><sources>` nel file `app.config` . Tuttavia, la parte di codice viene aggiunta all'elemento `<source>` estensibile mostrato nella sezione Configurazione del canale personalizzato WCF con la traccia dell'infrastruttura WCF . Pertanto, anche se il codice di traccia dell'infrastruttura WCF deve essere presente per il funzionamento della traccia XMS .NET, la traccia dell'infrastruttura WCF può essere disabilitata se non è richiesta, come descritto nella sezione Abilitazione della traccia 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>
```

Variabili di configurazione della traccia WCF

Tabella 78. Variabili di configurazione della traccia WCF

Variabile	Descrizione
nome	Specificare il nome come: `IBM.XMS.WCF`
switchValue	switchValue controlla il livello di traccia. Quando switchValue è impostata su `Off`, l'infrastruttura WCF TraceSource non viene generata. Qualsiasi altro valore, come ad esempio `Verbose`, genera TraceSource. Per informazioni dettagliate sul livello di traccia da Microsoft, consultare la documentazione WCF o visitare la pagina Web Microsoft WCF Tracing: https://msdn.microsoft.com/en-us/library/ms733025(vs.85).aspx

xmsTraceSpecificazione =*ComponentName*=*type*=*state*	*ComponentName* è il nome della classe che si desidera tracciare. È possibile utilizzare un carattere jolly * in questo nome. Ad esempio:
	`*=all=enabled`
	specifica che si desidera tracciare tutte le classi e
	`IBM.XMS.impl.*=all=enabled`
	specifica che è necessaria solo la traccia API.
	type può essere uno dei seguenti tipi di traccia:
	• tutti
	• Debug
	• evento
	• EntryExit
	state può essere abilitato o disabilitato.

xmsTraceFilePath= "*nomefile*"	Se non si specifica xmsTraceFilePatho se xmsTraceFilePath è presente ma contiene una stringa vuota, il file di traccia viene collocato nella directory corrente. Per memorizzare il file di traccia in una directory denominata, specificare il nome della directory in xmsTraceFilePath, ad esempio: `xmsTraceFilePath="c:\somepath"`
xmsTraceFileSize= "*dimensione*"	La dimensione massima consentita del file di traccia. Quando un file raggiunge questa dimensione, viene archiviato e ridenominato. Il valore massimo predefinito è 20 KB, che è specificato come: `xmsTraceFileSize="20000000".`
xmsTraceFileNumber= "*numero*"	Il numero di file di traccia da conservare. Il valore predefinito è 4 (un file attivo e tre file di archivio). Il numero minimo consentito è due.

Tabella 78. Variabili di configurazione della traccia WCF (Continua)	
Variabile	**Descrizione**

xmsTraceFormat="*formato*"	Esistono due livelli di formato xmsTrace: `basic` e `advanced`. Il formato di traccia predefinito è di base se non si specifica un Formato xmsTraceo se il formato xmsTraceè presente ma contiene una stringa vuota. I file di traccia vengono prodotti in questo formato se si specifica:
	`xmsTraceFormat="basic"`
	Se si richiede una traccia compatibile con gli strumenti del programma di analisi della traccia, è necessario specificare:
	`traceFormat="advanced"`

Abilitazione della traccia WCF

Esistono quattro combinazioni per abilitare e disabilitare i due diversi metodi di traccia. Le quattro combinazioni richiedono la modifica dei valori delle sezioni codificate descritte nelle sezioni precedenti.

È anche possibile impostare una variabile di ambiente; per ulteriori informazioni, consultare "Abilitazione della traccia WCF con la variabile di ambiente WCF_TRACE_ON" a pagina 628.

Questa tabella e i valori visualizzati dipendono dalle parti di codice dimostrate in precedenza che sono già state aggiunte al file `app.config` .

Tabella 79. Combinazioni di abilitazione traccia WCF.		
Tipo traccia	**Valore modificato**	**Esempio**
Traccia XMS abilitata. WCF TraceSource abilitato	`switchValue` non è impostato su `Off`	`<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>`
Traccia XMS abilitata. WCF TraceSource disabilitata	`switchValue` è impostato su `Disattivo` ed è stato fornito un `xmsTraceSpecification`	`<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>`

Tabella 79. Combinazioni di abilitazione traccia WCF. (Continua)

Tipo traccia	Valore modificato	Esempio
Traccia XMS disabilitata. WCF TraceSource abilitato	Esistono due modi per ottenere questo risultato: • La variabile `switchValue` non è impostata su `Off` e non è stato aggiunto un`xmsTraceSpecification` • La variabile `switchValue` non è impostata su `Off` e `xmsTraceSpecification` è stato impostato su `disabled`	`<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>`
Traccia XMS disabilitata. WCF TraceSource disabilitata	Esistono tre modi per ottenere questo risultato: • Nessun elemento `<source>` nel file `app.config` • La variabile `switchValue` è impostata su `Off` e non è stato aggiunto un`xmsTraceSpecification` • La variabile `switchValue` è impostata su `Off` e `xmsTraceSpecification` è stato impostato su `disabled`	`<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>`

Abilitazione della traccia WCF con la variabile di ambiente WCF_TRACE_ON

Oltre ai precedenti metodi descritti per abilitare la traccia WCF, la traccia .NET XMS può essere abilitata anche utilizzando la variabile di ambiente WCF_TRACE_ON.

L'impostazione della variabile di ambiente WCF_TRACE_ON su un qualsiasi valore non null equivale all'impostazione di `xmstraceSpecification` su `*=all=enabled`, ad esempio: "`set WCF_TRACE_ON=true`"

Tuttavia, se il `xmstraceSpecification` è esplicitamente impostato nel file `app.config` , la variabile di ambiente WCF_TRACE_ON viene sovrascritta.

File di output di traccia WCF e nomi file

I file di traccia XMS sono tradizionalmente denominati utilizzando il nome di base e il formato ID processo di: `xms_trace_`*`pid`*`.log`, dove *pid* è l'ID processo.

Poiché i file di traccia di XMS possono ancora essere prodotti in parallelo con i file di traccia del canale personalizzato WCF, la traccia del canale personalizzato WCF integrata con i file di output di traccia XMS .NET ha il seguente formato per evitare confusione: `wcfxms_trace_`*`pid`*`.log`, dove *pid* è l'ID processo.

Il file di output di traccia viene creato nella directory di lavoro corrente per impostazione predefinita, ma questa destinazione può essere ridefinita se necessario.

WCF XMS First Failure Support Technology (FFST)

È possibile raccogliere informazioni dettagliate sulle varie parti del codice di WebSphere MQ utilizzando la traccia WebSphere MQ . XMS FFST ha i propri file di configurazione e output per il canale personalizzato WCF.

I file di traccia XMS FFST sono tradizionalmente denominati utilizzando il formato ID processo e nome di base: `xmsffdc`*`pid_date`*`.txt`, dove *pid* è l'ID processo e *date* è la data e l'ora.

Poiché i file di traccia XMS FFST possono ancora essere prodotti in parallelo con i file XMS FFST del canale personalizzato WCF, i file di output XMS FFST del canale personalizzato WCF hanno il seguente formato per evitare confusione: `wcfffdc`*`pid_date`*`.txt`, dove *pid* è l'ID processo e *date* è la data e l'ora.

Questo file di output di traccia viene creato nella directory di lavoro corrente per impostazione predefinita, ma questa destinazione può essere ridefinita se necessario.

Il canale personalizzato WCF con intestazione di traccia XMS .NET è simile al seguente esempio:

```
************* Start Display XMS WCF Environment *************
  Product Name  :- value
  WCF Version   :- value
  Level    :- value
*************  End Display XMS WCF Environment  *************
```

I file di traccia FFST vengono formattati in modo standard, senza alcuna formattazione specifica per il canale personalizzato.

Informazioni sulla versione WCF

Le informazioni sulla versione WCF aiutano nella determinazione dei problemi e sono incluse nei metadati di assemblaggio del canale personalizzato.

Il canale personalizzato WebSphere MQ per metadati della versione WCF può essere richiamato in uno dei tre seguenti modi:

- Utilizzo del programma di utilità WebSphere MQ dspmqver. Per informazioni su come utilizzare dspmqver, consultare: dspmqver
- Utilizzando la finestra delle proprietà di Windows Explorer: in Windows Explorer, fare clic con il tasto destro del mouse su **IBM.XMS.WCF.dll** > **Proprietà** > **Versione**.
- Dalle informazioni di intestazione di uno dei canali FFST o dei file di traccia. Per ulteriori informazioni sull'intestazione FFST , consultare: “WCF XMS First Failure Support Technology (FFST)” a pagina 628

Suggerimenti e consigli WCF

I seguenti suggerimenti non sono in ordine significativo e potrebbero essere aggiunti quando vengono rilasciate nuove versioni della documentazione. Sono soggetti che potrebbero farti risparmiare tempo se sono rilevanti per il lavoro che stai facendo.

Esternalizzazione delle eccezioni dall'host del servizio WCF

Per i servizi ospitati utilizzando l'host del servizio WCF; eventuali eccezioni non gestite generate dal servizio, gli interni WCF o lo stack del canale non vengono esternalizzati per impostazione predefinita. Per essere informati di queste eccezioni, è necessario registrare un gestore errori.

Il seguente codice fornisce un esempio di definizione del comportamento del servizio del gestore errori che può essere applicato come un attributo di un servizio:

```
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)
        {
        }
    }
```

Gestione di nomi di elementi di risposta SOAP differenti

WCF prevede che il nome di un valore restituito sia in un formato specifico per impostazione predefinita, ma un servizio potrebbe non restituire un elemento con il relativo nome nel formato previsto.

WCF ha la convenzione di prevedere che il valore restituito venga denominato nel seguente formato: *methodName*`Result` dove *methodName* è il nome dell'operazione del servizio. Ad esempio, per un servizio denominato `getQuote`, WCF prevede che la risposta venga chiamata: `getQuoteResult`.

Tuttavia, il servizio può restituire un elemento con un nome non conforme a tale formato.

Quando si esegue lo strumento scvutil per generare un client proxy, se WSDL specifica un nome diverso, l'interfaccia proxy aggiunge parametri per indicare a WCF il nome da ricercare. Ad esempio:

```
[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);
```

Se si crea la propria interfaccia (ad esempio, aggiungendo un metodo richiesta - risposta a un'interfaccia proxy esistente), è necessario assicurarsi di aggiungere gli stessi parametri all'interfaccia se il servizio restituisce un nome diverso. In caso contrario, il problema più comune è che una chiamata al metodo di servizio restituisce sempre un valore null; se viene restituito un oggetto, il metodo restituisce un valore null, ma se viene restituito un valore numerico come un numero intero, il metodo restituisce 0.

Usando C++

WebSphere MQ fornisce classi C++ equivalenti agli oggetti di WebSphere MQ e alcune classi aggiuntive equivalenti ai tipi di dati array. Fornisce un certo numero di funzioni non disponibili tramite MQI.

A partire da WebSphere MQ Versione 7.0, i miglioramenti alle interfacce di programmazione WebSphere MQ non verranno applicati alle classi C + +.

WebSphere MQ C++ fornisce le funzioni riportate di seguito:

- Inizializzazione automatica delle strutture dati WebSphere MQ .
- Connessione e apertura della coda del gestore code just-in-time.
- Chiusura implicita della coda e disconnessione del gestore code.
- Ricezione e trasmissione dell'intestazione della lettera non recapitata.

- IMS trasmissione e ricezione dell'intestazione bridge.
- Trasmissione e ricezione dell'intestazione del messaggio di riferimento.
- Attivare la ricevuta del messaggio.
- Ricezione e trasmissione dell'intestazione bridge CICS .
- Trasmissione e ricezione intestazione lavoro.
- Definizione di canale client.

I seguenti diagrammi di classe di Booch mostrano che tutte le classi sono sostanzialmente parallele a quelle entità WebSphere MQ nell'MQI procedurale (ad esempio utilizzando C) che hanno handle o strutture di dati. Tutte le classi ereditano dalla classe ImqError (consultare ImqError C++ class), che consente di associare una condizione di errore a ciascun oggetto.

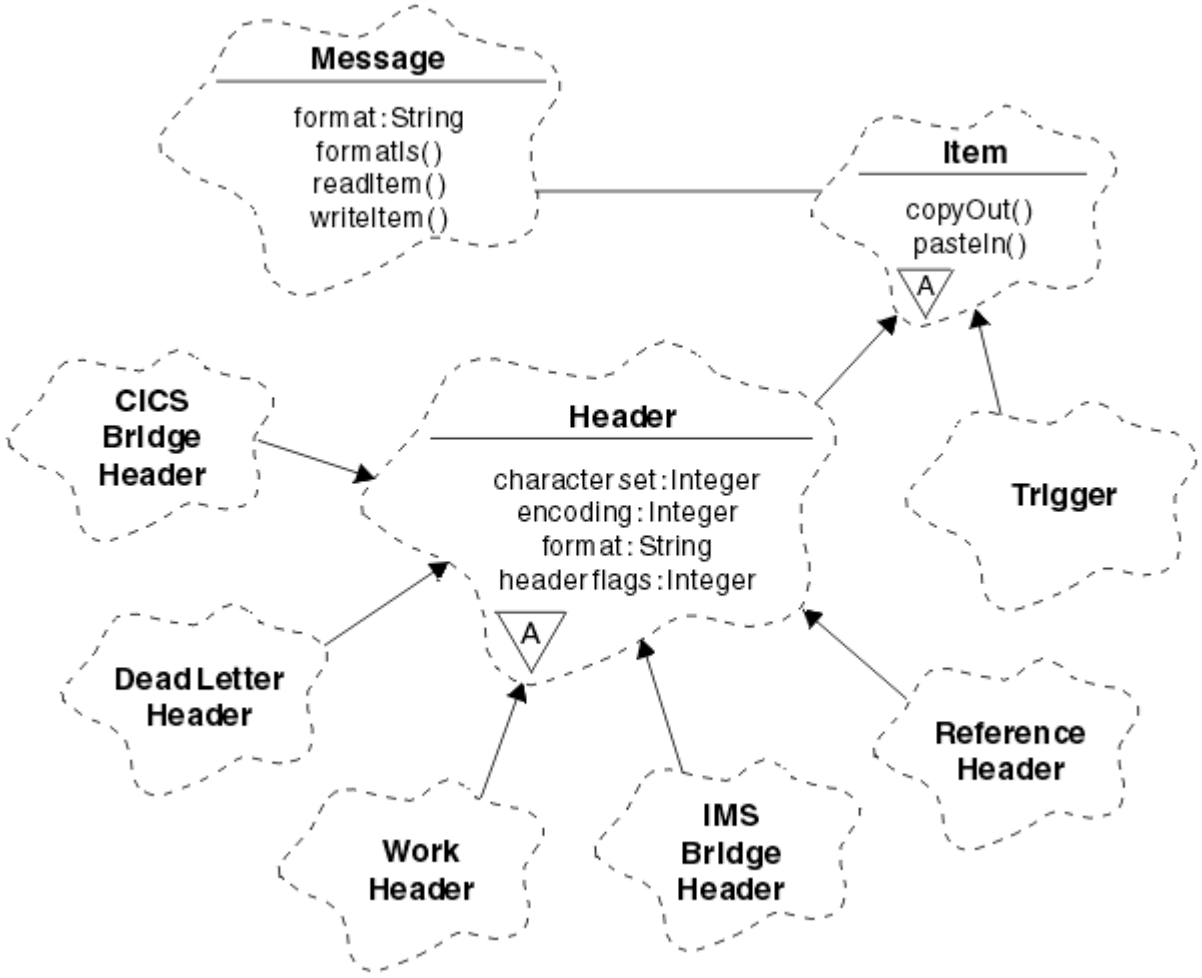


Figura 120. WebSphere MQ Classi C++ (gestione elementi)

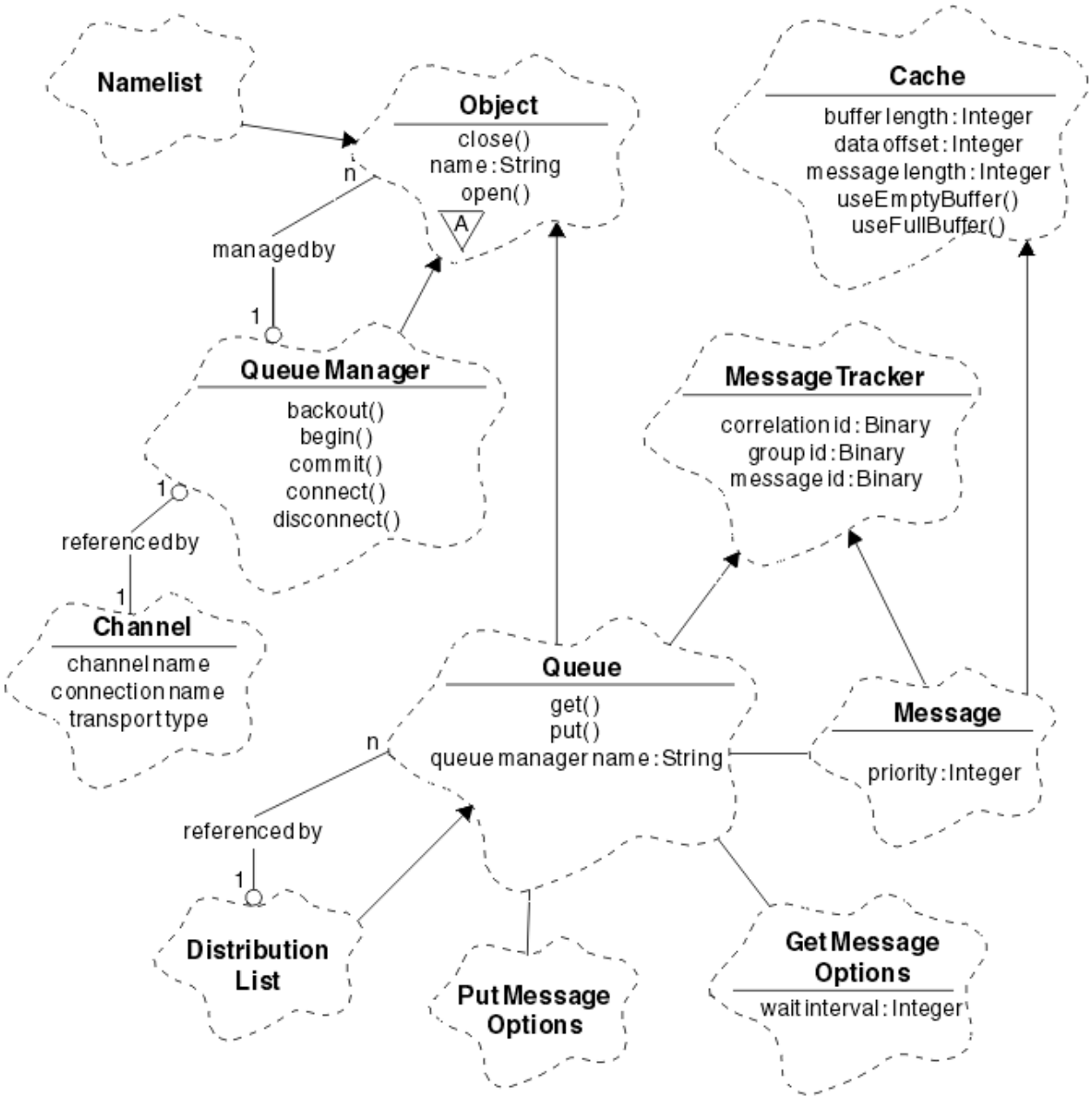


Figura 121. WebSphere MQ Classi C++ (gestione code)

Per interpretare correttamente i diagrammi di classe Booch, tenere presente le seguenti convenzioni:

- I metodi e gli attributi degni di nota sono riportati sotto il nome della *classe* .
- Un piccolo triangolo all'interno di un cloud denota una *classe astratta* .
- L' *eredità* è indicata da una freccia alla classe parent.
- Una linea non decorata tra i cloud denota una *relazione di cooperazione* tra le classi.
- Una riga decorata con un numero indica una *relazione referenziale* tra due classi. Il numero indica il numero di oggetti che possono partecipare a una particolare relazione in qualsiasi momento.

Le seguenti classi e tipi di dati vengono utilizzati nelle firme del metodo C+ + delle classi di gestione code (consultare Figura 121 a pagina 632) e le classi di gestione elementi (consultare Figura 120 a pagina 631):

- La classe ImqBinary (consultare ImqBinary C++ class), che incapsula gli array di byte come MQBYTE24.
- Il tipo di dati ImqBoolean , definito come **typedef unsigned char ImqBoolean**.

- La classe ImqString (consultare ImqString classe C++), che incapsula gli array di caratteri come MQCHAR64.

Le entità con strutture di dati vengono utilizzate all'interno delle classi di oggetti appropriate. I singoli campi della struttura dati (vedere Riferimento incrociato C++ e MQI) sono accessibili con i metodi.

Le entità con handle rientrano nella gerarchia di classi ImqObject (consultare ImqObject classe C++) e forniscono interfacce incapsulate a MQI. Gli oggetti di queste classi presentano un comportamento intelligente che può ridurre il numero di richiami del metodo richiesti rispetto all'MQI procedurale. Ad esempio, è possibile stabilire ed eliminare le connessioni del gestore code in base alle esigenze oppure è possibile aprire una coda con le opzioni appropriate, quindi chiuderla.

La classe ImqMessage (consultare ImqMessage classe C++) incapsula la struttura di dati MQMD e funge anche da punto di riferimento per i dati utente e gli *elementi* (consultare “Lettura dei messaggi in C++” a pagina 642) fornendo funzioni di buffer memorizzate nella cache. È possibile fornire buffer a lunghezza fissa per i dati utente e utilizzare il buffer molte volte. La quantità di dati presenti nel buffer può variare da un utilizzo all'altro. In alternativa, il sistema può fornire e gestire un buffer di lunghezza flessibile. Sia la dimensione del buffer (la quantità disponibile per la ricezione dei messaggi) che la quantità effettivamente utilizzata (o il numero di byte per la trasmissione o il numero di byte effettivamente ricevuti) diventano considerazioni importanti.

Concetti correlati

Panoramica tecnica

“Programmi di esempio C++” a pagina 633
Vengono forniti quattro programmi di esempio, per dimostrare la ricezione e l'inserimento di messaggi.

“Considerazioni sul linguaggio C++” a pagina 637
Questa raccolta di argomenti descrive in dettaglio gli aspetti dell'utilizzo del linguaggio C++ e le convenzioni che è necessario considerare quando si scrivono programmi applicativi che utilizzano MQI (Message Queue Interface).

“Preparazione dei dati dei messaggi in C + +” a pagina 641
I dati del messaggio vengono preparati in un buffer, che può essere fornito dal sistema o dall'applicazione. Ci sono vantaggi in entrambi i metodi. Vengono forniti esempi di utilizzo di un buffer.

“Scelta del linguaggio di programmazione da utilizzare” a pagina 79
Utilizza queste informazioni per scoprire i linguaggi di programmazione e i framework supportati da IBM WebSphere MQe alcune considerazioni per utilizzarli.

“Sviluppo delle applicazioni” a pagina 7
IBM WebSphere MQ fornisce diversi modi in cui è possibile sviluppare applicazioni per inviare e ricevere messaggi necessari per supportare i processi aziendali. È anche possibile sviluppare applicazioni per gestire i gestori code e le risorse correlate.

Riferimenti correlati

“Creazione di programmi C+ + WebSphere MQ” a pagina 648
Viene elencato l'URL dei compilatori supportati, insieme ai comandi da utilizzare per compilare, collegare ed eseguire programmi ed esempi C + + su piattaforme WebSphere MQ .

Riferimento incrociato C++ e MQI

Classi WebSphere MQ C++

Programmi di esempio C++

Vengono forniti quattro programmi di esempio, per dimostrare la ricezione e l'inserimento di messaggi.

I programmi di esempio sono:

- HELLO WORLD (imqwrld.cpp)
- SPUT (imqsput.cpp)
- SGET (imqsget.cpp)
- DPUT (imqdput.cpp)

I programmi di esempio si trovano nelle directory mostrate in Tabella 80 a pagina 634.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Tabella 80. Ubicazione dei programmi di esempio

Ambiente	**Directory contenente l'origine**	**Directory contenente la build Programmi**
AIX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/ia
HP-UX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/ samp/bin/ah (vedere la nota “2” a pagina 634)
Solaris	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/as
Linux	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/
Finestre	*MQ_INSTALLATION_PATH*\tools\cplus\ samples	*MQ_INSTALLATION_PATH*\tools\cplus\ esempi \bin\vn (vedere la nota “3” a pagina 634)

Note:

1. I programmi creati utilizzando il compilatore ILE C+ + per IBM i si trovano nella libreria QMQM. I file di inclusione sono in /QIBM/ProdData/mqm/inc.
2. I programmi creati utilizzando il compilatore HP ANSI C++ si trovano nella directory *MQ_INSTALLATION_PATH*/samp/bin/ah. Per ulteriori informazioni, fare riferimento a “Creazione di programmi C+ + su HP-UX” a pagina 649.
3. I programmi creati utilizzando Microsoft Visual Studio si trovano in *MQ_INSTALLATION_PATH*\tools\cplus\samples\bin\vn. Per ulteriori informazioni su questi compilatori, consultare “Creazione di programmi C+ + su Windows” a pagina 654.

Programma di esempio HELLO WORLD (imqwrld.cpp)

Questo programma di esempio C++ mostra come inserire e richiamare un datagramma regolare (struttura C) utilizzando la classe ImqMessage .

Questo programma mostra come inserire e richiamare un datagramma regolare (struttura C) utilizzando la classe ImqMessage . Questo esempio utilizza pochi richiami del metodo, sfruttando i richiami del metodo impliciti come **open**, **close**e **disconnect**.

Su tutte le piattaforme tranne z/OS

Se si utilizza una connessione server a WebSphere MQ, seguire una delle procedure riportate di seguito:

- Per utilizzare la coda predefinita esistente, SYSTEM.DEFAULT.LOCAL.QUEUE, eseguire il programma **imqwrlds** senza passare alcun parametro
- Per utilizzare una coda assegnata dinamicamente temporanea, eseguire **imqwrlds** inoltrando il nome della coda modello predefinita, SYSTEM.DEFAULT.MODEL.QUEUE.

Se si utilizza una connessione client a WebSphere MQ, seguire una delle procedure riportate di seguito:

- Configurare la variabile di ambiente MQSERVER (consultare MQSERVER per ulteriori informazioni) ed eseguire **imqwrldc**oppure

- Eseguire **imqwrldc** passando come parametri **queue-name**, **queue-manager-name**e **channel-definition**, dove un tipico **channel-definition** potrebbe essere SYSTEM.DEF.SVRCONN/TCP/*nomehost*(1414)

Codice d'esempio

```
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 ;
}
```

Programmi di esempio SPUT (imqsput.cpp) e SGET (imqsget.cpp)

Questi programmi C+ + inseriscono e richiamano messaggi da una coda denominata.

Questi esempi mostrano l'utilizzo delle seguenti classi:

- ImqError (consultare ImqError C++ class)
- ImqMessage (consultare ImqMessage C++ class)
- ImqObject (consultare ImqObject C++ class)
- ImqQueue (consultare ImqQueue classe C++)
- ImqQueueManager (consultare ImqQueueClasse C++ Manager)

Seguire le istruzioni appropriate per eseguire i programmi.

Su tutte le piattaforme tranne z/OS

1. Eseguire **imqsputs** *nome - coda*.
2. Immettere le righe di testo nella console. Queste righe vengono poste come messaggi nella coda specificata.
3. Immettere una riga nulla per terminare l'input.
4. Eseguire **imqsgets** *nome coda* per recuperare tutte le linee e visualizzarle nella console.

Programma di esempio DPUT (imqdput.cpp)

Questo programma di esempio C++ inserisce i messaggi in un elenco di distribuzione composto da due code.

DPUT mostra l'utilizzo della classe List ImqDistribution(consultare ImqDistributionList C++ class). Questo esempio non è supportato su z/OS.

1. Eseguire **imqdputs** *queue-name-1 queue-name-2* per inserire i messaggi nelle due code denominate.
2. Eseguire **imqsgets** *queue-name-1* e **imqsgets** *queue-name-2* per recuperare i messaggi da queste code.

Considerazioni sul linguaggio C++

Questa raccolta di argomenti descrive in dettaglio gli aspetti dell'utilizzo del linguaggio C++ e le convenzioni che è necessario considerare quando si scrivono programmi applicativi che utilizzano MQI (Message Queue Interface).

File di intestazione C+ +

I file di intestazione vengono forniti come parte della definizione di MQI, per consentire all'utente di scrivere programmi di applicazione WebSphere MQ nel linguaggio C + +.

Questi file di intestazione sono riepilogati nella seguente tabella.

Tabella 81. File di intestazione C/C++

Nome file	Contenuto
IMQI.HPP	Classi C++ MQI (include CMQC.H e IMQTYPE.H)
IMQTYPE.H	Definisce il tipo di dati **ImqBoolean**
CMQC.H	Strutture di dati MQI e costanti manifest

Per migliorare la portabilità delle applicazioni, codificare il nome del file di intestazione in minuscolo nella direttiva del preprocessore **#include** :

```
#include <imqi.hpp> // C++ classes
```

Metodi e attributi C++

I nomi dei metodi sono in caratteri misti. Varie considerazioni si applicano ai parametri e ai valori di ritorno. È possibile accedere agli attributi utilizzando i metodi set e get, come appropriato.

I parametri dei metodi *const* sono solo per l'immissione. Parametri con firme che includono un puntatore (*) o un riferimento (&) sono passati per riferimento. I valori di ritorno che non includono un puntatore o un riferimento vengono passati in base al valore; nel caso di oggetti restituiti, si tratta di nuove entità che diventano responsabilità del chiamante.

Alcune firme del metodo includono elementi che, se non specificati, assumono un valore predefinito. Tali elementi si trovano sempre alla fine delle firme e sono indicati da un segno uguale (=); il valore dopo il segno uguale indica il valore predefinito che si applica se l'elemento viene omesso.

Tutti i nomi dei metodi in queste classi sono maiuscoli e minuscoli. Ogni parola, tranne la prima all'interno di un nome metodo, inizia con una lettera maiuscola. Le abbreviazioni non sono usate a meno che il loro significato non sia ampiamente compreso. Le abbreviazioni utilizzate includono *id* (per l'identità) e *sync* (per la sincronizzazione).

È possibile accedere agli attributi dell'oggetto utilizzando i metodi set e get. Un metodo set inizia con la parola *set*; un metodo get non ha prefisso. Se un attributo è di *sola lettura*, non esiste alcun metodo set.

Gli attributi vengono inizializzati in stati validi durante la realizzazione dell'oggetto e lo stato di un oggetto è sempre congruente.

Tipi di dati in C++

Tutti i dati sono definiti dall'istruzione C **typedef** .

Il tipo **ImqBoolean** è definito come **unsigned char** in IMQTYPE.H e può avere i valori TRUE e FALSE.
È possibile utilizzare gli oggetti della classe **ImqBinary** al posto degli array **MQBYTE** e gli oggetti della classe **ImqString** al posto di **char ***. Molti metodi restituiscono oggetti invece di puntatori **char** o **MQBYTE** per semplificare la gestione della memoria. Tutti i valori di ritorno diventano responsabilità del chiamante e, nel caso di un oggetto restituito, la memoria può essere eliminata utilizzando l'eliminazione.

Manipolazione di stringhe binarie in C + +

Le stringhe di dati binari vengono dichiarate come oggetti della classe **ImqBinary** . Gli oggetti di questa classe possono essere copiati, confrontati e impostati utilizzando gli operatori C familiari. Viene fornito un codice di esempio.

Il seguente esempio di codice mostra le operazioni su una stringa binaria:

```
#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.
  ...
```

Manipolazione delle stringhe di carattere in C + +

I dati carattere vengono spesso restituiti in oggetti della classe **ImqString** che possono essere convertiti in **char *** utilizzando un operatore di conversione. La classe ImqString contiene i metodi per assistere nell'elaborazione delle stringhe di caratteri.

Quando i dati carattere vengono accettati o restituiti utilizzando i metodi MQI C + +, i dati carattere terminano sempre con valore null e possono essere di qualsiasi lunghezza. Tuttavia, alcuni limiti sono imposti da WebSphere MQ che potrebbe causare il troncamento delle informazioni. Per semplificare la gestione della memorizzazione, i dati di caratteri vengono spesso restituiti negli oggetti della classe **ImqString** . Questi oggetti possono essere assegnati a **char *** utilizzando l'operatore di conversione fornito e utilizzati per scopi di *sola lettura* in molte situazioni in cui è richiesto un **char *** .

Nota: Il risultato della conversione **char *** da un oggetto della classe **ImqString** potrebbe essere null.

Anche se le funzioni C possono essere utilizzate su **char ***, esistono metodi speciali della classe **ImqString** che sono preferibili; **operator length**() è l'equivalente di **strlen** e **storage**() indica la memoria assegnata per i dati carattere.

Stato iniziale degli oggetti in C + +

Tutti gli oggetti hanno uno stato iniziale coerente riflesso dai relativi attributi. I valori iniziali sono definiti nelle descrizioni delle classi.

Utilizzo di C da C++

Quando si utilizzano le funzioni C da un programma C + +, includere le intestazioni appropriate.

Il seguente esempio mostra `string.h` incluso in un programma C + +:

```
extern "C" {
#include <string.h>
}
```

Convenzioni notazionali C++

Questo esempio mostra come richiamare i metodi e dichiarare i parametri.

Questo esempio di codice utilizza i metodi e parametri**ImqBoolean ImqQueue**::**get**(**ImqMessage &** *msg*)

Dichiarare e utilizzare i parametri come segue:

```
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( );

  ...
}
```

Operazioni implicite in C++

Diverse operazioni possono verificarsi in modo implicito, *just in time*, per soddisfare le condizioni prerequisite per la corretta esecuzione di un metodo. Queste operazioni implicite sono connect, open, reopen, close e disconnect. È possibile controllare la connessione e aprire il comportamento implicito utilizzando gli attributi della classe.

Collega

Un oggetto gestore ImqQueueviene connesso automaticamente per qualsiasi metodo che risulti in una chiamata a MQI (consultare Riferimento incrociato C++ e MQI).

Aprire il

Un oggetto ImqObject viene aperto automaticamente per qualsiasi metodo che risulta in una chiamata MQGET, MQINQ, MQPUT o MQSET. Utilizzare il metodo **openFor** per specificare uno o più valori **open option** rilevanti.

Riapri

Un ImqObject viene riaperto automaticamente per qualsiasi metodo che risulta in una chiamata MQGET, MQINQ, MQPUT o MQSET, dove l'oggetto è già aperto, ma le **opzioni di apertura** esistenti non sono sufficienti per consentire la riuscita della chiamata MQI. L'oggetto viene temporaneamente chiuso utilizzando un valore **opzioni di chiusura** temporaneo di MQCO_NONE. Utilizzare il metodo **openFor** per aggiungere un**opzione di apertura**.

La riapertura può causare problemi in circostanze specifiche:

- Una coda dinamica temporanea viene eliminata quando viene chiusa e non può mai essere riaperta.

- Una coda aperta per l'input esclusivo (esplicitamente o per impostazione predefinita) potrebbe essere accessibile da altri utenti nella finestra di opportunità durante la chiusura e la riapertura.
- Una posizione del cursore di ricerca viene persa quando una coda viene chiusa. Questa situazione non impedisce la chiusura e la riapertura, ma impedisce l'uso successivo del cursore fino a quando MQGMO_BROWSE_FIRST non viene utilizzato nuovamente.
- Il contesto dell'ultimo messaggio richiamato viene perso quando una coda viene chiusa.

Se si verifica o è possibile prevedere una di queste circostanze, evitare le riaperture impostando esplicitamente le **opzioni di apertura** adeguate prima che un oggetto venga aperto (esplicitamente o implicitamente).

L'impostazione esplicita delle **opzioni di apertura** per situazioni di gestione delle code complesse consente di ottenere prestazioni migliori ed evita i problemi associati all'utilizzo della riapertura.

&Chiudi

Un ImqObject viene chiuso automaticamente in qualsiasi punto in cui lo stato dell'oggetto non è più valido, ad esempio se un riferimento di connessione ImqObject viene interrotto o se un oggetto ImqObject viene eliminato.

Disconnetti

Un gestore ImqQueueviene disconnesso automaticamente in qualsiasi momento in cui la connessione non è più praticabile, ad esempio se un riferimento di connessione ImqObject viene interrotto o se un oggetto gestore ImqQueueviene eliminato.

Stringhe binarie e di caratteri in C++

La classe ImqString incapsula il formato dati *char ** tradizionale. La classe ImqBinary incapsula la schiera di byte binari. Alcuni metodi che impostano i dati carattere potrebbero troncare i dati.

I metodi che impostano i dati carattere (**char ***) prendono sempre una copia dei dati, ma alcuni metodi potrebbero troncare la copia, poiché determinati limiti sono imposti da WebSphere MQ.

La classe ImqString (consultare ImqString C++ class) incapsula il **char *** tradizionale e fornisce supporto per:

- Confronto
- Concatenamento
- Copia in corso
- Conversione da intero a testo e da testo a intero
- Estrazione token (parola)
- Traduzione in maiuscolo

La classe ImqBinary (vedere ImqBinary classe C++) incapsula array di byte binari di dimensione arbitraria. In particolare, viene utilizzato per contenere i seguenti attributi:

- **token di account** (MQBYTE32)
- **tag di connessione** (MQBYTE128)
- **ID correlazione** (MQBYTE24)
- **token funzione** (MQBYTE8)
- **ID gruppo** (MQBYTE24)
- **ID istanza** (MQBYTE24)
- **ID messaggio** (MQBYTE24)
- **token messaggio** (MQBYTE16)
- **ID istanza transazione** (MQBYTE16)

Dove questi attributi appartengono agli oggetti delle seguenti classi:

- ImqCICSBridgeIntestazione (consultare ImqCICSBridgeclasse C++ Intestazione)
- ImqGetMessageOptions (vedere ImqGetMessageOptions classe C++)
- Intestazione ImqIMSBridge(consultare ImqIMSBridgeclasse C++ intestazione)
- ImqMessageTracker (consultare ImqMessageClasse C++ Tracker)
- ImqQueueManager (consultare ImqQueueClasse C++ Manager)
- ImqReferenceIntestazione (consultare ImqReferenceclasse C++ intestazione)
- ImqWorkIntestazione (vedere ImqWorkclasse C++ intestazione)

La classe ImqBinary fornisce anche supporto per il confronto e la copia.

Funzioni non supportate in C++

Le classi e metodi C++ di WebSphere MQ sono indipendenti dalla piattaforma WebSphere MQ . Potrebbero quindi offrire alcune funzioni che non sono supportate su determinate piattaforme.

Se si tenta di utilizzare una funzione su una piattaforma su cui non è supportato, la funzione viene rilevata da WebSphere MQ ma non dai bind di linguaggio C++. WebSphere MQ riporta l'errore al programma, come qualsiasi altro errore MQI.

Messaggistica in C++

Questa raccolta di argomenti descrive in dettaglio come preparare, leggere e scrivere i messaggi in C + +.

Preparazione dei dati dei messaggi in C + +

I dati del messaggio vengono preparati in un buffer, che può essere fornito dal sistema o dall'applicazione. Ci sono vantaggi in entrambi i metodi. Vengono forniti esempi di utilizzo di un buffer.

Quando si invia un messaggio, i dati del messaggio vengono preparati per la prima volta in un buffer gestito da un oggetto ImqCache (consultare ImqCache classe C++). Un buffer viene associato (per eredità) a ciascun oggetto ImqMessage (consultare ImqMessage C++ class): può essere fornito dall'applicazione (utilizzando il metodo **useEmptyBuffer** o **useFullBuffer**) o automaticamente dal sistema. Il vantaggio dell'applicazione che fornisce il buffer dei messaggi è che in molti casi non è necessaria alcuna copia dei dati, poiché l'applicazione può utilizzare direttamente le aree di dati preparate. Lo svantaggio è che il buffer fornito è di lunghezza fissa.

Il buffer può essere riutilizzato e il numero di byte trasmessi può essere modificato ogni volta, utilizzando il metodo **setMessageLength** prima della trasmissione.

Quando vengono forniti automaticamente dal sistema, il numero di byte disponibili viene gestito dal sistema e i dati possono essere copiati nel buffer dei messaggi utilizzando, ad esempio, il metodo ImqCache **write** o il metodo ImqMessage **writeItem** . Il buffer dei messaggi cresce in base alle esigenze. Quando il buffer cresce, non si verifica alcuna perdita di dati precedentemente scritti. Un messaggio di grandi dimensioni o a più parti può essere scritto in parti sequenziali.

I seguenti esempi mostrano gli invii di messaggi semplificati.

1. Utilizzare i dati preparati in un buffer fornito dall'utente

```
char szBuffer[ ] = "Hello world" ;

msg.useFullBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
```

2. Utilizzare i dati preparati in un buffer fornito dall'utente, dove la dimensione del buffer supera la dimensione dei dati

```
char szBuffer[ 24 ] = "Hello world" ;
```

```
msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.setMessageLength( 12 );
```

3. Copiare i dati in un buffer fornito dall'utente

```
char szBuffer[ 12 ];

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

4. Copia i dati in un buffer fornito dal sistema

```
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

5. Copiare i dati in un buffer fornito dal sistema utilizzando oggetti (oggetti che impostano il formato del messaggio e il contenuto)

```
ImqString strText( "Hello world" );

msg.writeItem( strText );
```

Lettura dei messaggi in C++

Un buffer può essere fornito dall'applicazione o dal sistema. È possibile accedere ai dati direttamente dal buffer o leggerli in modo sequenziale. Esiste una classe equivalente a ciascun tipo di messaggio. Viene fornito un codice di esempio.

Quando si ricevono i dati, l'applicazione o il sistema possono fornire un buffer di messaggi adatto. Lo stesso buffer può essere utilizzato sia per la trasmissione multipla che per la ricezione multipla per un particolare oggetto ImqMessage . Se il buffer di messaggi viene fornito automaticamente, aumenta in modo da contenere la lunghezza dei dati ricevuti. Tuttavia, un buffer di messaggi fornito dall'applicazione potrebbe non essere abbastanza grande da contenere i dati ricevuti. Quindi, potrebbe verificarsi un troncamento o un errore, a seconda delle opzioni utilizzate per la ricezione del messaggio.

È possibile accedere ai dati in entrata direttamente dal buffer dei messaggi, nel qual caso la lunghezza dei dati indica la quantità totale di dati in entrata. In alternativa, i dati in entrata possono essere letti sequenzialmente dal buffer di messaggi. In questo caso, il puntatore dati si indirizza al byte successivo di dati in entrata e il puntatore dati e la lunghezza dati vengono aggiornati ogni volta che i dati vengono letti.

Gli *elementi* sono parti di un messaggio, tutte nell'area utente del buffer di messaggi, che devono essere elaborate in modo sequenziale e separato. A parte i dati utente regolari, un elemento potrebbe essere un'intestazione dead-letter o un messaggio trigger. Gli elementi sono sempre associati ai formati dei messaggi; i formati dei messaggi ***non*** sono sempre associati agli elementi.

Esiste una classe di oggetti per ogni elemento che corrisponde ad un formato di messaggio riconoscibile WebSphere MQ . Ce n' è uno per un'intestazione di lettera non recapitabile e uno per un messaggio trigger. Non esiste alcuna classe di oggetto per i dati utente. Vale a dire, una volta esauriti i formati riconoscibili, l'elaborazione del resto viene lasciata al programma applicativo. Le classi per i dati utente possono essere scritte specializzando la classe ImqItem .

Il seguente esempio mostra una ricevuta di messaggio che tiene conto di un numero di potenziali elementi che possono precedere i dati dell'utente, in una situazione immaginaria. I dati utente non elementi sono definiti come qualsiasi cosa che si verifica dopo gli elementi che possono essere

identificati. Un buffer automatico (predefinito) viene utilizzato per contenere una quantità arbitraria di dati del messaggio.

```
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.                      */
      ...

    }

  }
}
```

In questo esempio, FMT_USERCLASS è una costante che rappresenta il nome formato di 8 caratteri associato a un oggetto della classe `UserClass`ed è definito dall'applicazione.

`UserClass` deriva dalla classe ImqItem (vedere ImqItem C++ class) e implementa i metodi **copyOut** e **pasteIn** virtuali di tale classe.

I successivi due esempi mostrano il codice dalla classe ImqDeadLetterHeader (consultare ImqDeadLetterHeader C++ class). Il primo esempio mostra un messaggio incapsulato personalizzato - codice di*scrittura* .

```
// 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 ;
}
```

Il secondo esempio mostra il messaggio incapsulato personalizzato - codice di*lettura* .

```
// 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 ;
}
```

Con un buffer automatico, la memoria buffer è *volatile*. Vale a dire, i dati del buffer potrebbero essere conservati in un'altra ubicazione fisica dopo ogni richiamo del metodo **get** . Pertanto, ogni volta che si fa riferimento ai dati del buffer, utilizzare i metodi **bufferPointer** o **dataPointer** per accedere ai dati del messaggio.

È possibile che si desideri che un programma riservi un'area fissa per la ricezione dei dati del messaggio. In questo caso, richiamare il metodo **useEmptyBuffer** prima di utilizzare il metodo **get** .

L'uso di un'area fissa e non automatica limita i messaggi ad una dimensione massima, pertanto è importante considerare l'opzione MQGMO_ACCEPT_TRUNCATED_MSG dell'oggettoMessageOptions ImqGet. Se questa opzione non viene specificata (impostazione predefinita), è possibile che sia previsto il codice motivo MQRC_TRUNCATED_MSG_FAILED. Se questa opzione viene specificata, è possibile che il codice motivo MQRC_TRUNCATED_MSG_ACCEPTED sia previsto in base alla progettazione dell'applicazione.

L'esempio successivo mostra come è possibile utilizzare un'area fissa di memoria per ricevere i messaggi:

```
char * pszBuffer = new char[ 100 ];

msg.useEmptyBuffer( pszBuffer, 100 );
gmo.setOptions( MQGMO_ACCEPT_TRUNCATED_MSG );
queue.get( msg, gmo );

delete [ ] pszBuffer ;
```

In questo frammento di codice, il buffer può sempre essere indirizzato direttamente, con *pszBuffer*, invece di utilizzare il metodo **bufferPointer** . Tuttavia, è preferibile utilizzare il metodo **dataPointer** per l'accesso per scopi generici. L'applicazione (non l'oggetto classe ImqCache) deve eliminare un buffer definito dall'utente (non automatico).

Attenzione: la specifica di un puntatore null e di una lunghezza zero con **useEmptyBuffer** non denomina un buffer a lunghezza fissa di lunghezza zero come potrebbe essere previsto. Questa combinazione viene interpretata come una richiesta di ignorare qualsiasi buffer definito dall'utente precedente e di tornare invece all'utilizzo di un buffer automatico.

Scrittura di un messaggio nella coda di messaggi non recapitabili in C + +

Codice programma di esempio per la scrittura di un messaggio nella coda di messaggi non recapitabili.

Un caso tipico di un messaggio multipart è quello che contiene un'intestazione di lettera non recapitabile. I dati da un messaggio che non può essere elaborato vengono accodati all'intestazione del messaggio non instradabile.

```
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 );
```

Scrittura di un messaggio nel bridge IMS in C++

Codice di programma di esempio per la scrittura di un messaggio sul bridge IMS .

I messaggi inviati al bridge WebSphere MQ-IMS potrebbero utilizzare un'intestazione speciale. L'intestazione del bridge IMS ha come prefisso i dati dei messaggi regolari.

```
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 );
```

Scrittura di un messaggio sul bridge CICS in C + +

Codice programma di esempio per la scrittura di un messaggio sul bridge CICS .

I messaggi inviati a WebSphere MQ per z/OS utilizzando il bridge CICS richiedono un'intestazione speciale. L'intestazione del bridge CICS ha come prefisso i dati dei messaggi regolari.

```
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 );
```

Scrittura di un messaggio con un'intestazione di lavoro in C++

Codice di esempio del programma per la scrittura di un messaggio destinato a una coda gestita da z/OS Workload Manager.

I messaggi inviati a WebSphere MQ per z/OS, che sono destinati a una coda gestita da z/OS Workload Manager, richiedono un'intestazione speciale. L'intestazione del lavoro ha come prefisso i dati dei messaggi regolari.

```
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 );
```

Creazione di programmi C+ + WebSphere MQ

Viene elencato l'URL dei compilatori supportati, insieme ai comandi da utilizzare per compilare, collegare ed eseguire programmi ed esempi C + + su piattaforme WebSphere MQ .

I compilatori per ogni versione e piattaforma supportata di WebSphere MQ sono elencati nella pagina dei requisiti di sistema di WebSphere MQ all'indirizzo Requisiti di sistema per IBM WebSphere MQ.

Il comando necessario per compilare e collegare il proprio programma C++ WebSphere MQ dipende dall'installazione e dai requisiti. Gli esempi che seguono mostrano i tipici comandi di compilazione e collegamento per alcuni compilatori che utilizzano l'installazione predefinita di WebSphere MQ su diverse piattaforme.

Creazione di programmi C++ su AIX

Crea programmi C++ WebSphere MQ su AIX utilizzando il compilatore XL C Enterprise Edition .

Client

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Applicazione senza thread a 32 bit

```
xlC -o imqsputc_32 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqc23ia -limqb23ia -lmqic
```

Applicazione con thread a 32 bit

```
xlC_r -o imqsputc_32_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqc23ia_r -limqb23ia_r -lmqic_r
```

Applicazione senza thread a 64 bit

```
xlC -q64 -o imqsputc_64 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqc23ia -limqb23ia -lmqic
```

Applicazione con thread a 64 bit

```
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
```

Server

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Applicazione senza thread a 32 bit

```
xlC -o imqsput_32 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqs23ia -limqb23ia -lmqm
```

Applicazione con thread a 32 bit

```
xlC_r -o imqsput_32_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqs23ia_r -limqb23ia_r -lmqm_r
```

Applicazione senza thread a 64 bit

```
xlC -q64 -o imqsput_64 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqs23ia -limqb23ia -lmqm
```

Applicazione con thread a 64 bit

```
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
```

Creazione di programmi C+ + su HP-UX

Creare programmi WebSphere MQ C++ su HP-UX utilizzando i compilatori aC+ + o aCC .

Su HP-UX Itanium, WebSphere MQ supporta solo il runtime Standard. Utilizzare il compilatore aCC .

- libimqi23bh.sl fornisce la classe C++ WebSphere MQ per il runtime Standard.
- Per compatibilità con le versioni precedenti, viene fornito un link simbolico da libimqi23ah.sl a libimqi23bh.sl.

IA64 (IPF)

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Client: IA64 (IPF)

Applicazione senza thread a 32 bit

```
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
```

Applicazione con thread a 32 bit

```
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
```

Applicazione senza thread a 64 bit

```
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
```

Applicazione con thread a 64 bit

```
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
```

Server: IA64 (IPF)

Applicazione senza thread a 32 bit

```
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
```

Applicazione con thread a 32 bit

```
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
```

Applicazione senza thread a 64 bit

```
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
```

Applicazione con thread a 64 bit

```
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
```

Creazione di programmi C++ su Linux

Creare programmi C++ WebSphere MQ su Linux utilizzando il compilatore GNU g + +.

Sistema p

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Client: System p

Applicazione senza thread a 32 bit

```
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
```

Applicazione con thread a 32 bit

```
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
```

Applicazione senza thread a 64 bit

```
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
```

Applicazione con thread a 64 bit

```
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
```

Server: System p

Applicazione senza thread a 32 bit

```
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
```

Applicazione con thread a 32 bit

```
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
```

Applicazione senza thread a 64 bit

```
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
```

Applicazione con thread a 64 bit

```
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 rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Client: System z

Applicazione senza thread a 32 bit

```
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
```

Applicazione con thread a 32 bit

```
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
```

Applicazione senza thread a 64 bit

```
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
```

Applicazione con thread a 64 bit

```
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
```

Server: System z

Applicazione senza thread a 32 bit

```
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
```

Applicazione con thread a 32 bit

```
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
```

Applicazione senza thread a 64 bit

```
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
```

Applicazione con thread a 64 bit

```
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 bit)

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Client: System x (32 bit)

Applicazione senza thread a 32 bit

```
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
```

Applicazione con thread a 32 bit

```
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
```

Applicazione senza thread a 64 bit

```
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
```

Applicazione con thread a 64 bit

```
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
```

Server: System x (32 bit)

Applicazione senza thread a 32 bit

```
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
```

Applicazione con thread a 32 bit

```
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
```

Applicazione senza thread a 64 bit

```
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
```

Applicazione con thread a 64 bit

```
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
```

Creazione di programmi C++ su Solaris

Crea programmi C++ WebSphere MQ su Solaris utilizzando il compilatore Sun ONE.

SPARC

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Client: SPARC

Applicazione a 32 bit

```
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
```

Applicazione a 64 bit

```
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
```

Server: SPARC

Applicazione a 32 bit

```
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
```

Applicazione a 64 bit

```
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 rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Client: x86-64

Applicazione a 32 bit

```
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
```

Applicazione a 64 bit

```
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
```

Server: x86-64

Applicazione a 32 bit

```
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
```

Applicazione a 64 bit

```
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
```

Creazione di programmi C+ + su Windows

Creare programmi C++ di WebSphere MQ su Windows utilizzando il compilatore Microsoft Visual Studio C++.

I file della libreria (.lib) e i file dll da utilizzare con applicazioni a 32 bit sono installati in *MQ_INSTALLATION_PATH*/Tools/Lib, i file da utilizzare con applicazioni a 64 bit sono installati in *MQ_INSTALLATION_PATH*/Tools/Lib64. *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Client

```
cl -MD imqsput.cpp /Feimqsputc.exe imqb23vn.lib imqc23vn.lib
```

Server

```
 cl -MD imqsput.cpp /Feimqsput.exe imqb23vn.lib imqs23vn.lib
```

Utilizzo di classi WebSphere MQ per Java

Le classi WebSphere MQ per Java consentono di utilizzare WebSphere MQ in un ambiente Java. Un'applicazione Java può utilizzare le classi WebSphere MQ per Java o WebSphere MQ per JMS per accedere alle risorse WebSphere MQ .

Le classi WebSphere MQ per Java consentono a un'applicazione Java di:

- Connettersi a WebSphere MQ come client WebSphere MQ
- Connetti direttamente a un gestore code WebSphere MQ

WebSphere Le classi MQ per Java incapsulano l'API MQI (Message Queue Interface), l'API nativa WebSphere MQ .

WebSphere MQ classes per Java utilizza un modello oggetto simile alle interfacce C + + e .NET per WebSphere MQ.

Perché utilizzare le classi WebSphere MQ per Java?

Se i seguenti punti sono significativi nell'installazione, considerare l'utilizzo delle classi WebSphere MQ per Java:

- WebSphere Le classi MQ per Java incapsulano l'API MQI (Message Queue Interface), l'API nativa WebSphere MQ .
  - Se si ha familiarità con l'utilizzo di MQI nei linguaggi procedurali, è possibile trasferire questa conoscenza all'ambiente Java.

  - È possibile sfruttare l'intera gamma di funzioni di WebSphere MQ, oltre a quelle disponibili tramite JMS.
- WebSphere MQ classes per Java utilizza un modello oggetto simile alle interfacce C + + e .NET per WebSphere MQ. Se si ha dimestichezza con queste interfacce, è possibile trasferire questa conoscenza all'ambiente Java.

Nota: La riconnessione automatica del client non è supportata dalle classi WebSphere MQ per Java.

Introduzione alle classi WebSphere MQ per Java

Questa raccolta di argomenti fornisce una panoramica delle classi WebSphere MQ per Java e dei loro utilizzi.

Cosa sono le classi WebSphere MQ per Java?

Le classi WebSphere MQ per Java consentono di utilizzare WebSphere MQ in un ambiente Java.

Le classi WebSphere MQ per Java consentono a un'applicazione Java di:

- Connettersi a WebSphere MQ come client WebSphere MQ
- Connetti direttamente a un gestore code WebSphere MQ

WebSphere Le classi MQ per Java incapsulano l'API MQI (Message Queue Interface), l'API nativa WebSphere MQ .

WebSphere MQ classes per Java utilizza un modello oggetto simile alle interfacce C + + e .NET per WebSphere MQ.

Perché utilizzare le classi WebSphere MQ per Java?

Un'applicazione Java può utilizzare le classi WebSphere MQ per Java o WebSphere MQ per JMS per accedere alle risorse WebSphere MQ . Esistono numerosi vantaggi nell'utilizzo delle classi WebSphere MQ per Java.

Se i seguenti punti sono significativi nell'installazione, considerare l'utilizzo delle classi MQ Websphere per Java:

- WebSphere Le classi MQ per Java incapsulano l'API MQI (Message Queue Interface), l'API nativa WebSphere MQ .
  - Se si ha familiarità con l'utilizzo di MQI nei linguaggi procedurali, è possibile trasferire questa conoscenza all'ambiente Java.
  - È possibile sfruttare l'intera gamma di funzioni di WebSphere MQ, oltre a quelle disponibili tramite JMS.
- WebSphere MQ classes per Java utilizza un modello oggetto simile alle interfacce C + + e .NET per WebSphere MQ. Se si ha dimestichezza con queste interfacce, è possibile trasferire questa conoscenza all'ambiente Java.

Opzioni di connessione per WebSphere MQ classes for Java

Le classi WebSphere MQ per Java possono connettersi in modalità client o bind.

Le opzioni programmabili consentono alle classi WebSphere MQ per Java di connettersi a WebSphere MQ in uno dei seguenti modi:

- Come client WebSphere MQ MQI che utilizza TCP/IP (Transmission Control Protocol/Internet Protocol)
- In modalità bind, connessione diretta a WebSphere MQ utilizzando JNI (Java Native Interface)

I client non possono essere eseguiti su z/OS, ma i client su altre piattaforme possono connettersi a un gestore code WebSphere MQ per z/OS se è installato Client Attach Facility.

Le seguenti sezioni descrivono le opzioni di connessione in modalità client e in modalità bind in modo più dettagliato.

Connessione client

Per connettersi a un gestore code in modalità client, è possibile eseguire le classi WebSphere MQ per l'applicazione Java sullo stesso sistema su cui è in esecuzione il gestore code o su un sistema differente. In ogni caso, WebSphere MQ classes for Java si connette al gestore code su TCP/IP.

Le classi WebSphere MQ per l'applicazione Java possono connettersi a qualsiasi gestore code supportato utilizzando la modalità client.

Per ulteriori informazioni su come scrivere le applicazioni per utilizzare le connessioni in modalità client, consultare “Classi WebSphere MQ per le modalità di connessione Java” a pagina 670.

Connessione binding

Quando utilizzato in modalità bind, WebSphere MQ classes for Java utilizza JNI (Java Native Interface) per richiamare direttamente l'API del gestore code esistente, piuttosto che comunicare attraverso una rete. Nella maggior parte degli ambienti, la connessione in modalità bind fornisce prestazioni migliori per le classi WebSphere MQ per le applicazioni Java rispetto alla connessione in modalità client, evitando il costo della comunicazione TCP/IP.

Le applicazioni che utilizzano le classi di WebSphere MQ per Java per connettersi in modalità di bind devono essere eseguite sullo stesso sistema del gestore code a cui si stanno collegando.

Java Runtime Environment, utilizzato per eseguire le classi WebSphere MQ per l'applicazione Java, deve essere configurato per caricare le classi WebSphere MQ per le librerie Java; per ulteriori informazioni, consultare WebSphere MQ classes per le librerie Java .

Per ulteriori informazioni su come scrivere le applicazioni per utilizzare le connessioni in modalità bind, consultare “Classi WebSphere MQ per le modalità di connessione Java” a pagina 670.

Prerequisiti per le classi WebSphere MQ per Java

Per utilizzare le classi WebSphere MQ per Java, sono necessari alcuni altri prodotti software.

Per le informazioni più recenti sui prerequisiti per WebSphere MQ classes for Java, consultare il file README WebSphere MQ .

Per sviluppare classi WebSphere MQ per applicazioni Java, è necessario un JDK (Java Development Kit). I dettagli relativi ai JDK supportati con il proprio sistema operativo sono disponibili nella pagina dei requisiti di sistema di WebSphere MQ all'indirizzo Requisiti di sistema per IBM WebSphere MQ.

Per eseguire le classi di WebSphere MQ per le applicazioni Java, sono necessari i seguenti componenti software:

- Un gestore code WebSphere MQ , per applicazioni che si connettono a un gestore code
- Un JRE (Java Runtime Environment) per ogni sistema su cui si eseguono le applicazioni. Un JRE adatto viene fornito con WebSphere MQ.

Se si richiedono connessioni SSL per utilizzare moduli crittografici certificati FIPS 140-2, è necessario il provider IBM Java JSSE FIPS (IBMJSSEFIPS). Ogni IBM JDK e JRE alla versione 1.4.2 o successiva contiene IBMJSSEFIPS.

È possibile utilizzare gli indirizzi Internet Protocol Versione 6 (IPv6) nelle classi WebSphere MQ per le applicazioni Java se IPv6 è supportato dalla JVM (Java virtual machine) e dall'implementazione TCP/IP sul proprio sistema operativo.

Installazione e configurazione di classi WebSphere MQ per Java

Questa sezione descrive le directory e i file che vengono creati quando si installano le classi WebSphere MQ per Java e indica come configurare le classi di WebSphere MQ per Java dopo l'installazione.

Elementi installati per le classi WebSphere MQ per Java

L'ultima versione di WebSphere MQ classes for Java è installata con WebSphere MQ. Potrebbe essere necessario sovrascrivere le opzioni di installazione predefinite per assicurarsi che ciò avvenga.

Per ulteriori informazioni sull'installazione di WebSphere MQ , consultare:

Installazione di un server WebSphere MQ
Installazione di un client IBM WebSphere MQ

Le classi WebSphere MQ per Java sono contenute nei file JAR (Java archive) com.ibm.mq.jare com.ibm.mq.jmqi.jar.

Il supporto per le intestazioni di messaggi standard, come PCF (Programmable Command Format), è contenuto nel file JAR com.ibm.mq.headers.jar.

Il supporto per PCF (Programmable Command Format) è contenuto nel file JAR com.ibm.mq.pcf.jar.

Installazione e aggiornamento delle classi WebSphere MQ per file JAR Java

L'unico modo supportato per ottenere le classi WebSphere MQ per i file JAR Java su un sistema, è installare il prodotto WebSphere MQ , il client WebSphere MQ MQI SupportPaco utilizzare uno strumento di gestione software come Apache Maven, per ulteriori informazioni consultare “IBM WebSphere MQ classes for Java e strumenti di gestione software” a pagina 665.

Non spostare o copiare le classi WebSphere MQ per i file JAR Java da altre macchine, a meno che non si stia utilizzando uno strumento di gestione software.

- I fix pack non possono essere applicati a una "installazione" in cui i file JAR sono stati copiati da un'altra macchina e rendono molto più difficile garantire che tutti i file JAR siano mantenuti in linea tra loro e a livelli compatibili.
- La copia dei file JAR delle classi WebSphere MQ per JMS tra le macchine può anche causare più copie dei file che si trovano sulla stessa macchina, il che può causare problemi di manutenzione del codice e di debug.

Non includere le classi WebSphere MQ per file JAR Java negli archivi dell'applicazione.

- Gli aggiornamenti alle classi WebSphere MQ per Java non possono essere applicati utilizzando un fix pack di WebSphere MQ .
- Non è possibile per il supporto IBM determinare facilmente la versione delle classi WebSphere MQ per Java utilizzate dall'applicazione.
- I problemi possono verificarsi se più applicazioni in esecuzione nello stesso JRE (Java Runtime Environment) includono versioni differenti delle classi WebSphere MQ per Java, poiché più versioni delle classi WebSphere MQ per Java vengono caricate contemporaneamente in JRE (Java Runtime Environment).
- Se un'applicazione utilizza il trasporto BINDINGS per connettersi ad un gestore code, qualsiasi aggiornamento principale al gestore code richiede anche che l'applicazione venga aggiornata in modo da includere il livello corrispondente delle classi WebSphere MQ per Java.

  Ad esempio, se un gestore code viene aggiornato al livello WebSphere MQ Versione 7.1 , anche tutte le applicazioni che si connettono al gestore code utilizzando il trasporto BINDINGS devono essere aggiornate per includere le classi WebSphere MQ Versione 7.1 per Java.

La seguente libreria Java viene distribuita con le classi WebSphere MQ per Java:

- connector.jar (Versione 1.0)

L'applicazione di esempio denominata Postcard si trova nel file JAR com.ibm.mq.postcard.jar.

Lo strumento Javadoc è stato utilizzato per generare le pagine HTML contenenti le specifiche delle WebSphere MQ classes per Java e WebSphere MQ classes per le API JMS. Le pagine HTML si trovano nella sottodirectory doc della directory di installazione di WebSphere MQ classes per JMS. Su sistemi UNIX, Linuxe Windows , la sottodirectory doc contiene le singole pagine HTML.

Al termine dell'installazione, i file e gli esempi vengono installati nelle ubicazioni mostrate in “Directory di installazione per classi WebSphere MQ per Java” a pagina 658.

Dopo l'installazione, su qualsiasi piattaforma diversa da Windows, è necessario aggiornare le proprie variabili di ambiente come descritto in “Variabili di ambiente relative alle classi WebSphere MQ per Java” a pagina 658.

Directory di installazione per classi WebSphere MQ per Java

Le classi WebSphere MQ per file Java sono installate in ubicazioni differenti in base alla piattaforma.

Tabella 82 a pagina 658 mostra dove sono installati i file WebSphere MQ classes per Java.

Tabella 82. WebSphere MQ classes per le directory di installazione Java

Piattaforma	**Cartella**
AIX	*MQ_INSTALLATION_PATH*/java/lib
HP-UX, Linuxe Solaris	*MQ_INSTALLATION_PATH*/java/lib
Finestre	*MQ_INSTALLATION_PATH*\java\lib
MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .	

Alcune applicazioni di esempio, come IVP (Installation Verification Program), vengono fornite con WebSphere MQ. Tabella 83 a pagina 658 mostra dove sono installate le applicazioni di esempio. Le classi WebSphere MQ per gli esempi Java si trovano in una sottodirectory denominata `wmqjava`. Gli esempi PCF si trovano in una sottodirectory denominata `pcf`.

Tabella 83. Directory di esempi

Piattaforma	**Cartella**
AIX	*MQ_INSTALLATION_PATH*/samp/wmqjava/
HP-UX, Linuxe Solaris	*MQ_INSTALLATION_PATH*/samp/wmqjava/
Finestre	*MQ_INSTALLATION_PATH*\tools\wmqjava\
MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .	

Variabili di ambiente relative alle classi WebSphere MQ per Java

Se si desidera eseguire le classi WebSphere MQ per le applicazioni Java, i relativi percorsi classe devono includere le classi WebSphere MQ per Java e le directory degli esempi.

Per le classi WebSphere MQ per le applicazioni Java da eseguire, il loro percorso classe deve includere la directory WebSphere MQ appropriata per le classi Java. Per eseguire le applicazioni di esempio, il percorso classe deve includere anche le directory degli esempi appropriate. Queste informazioni possono essere fornite nel comando di richiamo Java o nella variabile di ambiente CLASSPATH.

La Tabella 84 a pagina 658 mostra l'impostazione CLASSPATH appropriata da utilizzare su ciascuna piattaforma per eseguire le classi WebSphere MQ per le applicazioni Java, incluse le applicazioni di esempio.

Tabella 84. Impostazione CLASSPATH per eseguire classi WebSphere MQ per applicazioni Java, incluse le classi WebSphere MQ per applicazioni di esempio Java

Piattaforma	**impostazione CLASSPATH**
AIX	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:

Tabella 84. Impostazione CLASSPATH per eseguire classi WebSphere MQ per applicazioni Java, incluse le classi WebSphere MQ per applicazioni di esempio Java (Continua)	
Piattaforma	**impostazione CLASSPATH**
HP-UX, Linuxe Solaris	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
Finestre	CLASSPATH=*MQ_INSTALLATION_PATH*\Java\lib\com.ibm.mq.jar; *MQ_INSTALLATION_PATH*\tools\wmqjava\samples;
MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .	

Se si esegue la compilazione utilizzando l'opzione -Xlint, potrebbe essere visualizzato un messaggio che indica che com.ibm.mq.ese.jar non è presente. È possibile ignorare l'avvertenza. Questo file è presente solo se è stato installato IBM WebSphere MQ Advanced Message Security.

Gli script forniti con le classi WebSphere MQ per Java utilizzano le seguenti variabili di ambiente:

MQ_JAVA_DATA_PATH
Questa variabile di ambiente specifica la directory per l'output di log e traccia.

PERCORSO_INSTALL_JAVA_MQ_
Questa variabile di ambiente specifica la directory in cui sono installate le classi WebSphere MQ per Java, come mostrato in WebSphere MQ per le directory di installazione Java.

MQ_JAVA_LIB_PATH
Questa variabile di ambiente specifica la directory in cui sono memorizzate le classi WebSphere MQ per le librerie Java, come mostrato in L'ubicazione delle classi WebSphere MQ per le librerie Java per ciascuna piattaforma. Alcuni script forniti con WebSphere MQ classes for Java, come IVTRun, utilizzano questa variabile di ambiente.

In Windows, tutte le variabili di ambiente vengono impostate automaticamente durante l'installazione. Su qualsiasi altra piattaforma, è necessario impostarli da soli.Su un sistema Unix , è possibile utilizzare lo script **setjmsenv** (se si utilizza una JVM a 32 bit) o **setjmsenv64** (se si utilizza una JVM a 64 bit) per impostare le variabili di ambiente. Su AIX, HP-UX, Linuxe Solaris, questi script si trova nella directory *MQ_INSTALLATION_PATH*/java/bin .

Le classi IBM WebSphere MQ per le librerie Java

L'ubicazione delle classi IBM WebSphere MQ per le librerie Java varia in base alla piattaforma. Specificare questa ubicazione quando si avvia una applicazione.

Per specificare l'ubicazione delle librerie JNI (Java Native Interface), avviare l'applicazione utilizzando un comando **java** con il seguente formato:

```
java -Djava.library.path=library_path application_name
```

dove *library_path* è il percorso delle classi WebSphere MQ per le librerie Java, che includono le librerie JNI. Tabella 85 a pagina 659 mostra l'ubicazione delle classi WebSphere MQ per le librerie Java per ogni piattaforma.

Tabella 85. L'ubicazione delle classi WebSphere MQ per le librerie Java per ciascuna piattaforma	
Piattaforma	**Directory contenente le classi WebSphere MQ per le librerie Java**
AIX	*MQ_INSTALLATION_PATH*/java/lib (librerie a 32 bit) *MQ_INSTALLATION_PATH*/java/lib64 (librerie a 64 bit)

Tabella 85. L'ubicazione delle classi WebSphere MQ per le librerie Java per ciascuna piattaforma (Continua)

Piattaforma	**Directory contenente le classi WebSphere MQ per le librerie Java**
HP-UX Linux (POWER, x86-64 e zSeries s390x piattaforme) Solaris (piattaformex86-64 e SPARC)	*MQ_INSTALLATION_PATH*/java/lib (librerie a 32 bit) *MQ_INSTALLATION_PATH*/java/lib64 (librerie a 64 bit)
Linux (piattaforma x86)	*MQ_INSTALLATION_PATH*/java/lib
Finestre	*MQ_INSTALLATION_PATH*\Java\lib (librerie a 32 bit) *MQ_INSTALLATION_PATH*\Java\lib64 (librerie a 64 bit)
MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .	

Nota:

1. Su AIX, HP-UX, Linux (Power platform) o Solaris, utilizzare le librerie a 32 bit o a 64 bit. Utilizzare le librerie a 64 bit solo se si sta eseguendo l'applicazione in una JVM (Java virtual machine) a 64 bit su una piattaforma a 64 bit. Altrimenti, utilizzare le librerie a 32 bit.
2. In Windows, è possibile utilizzare la variabile di ambiente PATH per specificare l'ubicazione delle librerie WebSphere MQ per le librerie Java invece di specificarne l'ubicazione nel comando **java** .
3. Per utilizzare le classi WebSphere MQ per Java in modalità bind su IBM i, verificare che la libreria QMQMJAVA sia presente nell'elenco librerie.

Attività correlate

Utilizzo di classi WebSphere MQ per Java

Supporto per OSGi su IBM WebSphere MQ classes for Java

OSGi fornisce un framework che supporta la distribuzione delle applicazioni come bundle. Un bundle OSGi viene fornito come parte di IBM WebSphere MQ classes for Java .

OSGi fornisce un frameworkJava generico, sicuro e gestito, che supporta la distribuzione di applicazioni fornite sotto forma di bundle. I dispositivi conformi a OSGi possono scaricare e installare i bundle e rimuoverli quando non sono più necessari. Il framework gestisce l'installazione e l'aggiornamento dei bundle in modo dinamico e scalabile.

Il IBM WebSphere MQ classes for Java. include il seguente bundle OSGi.

com.ibm.mq.osgi.java_< numero versione> .jar
I file JAR per consentire alle applicazioni di utilizzare IBM WebSphere MQ classes for Java.

dove < numero di versione> è il numero di versione di WebSphere MQ che è stato installato.

Il bundle viene installato nella sottodirectory `java/lib/OSGi` dell'installazione di IBM WebSphere MQ o nella cartella `java\lib\OSGi` su Windows.

Nove altri bundle vengono installati anche nella sottodirectory `java/lib/OSGi` dell'installazione di IBM WebSphere MQ o nella cartella `java\lib\OSGi` su Windows. Questi bundle fanno parte di IBM WebSphere MQ classes for JMS e non devono essere caricati in un ambiente di runtime OSGi che ha il bundle IBM WebSphere MQ classes for Java caricato. Se il bundle OSGi IBM WebSphere MQ classes for

Java è caricato in un ambiente di runtime OSGi che ha anche i bundle IBM WebSphere MQ classes for JMS caricati, si verificano errori quali:

```
java.lang.ClassCastException: com.ibm.mq.MQException incompatible with com.ibm.mq.MQException
```

si verificano quando vengono eseguite le applicazioni che utilizzano il bundle IBM WebSphere MQ classes for Java o i bundle IBM WebSphere MQ classes for JMS .

Il bundle OSGi per IBM WebSphere MQ classes for Java è stato scritto nella specifica OSGi Release 4; non funziona in un ambiente OSGi Release 3.

È necessario impostare correttamente il percorso di sistema o il percorso della libreria in modo che l'ambiente di runtime OSGi possa trovare i file DLL o le librerie condivise richiesti.

Se si utilizza il bundle OSGi per IBM WebSphere MQ classes for Java, le classi di uscita del canale scritte in Java non sono supportate a causa di un problema inerente al caricamento delle classi in un ambiente con più programmi di caricamento classi come OSGi. Un bundle utente può essere a conoscenza del bundle IBM WebSphere MQ classes for Java , ma il bundle IBM WebSphere MQ classes for Java non è a conoscenza di alcun bundle utente. Di conseguenza, il programma di caricamento classi utilizzato in un bundle IBM WebSphere MQ classes for Java non può caricare una classe di uscita canale che si trova in un bundle utente.

Per ulteriori informazioni su OSGi, consultare il sito Web OSGi alliance .

Il file di configurazione di IBM WebSphere MQ classes for Java

Un file di configurazione di IBM WebSphere MQ classes for Java specifica le proprietà utilizzate per configurare IBM WebSphere MQ classes for Java.

Il formato di un file di configurazione di IBM WebSphere MQ classes for Java è quello di un file delle proprietà standard di Java .

V 7.5.0.9 Da IBM WebSphere MQ Version 7.5.0, Fix Pack 9, un file di configurazione di esempio denominato `mqjava.config` viene fornito nella sottodirectory `bin` della directory di installazione di IBM WebSphere MQ classes for Java . Questo file documenta tutte le proprietà supportate e i relativi valori predefiniti.

Nota: Il file di configurazione di esempio viene sovrascritto quando l'installazione di IBM WebSphere MQ viene aggiornata a un fix pack futuro. Di conseguenza, si consiglia di creare una copia del file di configurazione di esempio da utilizzare con le applicazioni.

È possibile scegliere il nome e l'ubicazione di un file di configurazione IBM WebSphere MQ classes for Java . Quando si avvia l'applicazione, utilizzare un comando **java** con il seguente formato:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

Nel comando *config_file_url* è un URL (uniform resource locator) che specifica il nome e l'ubicazione del file di configurazione IBM WebSphere MQ classes for Java . Sono supportati URL dei seguenti tipi: `http`, `file`, `ftpe` `jar`.

Il seguente esempio mostra un comando **java** :

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/mqjava.config MyAppClass
```

Questo comando identifica il file di configurazione IBM WebSphere MQ classes for Java come file `D:\mydir\mqjava.config` sul sistema Windows locale.

Un file di configurazione di IBM WebSphere MQ classes for Java può essere utilizzato con qualsiasi trasporto supportato tra un'applicazione e un gestore code o broker.

Sovrascrittura delle proprietà specificate in un file di configurazione IBM WebSphere MQ classes for Java

Un file di configurazione IBM WebSphere MQ MQI client può specificare anche le proprietà utilizzate per configurare IBM WebSphere MQ classes for Java. Tuttavia, le proprietà specificate in un file di configurazione IBM WebSphere MQ MQI client vengono applicate solo quando un'applicazione si connette a un gestore code in modalità client.

Se necessario, è possibile sovrascrivere qualsiasi attributo in un file di configurazione di IBM WebSphere MQ MQI client specificandolo come proprietà in un file di configurazione di IBM WebSphere MQ classes for Java . Per sovrascrivere un attributo in un file di configurazione IBM WebSphere MQ MQI client , utilizzare una voce con il formato seguente nel file di configurazione IBM WebSphere MQ classes for Java :

```
com.ibm.mq.cfg.stanza.propName=propValue
```

Le variabili nella voce hanno i seguenti significati:

stanza
Il nome della sezione del file di configurazione IBM WebSphere MQ MQI client che contiene l'attributo.

propName
Il nome dell'attributo come specificato nel file di configurazione IBM WebSphere MQ MQI client .

propValue
Il valore della proprietà che sovrascrive il valore dell'attributo specificato nel file di configurazione IBM WebSphere MQ MQI client .

In alternativa, è possibile sovrascrivere un attributo in un file di configurazione di IBM WebSphere MQ MQI client specificando la proprietà come proprietà di sistema nel comando **java** . Utilizzare il formato precedente per specificare la proprietà come proprietà di sistema.

Solo i seguenti attributi in un file di configurazione IBM WebSphere MQ MQI client sono rilevanti per IBM WebSphere MQ classes for Java. Se si specificano o si sovrascrivono altri attributi, non ha alcun effetto. In particolare, notare che `ChannelDefinitionFile` e `ChannelDefinitionDirectory` nella stanza CHANNELS del file di configurazione client non vengono utilizzati. Consultare “Utilizzo di una tabella di definizione di canale client con IBM WebSphere MQ classes for Java” a pagina 674 per i dettagli su come utilizzare CCDT con IBM WebSphere MQ classes for Java.

Tabella 86. Quale stanza del file di configurazione client contiene quale attributo

stanza	Attributo
ClientExitStanza del percorso del file di configurazione client	ExitsDefaultPath
ClientExitStanza del percorso del file di configurazione client	ExitsDefaultPath64
ClientExitStanza del percorso del file di configurazione client	Percorso classi JavaExits
Stanza MessageBuffer del file di configurazione client	MaximumSize
Stanza MessageBuffer del file di configurazione client	PurgeTime
Stanza MessageBuffer del file di configurazione client	UpdatePercentage
Stanza TCP del file di configurazione client	ClntRcvBufSize
Stanza TCP del file di configurazione client	ClntSndBufSize

Tabella 86. Quale stanza del file di configurazione client contiene quale attributo (Continua)	
stanza	**Attributo**
Stanza TCP del file di configurazione client	Timeout connessione
Stanza TCP del file di configurazione client	KeepAlive

Per ulteriori informazioni sulla configurazione di IBM WebSphere MQ MQI client , consultare Configurazione di un client utilizzando un file di configurazione.

Attività correlate

Traccia delle applicazioni IBM WebSphere MQ classes for Java

Java Stanza Traccia ambiente standard

È possibile utilizzare la stanza Java Standard Environment Trace Settings per configurare la funzionalità di traccia IBM WebSphere MQ classes for Java .

com.ibm.msg.client.commonservices.trace.outputName = `traceOutputName`

`traceOutputName` è la directory e il nome file a cui viene inviato l'output di traccia.

Il nome predefinito del file di traccia dipende dalla versione di IBM WebSphere MQ classes for Java utilizzata da una applicazione:

- Per IBM WebSphere MQ classes for Java per Version 7.5.0, Fix Pack 8 o versioni precedenti, *traceOutputName* assume come valore predefinito un file denominato `mqjms_%PID%.trc` nella directory di lavoro corrente.
- V 7.5.0.9 Da IBM WebSphere MQ classes for Java da Version 7.5.0, Fix Pack 9, *traceOutputName* assume come valore predefinito un file denominato `mqjava_%PID%.trc` nella directory di lavoro corrente.

dove *%PID%* è l'ID processo corrente. Se un ID processo non è disponibile, viene generato un numero casuale con la lettera `f`come prefisso. Per includere l'ID processo in un nome file specificato, utilizzare la stringa `%PID%` .

Se si specifica una directory alternativa, è necessario che esista e che si disponga dell'autorizzazione di scrittura per questa directory. Se non si dispone dell'autorizzazione di scrittura, l'output di traccia viene scritto in `System.err`.

com.ibm.msg.client.commonservices.trace.include = `includeList`

`includeList` è un elenco di pacchetti e classi di cui viene eseguita la traccia o i valori speciali `ALL` o `NONE`.

Separare i nomi pacchetto o classe con un punto e virgola (`;`). **`includeList`** assume il valore predefinito `ALL`e traccia tutti i package e le classi in IBM WebSphere MQ classes for Java.

Nota: È possibile includere un package, ma escludere i package secondari di tale package. Ad esempio, se si include il pacchetto `a.b` e si esclude il pacchetto `a.b.x`, la traccia include tutto in `a.b.y` e `a.b.z`, ma non in `a.b.x` o `a.b.x.1`.

com.ibm.msg.client.commonservices.trace.exclude = `excludeList`

`excludeList` è un elenco di package e classi di cui non viene eseguita la traccia o i valori speciali `ALL` o `NONE`.

Separare i nomi pacchetto o classe con un punto e virgola (`;`). **`excludeList`** assume il valore predefinito `NONE`e pertanto non esclude dalla traccia alcun package e classe in IBM WebSphere MQ classes for Java .

Nota: È possibile escludere un package ma includere i package secondari di tale package. Ad esempio, se si esclude il pacchetto `a.b` e si include il pacchetto `a.b.x`, la traccia include tutto in `a.b.x` e `a.b.x.1`, ma non `a.b.y` o `a.b.z`.

Sono inclusi tutti i package o le classi specificati, allo stesso livello, inclusi ed esclusi.

com.ibm.msg.client.commonservices.trace.maxBytes = maxArrayBytes
maxArrayBytes è il numero massimo di byte tracciati da qualsiasi array di byte.

Se **maxArrayBytes** è impostato su un numero intero positivo, limita il numero di byte in una schiera di byte scritti nel file di traccia. Tronca la matrice di byte dopo aver scritto *maxArrayBytes* out. L'impostazione **maxArrayBytes** riduce la dimensione del file di traccia risultante e l'effetto della traccia sulle prestazioni dell'applicazione.

Un valore di `0` per questa proprietà indica che nessuno dei contenuti delle schiere di byte viene inviato al file di traccia.

Il valore predefinito è `-1`, che rimuove qualsiasi limite sul numero di byte in una schiera di byte inviati al file di traccia.

com.ibm.msg.client.commonservices.trace.limit = maxTraceBytes
maxTraceBytes è il numero massimo di byte scritti in un file di output di traccia.

maxTraceBytes funziona con **traceCycles**. Se il numero di byte di traccia scritti è vicino al limite, il file viene chiuso e viene avviato un nuovo file di output di traccia.

Il valore `0` indica che un file di output di traccia ha lunghezza zero. Il valore predefinito è `-1`, che significa che la quantità di dati da scrivere in un file di output di traccia è illimitata.

com.ibm.msg.client.commonservices.trace.count = traceCycles
traceCycles è il numero di file di output di traccia da scorrere.

Se il file di output di traccia corrente raggiunge il limite specificato da **maxTraceBytes**, il file viene chiuso. Un ulteriore output di traccia viene scritto nel successivo file di output di traccia in sequenza. Ogni file di output di traccia è distinto da un suffisso numerico che viene aggiunto al nome file. Il file di output di traccia corrente o più recente ha il suffisso `.trc.0`, il successivo file di output di traccia più recente termina con `.trc.1`e così via. I file di traccia più vecchi seguono lo stesso schema di numerazione fino al limite.

Il valore predefinito di **traceCycles** è 1. Se **traceCycles** è 1, quando il file di output di traccia corrente raggiunge la dimensione massima, il file viene chiuso ed eliminato. Viene avviato un nuovo file di output di traccia con lo stesso nome. Pertanto, esiste un solo file di output di traccia alla volta.

com.ibm.msg.client.commonservices.trace.parameter = traceParameters
traceParameters controlla se i parametri del metodo e i valori di ritorno sono inclusi nella traccia.

Il valore predefinito di **traceParameters** è TRUE. Se **traceParameters** è impostato su FALSE, viene eseguita la traccia solo delle firme del metodo.

com.ibm.msg.client.commonservices.trace.compress = compressedTrace
Impostare **compressedTrace** su TRUE per comprimere l'emissione della traccia.

Il valore predefinito di **compressedTrace** è FALSE.

Se **compressedTrace** è impostato su TRUE, l'output di traccia viene compresso. Il nome file di output di traccia predefinito ha l'estensione `.trz`. Se la compressione è impostata su FALSE, il valore predefinito, il file ha l'estensione `.trc` per indicare che è decompresso. Tuttavia, se il nome file per l'output di traccia è specificato in **traceOutputName** , tale nome viene utilizzato e non viene applicato alcun suffisso al file.

L'output della traccia compressa è più piccolo di quello decompresso. Poiché c'è meno I/O, può essere scritto più velocemente della traccia non compressa. La traccia compressa ha un effetto minore sulle prestazioni di IBM WebSphere MQ classes for Java rispetto alla traccia non compressa.

Se **maxTraceBytes** e **traceCycles** sono impostati, vengono creati più file di traccia compressi invece di più file flat.

Se il IBM WebSphere MQ classes for Java termina in modo non controllato, un file di traccia compresso potrebbe non essere valido. Per questo motivo, la compressione della traccia deve essere utilizzata solo quando IBM WebSphere MQ classes for Java viene chiuso in modo controllato. Utilizzare la compressione della traccia solo se i problemi esaminati non causano l'arresto imprevisto della JVM stessa. Non utilizzare la compressione della traccia quando si stanno diagnosticando

problemi che possono causare arresti `System.Halt()` o terminazioni JVM anomale e non controllate.

com.ibm.msg.client.commonservices.trace.level = `traceLevel`

`traceLevel` specifica un livello di filtro per la traccia. I livelli di traccia definiti sono i seguenti:

Tabella 87. Cosa viene tracciato per ciascun livello di traccia

Valore	Cosa viene tracciato
0	La traccia è disattivata
1	Eccezioni
3	Eccezioni Avvertenze
6	Eccezioni Avvertenze Punti di traccia informativi
8	Eccezioni Avvertenze Punti di traccia informativi Entrata e uscita metodo
9	Eccezioni Avvertenze Punti di traccia informativi Entrata e uscita metodo I dati inviati tra IBM WebSphere MQ classes for Java e un gestore code.

Nota: Utilizzare sempre il valore 9 se non diversamente indicato dal supporto IBM .

IBM WebSphere MQ classes for Java e strumenti di gestione software

Gli strumenti di gestione software come Apache Maven possono essere utilizzati con IBM WebSphere MQ classes for Java.

Molte grandi organizzazioni di sviluppo utilizzano questi strumenti per gestire centralmente i repository di librerie di terze parti.

I IBM WebSphere MQ classes for Java sono composti da un numero di file JAR. Quando si stanno sviluppando applicazioni in linguaggio Java utilizzando questa API, è richiesta un'installazione di IBM WebSphere MQ Server, IBM WebSphere MQ Client o IBM WebSphere MQ Client SupportPac sulla macchina su cui si sta sviluppando l'applicazione.

Se si desidera utilizzare uno strumento di gestione software e aggiungere i file JAR che costituiscono IBM WebSphere MQ classes for Java a un repository gestito centralmente, è necessario osservare i seguenti punti:

- Un repository o un contenitore deve essere reso disponibile solo agli sviluppatori all'interno della propria organizzazione. Non è consentita alcuna distribuzione al di fuori dell'organizzazione.
- Il repository deve contenere una serie completa e coerente di file JAR da una singola release IBM WebSphere MQ o Fix Pack.

- Sei responsabile dell'aggiornamento del repository con qualsiasi manutenzione fornita dal supporto IBM .

Per IBM WebSphere MQ Version 7.5, i seguenti file JAR devono essere installati nel repository:

- 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

Configurazione post - installazione per applicazioni IBM WebSphere MQ

Dopo l'installazione di IBM WebSphere MQ, è possibile configurare l'installazione in modo da eseguire le proprie applicazioni.

Ricordarsi di controllare il file README IBM WebSphere MQ per informazioni successive o più specifiche per il proprio ambiente.

Prima di provare ad eseguire un'applicazione IBM WebSphere MQ classes per Java in modalità bind, accertarsi di aver configurato IBM WebSphere MQ come descritto in Configurazione .

Configurazione del gestore code per accettare le connessioni client dalle classi WebSphere MQ per Java

Per configurare il gestore code per accettare le richieste di connessione in entrata dai client, definire e consentire l'utilizzo di un canale di connessione server e avviare un programma listener.

Vedi “Preparazione ed esecuzione dei programmi di esempio” a pagina 110 per i dettagli.

Esecuzione delle classi WebSphere MQ per le applicazioni Java in Java Security Manager

Le classi WebSphere MQ per Java possono essere eseguite con Java Security Manager abilitato. Per eseguire correttamente le applicazioni con il Security Manager abilitato, è necessario configurare la JVM (Java virtual machine) con un file di definizione della politica adatto.

Il modo più semplice per eseguire questa operazione è modificare il file delle politiche fornito con JRE. Sulla maggior parte dei sistemi, questo file è memorizzato nel percorso `lib/security/java.policy`, relativo alla directory JRE. È possibile modificare i file delle politiche utilizzando il proprio editor preferito o il programma `policytool` fornito con JRE.

È necessario fornire l'autorizzazione al file com.ibm.mq.jmqi.jar in modo che possa:

- Crea socket (in modalità client)
- Carica la libreria nativa (in modalità bind)
- Leggi varie proprietà dall'ambiente

La proprietà di sistema os.name deve essere disponibile per le classi WebSphere MQ per Java durante l'esecuzione in Java Security Manager.

Di seguito viene riportato un esempio di voce del file delle politiche che consente alle classi WebSphere MQ per Java di essere eseguite correttamente nel gestore di sicurezza predefinito:

```
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";
};
```

Note:

- *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ .
- Questo esempio di un file delle politiche consente alle classi WebSphere MQ per Java di funzionare correttamente nel gestore della sicurezza, ma potrebbe essere ancora necessario abilitare il proprio codice per essere eseguito correttamente prima che le applicazioni funzionino.
- Per consentire alle classi WebSphere MQ per Java di accedere ai file JAR (Java archive) di un'applicazione, aggiungere la seguente autorizzazione alla prima istruzione `grant` :

  ```
  permission java.io.FilePermission "/path_to_your_app/-", "read";
  ```

- Per utilizzare queste istruzioni `grant` nel file di configurazione della politica, potrebbe essere necessario modificare i nomi percorso in base a dove sono state installate le classi WebSphere MQ per Java e dove sono state memorizzate le proprie applicazioni.
- Il codice di esempio fornito con WebSphere MQ classes for Java non è stato specificamente abilitato per l'utilizzo con il gestore della sicurezza; tuttavia, i test IVT vengono eseguiti con questo file delle politiche e con il gestore della sicurezza predefinito.

Verifica delle classi IBM WebSphere MQ per l'installazione di Java

Un programma di verifica dell'installazione, MQIVP, viene fornito con le classi IBM WebSphere MQ per Java. È possibile utilizzare questo programma per verificare tutte le modalità di connessione delle classi IBM WebSphere MQ per Java.

Il programma richiede un numero di scelte e altri dati per determinare la modalità di connessione che si desidera verificare. Utilizzare la seguente procedura per verificare l'installazione:

1. Se si sta per eseguire il programma in modalità client, configurare il gestore code come descritto in “Preparazione ed esecuzione dei programmi di esempio” a pagina 110. La coda da utilizzare è SYSTEM.DEFAULT.LOCAL.QUEUE.
2. Se si sta per eseguire il programma in modalità client, consultare anche “Utilizzo di classi WebSphere MQ per Java” a pagina 654.

   Eseguire i passi rimanenti di questa procedura sul sistema su cui si sta per eseguire il programma.
3. Accertarsi di aver aggiornato la propria variabile di ambiente CLASSPATH in conformità con le istruzioni in “Variabili di ambiente relative alle classi WebSphere MQ per Java” a pagina 658.

4. Passare alla directory MQ_INSTALLATION_PATH/mqm/VRM/java/samples/wmqjava, dove MQ_INSTALLATION_PATH è il percorso per l'installazione di IBM WebSphere MQ e VRM è il numero di versione, release e modifica del prodotto. Quindi, nel prompt dei comandi, immettere:

   ```
   java -Djava.library.path=library_path MQIVP
   ```

   dove *library_path* è il percorso delle classi IBM WebSphere MQ per le librerie Java (consultare The WebSphere MQ classes per le librerie Java).

   Al prompt contrassegnato (1):

   - Per utilizzare una connessione TCP/IP, immettere un nome host server IBM WebSphere MQ .
   - Per utilizzare la connessione nativa (modalità bind), lasciare il campo vuoto (non immettere un nome).

   Il programma tenta di:

   - 1. Collegarsi al gestore code
   - 2. Aprire la coda SYSTEM.DEFAULT.LOCAL.QUEUE, inserire un messaggio nella coda, richiamare un messaggio dalla coda e chiudere la coda
   - 3. Disconnettersi dal gestore code
   - 4. Restituire un messaggio se le operazioni hanno esito positivo

Di seguito è riportato un esempio delle richieste e delle risposte che potrebbero essere visualizzate. Le richieste e le risposte effettive dipendono dalla rete 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 ...
```

Nota:

1. Se si sceglie la connessione server, non vengono visualizzati i prompt contrassegnati con (2).

Risoluzione dei problemi di IBM WebSphere MQ

Inizialmente, eseguire il programma di verifica dell'installazione. Potrebbe anche essere necessario utilizzare la funzionalità di traccia.

Se un programma non viene completato correttamente, eseguire il programma di verifica dell'installazione e seguire i consigli forniti nei messaggi di diagnostica. Questo programma è descritto in “Verifica delle classi IBM WebSphere MQ per l'installazione di Java” a pagina 667.

Se il problema persiste e è necessario contattare il team di assistenza IBM , potrebbe essere richiesto di attivare la funzionalità di traccia. Eseguire questa operazione come mostrato nel seguente esempio.

Per tracciare il programma MQIVP :

- creare un file delle proprietà *com.ibm.mq.commonservices* (consultare Utilizzo di com.ibm.mq.commonservices .

- Immettere il seguente comando:

```
java -Dcom.ibm.mq.commonservices=commonservices_properties_file java
-Djava.library.path=library_path MQIVP -trace
```

dove:

- *commonservices_properties_file* è il percorso (incluso il nome file) del file delle proprietà *com.ibm.mq.commonservices* .
- *library_path* è il percorso delle classi WebSphere MQ per librerie Java (consultare The WebSphere MQ classes per librerie Java).

Per ulteriori informazioni su come utilizzare la traccia, consultare Traccia delle applicazioni IBM WebSphere MQ classes for Java.

Introduzione per i programmatori

Questa raccolta di argomenti contiene informazioni generali per i programmatori.

Per informazioni più dettagliate sulla scrittura di programmi, consultare “Scrittura delle classi WebSphere MQ per le applicazioni Java” a pagina 669.

Le classi WebSphere MQ per l'interfaccia Java

L'API (application programming interface) procedurale WebSphere MQ utilizza i verbi, che agiscono sugli oggetti. L'interfaccia di programmazione Java utilizza oggetti su cui si agisce richiamando i metodi.

L'API (application programming interface) WebSphere MQ procedurale è basata su verbi come questi:

```
       MQBACK, MQBEGIN, MQCLOSE, MQCONN, MQDISC,
       MQGET, MQINQ, MQOPEN, MQPUT, MQSET, MQSUB
```

Tutti questi verbi assumono, come parametro, un handle all'oggetto WebSphere MQ su cui devono operare. Poiché Java è orientato all'oggetto, l'interfaccia di programmazione Java gira questo round. Il programma è costituito da una serie di oggetti WebSphere MQ , su cui si agisce richiamando i metodi su tali oggetti.

Quando si utilizza l'interfaccia procedurale, ci si disconnette da un gestore code utilizzando la chiamata MQDISC (Hconn, CompCode, Reason), dove *Hconn* è un handle per il gestore code.

Nell'interfaccia Java, il gestore code è rappresentato da un oggetto di classe MQQueueManager. È possibile disconnettersi dal gestore code richiamando il metodo disconnect () su tale classe.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.disconnect();
```

Scrittura delle classi WebSphere MQ per le applicazioni Java

Questa raccolta di argomenti fornisce informazioni utili per la scrittura di applicazioni Java per interagire con i sistemi WebSphere MQ .

Per utilizzare le classi WebSphere MQ per Java per accedere alle code WebSphere MQ , scrivere le applicazioni Java che contengono le chiamate che inserono i messaggi e richiamano i messaggi dalle code WebSphere MQ . Per informazioni dettagliate sulle singole classi, vedere WebSphere MQ classes per Java.

Nota: La riconnessione automatica del client non è supportata dalle classi WebSphere MQ per Java.

Classi WebSphere MQ per le modalità di connessione Java

Il modo in cui si programma per WebSphere MQ classes for Java ha alcune dipendenze sulle modalità di connessione che si desidera utilizzare.

Se si utilizzano le connessioni client, esistono diverse differenze rispetto a IBM WebSphere MQ MQI client , ma è concettualmente simile. Se si utilizza la modalità di bind, è possibile utilizzare i bind del percorso rapido e immettere il comando MQBEGIN. Specificare la modalità da utilizzare impostando le variabili nella classe MQEnvironment.

Classi WebSphere MQ per connessioni client Java

Quando le classi WebSphere MQ per Java vengono utilizzate come client, è simile a IBM WebSphere MQ MQI client, ma presenta una serie di differenze.

Se si sta programmando per *WebSphere MQ classes for Java* da utilizzare come client, tenere presenti le seguenti differenze:

- Supporta solo TCP/IP.
- Non legge alcuna variabile di ambiente WebSphere MQ all'avvio.
- Le informazioni memorizzate in una definizione di canale e nelle variabili di ambiente possono essere memorizzate in una classe denominata Ambiente. In alternativa, queste informazioni possono essere trasmesse come parametri quando viene effettuata la connessione.
- Le condizioni di errore e di eccezione vengono scritte in un log specificato nella classe `MQException` . La destinazione errori predefinita è la Console Java.
- Solo i seguenti attributi in un file di configurazione client WebSphere MQ sono rilevanti per le classi WebSphere MQ per Java. Se si specificano altri attributi, essi sono inefficaci.

stanza	**Attributo**
ClientExitStanza del percorso del file di configurazione client	ExitsDefaultPath
ClientExitStanza del percorso del file di configurazione client	ExitsDefaultPath64
ClientExitStanza del percorso del file di configurazione client	Percorso classi JavaExits
Stanza MessageBuffer del file di configurazione client	MaximumSize
Stanza MessageBuffer del file di configurazione client	PurgeTime
Stanza MessageBuffer del file di configurazione client	UpdatePercentage
Stanza TCP del file di configurazione client	ClntRcvBufSize
Stanza TCP del file di configurazione client	ClntSndBufSize
Stanza TCP del file di configurazione client	Timeout connessione
Stanza TCP del file di configurazione client	KeepAlive

- Se ci si connette a un gestore code che richiede la conversione dei dati carattere, il client Java V7 è ora in grado di eseguire la conversione se il gestore code non è in grado di farlo. La JVM client deve supportare la conversione tra il CCSID del client e quello del gestore code.
- La riconnessione automatica del client non è supportata dalle classi WebSphere MQ per Java.

Quando utilizzato in modalità client, *WebSphere MQ classes for Java* non supporta la chiamata MQBEGIN.

Consultare “Opzioni di connessione per WebSphere MQ classes for Java” a pagina 655 per ulteriori informazioni relative agli ambienti supportati.

WebSphere MQ classi per la modalità di bind Java

La modalità di bind di WebSphere MQ classes for Java differisce dalla modalità client in tre modi principali.

Quando viene utilizzato in modalità di bind, le classi WebSphere MQ per Java utilizzano JNI (Java Native Interface) per richiamare direttamente l'API del gestore code esistente, piuttosto che comunicare tramite una rete.

Per impostazione predefinita, le applicazioni che utilizzano le classi di WebSphere MQ per Java in modalità di bind si connettono a un gestore code utilizzando *ConnectOption*, MQCNO_STANDARD_BINDINGS.

Le classi WebSphere MQ per Java supportano le seguenti *ConnectOptions*:

- MQCNO_FASTPATH_BINDING
- MQCNO_STANDARD_BINDING
- MQCNO_SHARED_BINDING
- MQCNO_ISOLATED_BINDING

Per ulteriori informazioni su *ConnectOptions*, consultare “Connessione a un gestore code utilizzando la chiamata MQCONNX” a pagina 209.

La modalità di bind supporta la chiamata MQBEGIN per avviare le unità di lavoro globali coordinate dal gestore code, su tutte le piattaforme tranne WebSphere MQ per IBM i e WebSphere MQ per z/OS.

La maggior parte dei parametri forniti dalla classe `MQEnvironment` non sono rilevanti per la modalità di bind e vengono ignorati.

Consultare “Opzioni di connessione per WebSphere MQ classes for Java” a pagina 655 per ulteriori informazioni relative agli ambienti supportati.

Definizione delle classi WebSphere MQ da utilizzare per la connessione Java

Il tipo di connessione da utilizzare è determinato dall'impostazione delle variabili nella classe MQEnvironment.

Vengono utilizzate due variabili:

MQEnvironment.properties
Il tipo di connessione è determinato dal valore associato con il nome chiave `CMQC.TRANSPORT_PROPERTY`. I valori possibili sono i seguenti:

CMQC.TRANSPORT_MQSERIES_BINDINGS
Connetti in modalità bind

CMQC.TRANSPORT_MQSERIES_CLIENT
Connetti in modalità client

CMQC.TRANSPORT_MQSERIES
La modalità di connessione è determinata dal valore della proprietà *hostname*

MQEnvironment.hostname
Impostare il valore di questa variabile come segue:

- Per le connessioni client, impostare il valore di questa variabile sul nome host del server IBM WebSphere MQ a cui si desidera connettersi
- Per la modalità di bind, non impostare questa variabile o impostarla su null

Operazioni sui gestori code

Questa raccolta di argomenti descrive come connettersi e disconnettersi da un gestore code utilizzando le classi WebSphere MQ per Java.

Impostazione dell'ambiente WebSphere MQ per le classi WebSphere MQ per Java

Perché un'applicazione si connetta a un gestore code in modalità client, l'applicazione deve specificare il nome canale, il nome host e il numero di porta.

Nota: Le informazioni contenute in questo argomento sono rilevanti solo se l'applicazione si connette a un gestore code in modalità client. *Non* è rilevante se si connette in modalità bind. Consultare: “Modalità di connessione per le classi WebSphere MQ per JMS” a pagina 774

È possibile specificare il nome del canale, il nome host e il numero di porta in due modi: come campi nella classe MQEnvironment o come proprietà dell'oggetto MQQueueManager .

Se si impostano i campi nella classe MQEnvironment, essi si applicano all'intera applicazione, tranne dove vengono sovrascritti da una tabella hash delle proprietà. Per specificare il nome canale e il nome host in MQEnvironment, utilizzare il seguente codice:

```
MQEnvironment.hostname = "host.domain.com";
MQEnvironment.channel  = "java.client.channel";
```

Ciò equivale all'impostazione di una variabile di ambiente **MQSERVER** :

```
"java.client.channel/TCP/host.domain.com".
```

Per impostazione predefinita, i client Java tentano di collegarsi a un listener WebSphere MQ sulla porta 1414. Per specificare una porta diversa, utilizzare il seguente codice:

```
MQEnvironment.port = nnnn;
```

dove nnnn è il numero di porta richiesto

Se si passano le proprietà a un oggetto gestore code al momento della sua creazione, esse si applicano solo a tale gestore code. Creare le voci in un oggetto Hashtable con chiavi di **hostname**, **channel**e, facoltativamente, **port**e con valori appropriati. Per utilizzare la porta predefinita 1414, è possibile omettere la voce **port** . Creare l'oggetto MQQueueManager utilizzando un costruttore che accetti la tabella hash delle proprietà.

Identificazione di una connessione al gestore code impostando un nome applicazione

Un'applicazione può impostare un nome che identifica la connessione al gestore code. Questo nome applicazione viene visualizzato dal comando **DISPLAY CONN MQSC/PCF** (dove il campo è denominato **APPLTAG**) o nel pannello WebSphere MQ Explorer **Application Connections** (dove il campo è denominato **App name**).

I nomi delle applicazioni sono limitati a 28 caratteri e i nomi più lunghi sono troncati per adattarsi. Se non viene specificato un nome applicazione, viene fornito un nome predefinito. Il nome predefinito si basa sulla classe di richiamo (principale), ma se queste informazioni non sono disponibili, viene utilizzato il testo `WebSphere MQ Client per Java` .

Se viene utilizzato il nome della classe di richiamo, viene adattato per adattarlo rimuovendo i nomi di pacchetto iniziali, se necessario. Ad esempio, se la classe di richiamo è `com.example.MainApp`, viene utilizzato il nome completo, ma se la classe di richiamo è `com.example.dictionaryAndThesaurus.multilingual.mainApp`, viene utilizzato il nome `multilingual.mainApp` , poiché è la combinazione più lunga di nome classe e nome pacchetto più a destra che si adatta alla lunghezza disponibile.

Se il nome della classe è più lungo di 28 caratteri, viene troncato per adattarlo. Ad esempio, `com.example.mainApplicationForSecondTestCase` diventa `mainApplicationForSecondTest`.

Nota: I gestori code in esecuzione su piattaforme z/OS non supportano l'impostazione dei nomi applicazione.

Per impostare un nome applicazione nella classe MQEnvironment, aggiungere il nome alla tabella hash MQEnvironment.properties , con chiave **MQConstants.APPNAME_PROPERTY**, utilizzando il seguente codice:

```
MQEnvironment.properties.put(MQConstants.APPNAME_PROPERTY, "my_application_name");
```

Per impostare un nome dell'applicazione nella tabella hash delle proprietà passata al costruttore MQQueueManager , aggiungere il nome alla tabella hash delle proprietà con una chiave **MQConstants.APPNAME_PROPERTY**.

Sovrascrittura delle proprietà specificate in un file di configurazione client WebSphere MQ

Un file di configurazione del client WebSphere MQ può specificare anche le proprietà utilizzate per configurare le classi WebSphere MQ per Java. Tuttavia, le proprietà specificate in un WebSphere MQ file di configurazione del client MQI si applicano solo quando un'applicazione si connette a un gestore code in modalità client.

Se richiesto, è possibile sovrascrivere qualsiasi attributo in un file di configurazione WebSphere MQ in uno dei modi riportati di seguito. Le opzioni vengono visualizzate in ordine di precedenza.

- Impostare una proprietà di sistema Java per la proprietà di configurazione.
- Impostare la proprietà nella mappa MQEnvironment.properties .
- In Java5 e release successivi, impostare una variabile di ambiente di sistema.

Solo i seguenti attributi in un file di configurazione client WebSphere MQ sono rilevanti per le classi WebSphere MQ per Java. Se si specificano o si sovrascrivono altri attributi, non ha alcun effetto.

stanza	Attributo
ClientExitStanza del percorso del file di configurazione client	ExitsDefaultPath
ClientExitStanza del percorso del file di configurazione client	ExitsDefaultPath64
ClientExitStanza del percorso del file di configurazione client	Percorso classi JavaExits
Stanza MessageBuffer del file di configurazione client	MaximumSize
Stanza MessageBuffer del file di configurazione client	PurgeTime
Stanza MessageBuffer del file di configurazione client	UpdatePercentage
Stanza TCP del file di configurazione client	ClntRcvBufSize
Stanza TCP del file di configurazione client	ClntSndBufSize
Stanza TCP del file di configurazione client	Timeout connessione
Stanza TCP del file di configurazione client	KeepAlive

Connessione a un gestore code nelle classi WebSphere MQ per Java

Connettersi a un gestore code creando una nuova istanza della classe MQQueueManager . Disconnettersi da un gestore code richiamando il metodo disconnect ().

Sei ora pronto a connetterti a un gestore code creando una nuova istanza della classe MQQueueManager :

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Per disconnettersi da un gestore code, richiamare il metodo disconnect () sul gestore code:

```
queueManager.disconnect();
```

Se si richiama il metodo di disconnessione, tutte le code aperte e i processi a cui è stato effettuato l'accesso tramite tale gestore code vengono chiusi. Tuttavia, è buona pratica di programmazione chiudere esplicitamente queste risorse quando si finisce di utilizzarle. Per eseguire questa operazione, utilizzare il metodo close () sugli oggetti pertinenti.

I metodi commit () e backout () su un gestore code sono equivalenti ai richiami MQCMIT e MQBACK utilizzati con l'interfaccia procedurale.

Utilizzo di una tabella di definizione di canale client con IBM WebSphere MQ classes for Java

Le classi IBM WebSphere MQ classes for Java possono utilizzare le definizioni di canale di connessione client memorizzate in una CCDT (client channel definition table).

Come alternativa alla creazione di una definizione di canale di connessione client impostando determinati campi e proprietà di ambiente nella classe `MQEnvironment` o trasmettendoli a `MQQueueManager` in una tabella hash delle proprietà, un'applicazione client IBM WebSphere MQ classes for Java può utilizzare le definizioni di canale di connessione client memorizzate in una tabella di definizione di canale client. Queste definizioni vengono create dai comandi MQSC (IBM WebSphere MQ Script) o PCF (IBM WebSphere MQ Programmable Command Format) oppure utilizzando IBM WebSphere MQ Explorer.

Quando l'applicazione crea un oggetto `MQQueueManager` , il client delle classi IBM WebSphere MQ classes for Java ricerca la tabella di definizione del canale client per una definizione del canale di connessione client adatta e utilizza la definizione del canale per avviare un canale MQI. Per ulteriori informazioni sulle tabelle di definizione dei canali client e su come crearne una, consultare Tabella di definizione dei canali client.

Per utilizzare una tabella di definizione del canale client, un'applicazione deve prima creare un oggetto URL. L'oggetto URL incapsula un URL (uniform resource locator) che identifica il nome e l'ubicazione del file contenente la tabella di definizione del canale client e specifica il modo in cui è possibile accedere al file.

Ad esempio, se il file `ccdt1.tab` contiene una tabella di definizione del canale client ed è memorizzato nello stesso sistema su cui è in esecuzione l'applicazione, l'applicazione può creare un oggetto URL nel modo seguente:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
```

Come altro esempio, si supponga che il file `ccdt2.tab` contenga una tabella di definizione del canale client e che sia memorizzato su un sistema diverso da quello su cui è in esecuzione l'applicazione. Se è possibile accedere al file utilizzando il protocollo FTP, l'applicazione può creare un oggetto URL nel modo seguente:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
```

Una volta creato un oggetto URL, l'applicazione può creare un oggetto `MQQueueManager` utilizzando uno dei costruttori che utilizza un oggetto URL come parametro. Di seguito è riportato un esempio:

```
MQQueueManager mars = new MQQueueManager("MARS", chanTab2);
```

Questa istruzione fa sì che le classi IBM WebSphere MQ classes for Java acceda alla tabella di definizione del canale client identificata dall'oggetto URL chanTab2, ricerchi la tabella per una definizione di canale di

connessione client adatta e quindi utilizzi la definizione del canale per avviare un canale MQI sul gestore code denominato MARS.

Notare i seguenti punti che si applicano se un'applicazione utilizza una tabella di definizione del canale client:

- Quando l'applicazione crea un oggetto `MQQueueManager` utilizzando un costruttore che utilizza un oggetto URL come parametro, non è necessario impostare alcun nome di canale nella classe `MQEnvironment` , come campo o come proprietà di ambiente. Se è impostato un nome canale, le classi IBM WebSphere MQ classes for Java lanciano un `MQException`. La proprietà del campo o dell'ambiente che specifica il nome del canale viene considerata impostata se il suo valore è diverso da null, una stringa vuota o una stringa contenente tutti caratteri vuoti.
- Il parametro **queueManagerName** nel costruttore `MQQueueManager` può avere uno dei seguenti valori:
  - Il nome di un gestore code
  - Un asterisco (*) seguito dal nome di un gruppo di gestori code
  - Un asterisco (*)
  - Null, una stringa vuota o una stringa contenente tutti i caratteri vuoti

  Si tratta degli stessi valori che possono essere utilizzati per il parametro **QMgrName** in una chiamata MQCONN emessa da un'applicazione client che utilizza MQI (Message Queue Interface). Per ulteriori informazioni sul significato di questi valori, consultare “Panoramica su Message Queue Interface” a pagina 195.

  Se l'applicazione utilizza il pool di connessioni, consultare “Controllo del pool di connessioni predefinito nelle classi WebSphere MQ per Java” a pagina 695.
- Quando il client IBM WebSphere MQ classes for Java trova una definizione di canale di connessione client adatta nella tabella di definizione di canale client, utilizza solo le informazioni estratte da questa definizione di canale per avviare un canale MQI. Tutti i campi relativi al canale o le proprietà dell'ambiente che l'applicazione potrebbe aver impostato nella classe `MQEnvironment` vengono ignorate.

  In particolare, notare i seguenti punti se si utilizza SSL (Secure Sockets Layer):
  - Un canale MQI utilizza SSL solo se la definizione di canale estratta dalla tabella di definizione di canale del client specifica il nome di un CipherSpec supportato dalle classi IBM WebSphere MQ classes for Java .
  - Una tabella di definizione del canale client contiene anche informazioni sull'ubicazione dei server LDAP (Lightweight Directory Access Protocol) che contengono i CRL (Certificate Revocation List). Le classi IBM WebSphere MQ classes for Java utilizzano solo queste informazioni per accedere a server LDAP che contengono CRL.
  - Una tabella di definizione di canale client può contenere anche l'ubicazione di un responder OCSP (Online Certificate Status Protocol). IBM WebSphere MQ classes for Java non può utilizzare le informazioni OCSP in un file di tabella di definizione del canale client. Tuttavia, è possibile configurare OCSP come descritto nella sezione Utilizzo di Online Certificate Protocol.

  Per ulteriori informazioni sull'utilizzo di SSL con una tabella di definizione del canale client, consultare Specifica che un canale MQI utilizza SSL.

  Notare anche i seguenti punti se si utilizzano le uscite del canale:
  - Un canale MQI utilizza le uscite canale e i dati utente associati specificati dalla definizione del canale estratta dalla tabella di definizione del canale client, preferendo le uscite canale e i dati specificati utilizzando altri metodi.
  - Una definizione di canale estratta da una tabella di definizione di canale client può specificare uscite di canale scritte in Java, C o C + +. Per ulteriori informazioni su come scrivere un'uscita di canale in Java, consultare “Creazione di un'uscita di canale in classi WebSphere MQ per Java” a pagina 688. Per ulteriori informazioni su come scrivere un'uscita canale in altre lingue, consultare “Utilizzo di uscite di canale non scritte in Java con classi WebSphere MQ per Java” a pagina 692.

Specifica di un intervallo di porte per le connessioni client IBM WebSphere MQ classes for Java

È possibile specificare una porta o un intervallo di porte a cui un'applicazione può collegarsi in uno dei due modi.

Quando un'applicazione IBM WebSphere MQ classes for Java tenta di connettersi a un gestore code IBM WebSphere MQ in modalità client, un firewall potrebbe consentire solo le connessioni che hanno origine da porte specificate o da un intervallo di porte. In questa situazione, è possibile specificare una porta o un intervallo di porte a cui l'applicazione può collegarsi. È possibile specificare le porte nei modi seguenti:

- È possibile impostare il campo di impostazione localAddressnella classe MQEnvironment. Di seguito è riportato un esempio:

```
MQEnvironment.localAddressSetting = "192.0.2.0(2000,3000)";
```

- È possibile impostare la proprietà di ambiente CMQC.LOCAL_ADDRESS_PROPERTY. Di seguito è riportato un esempio:

```
(MQEnvironment.properties).put(CMQC.LOCAL_ADDRESS_PROPERTY,
                               "192.0.2.0(2000,3000)");
```

- Quando è possibile creare l'oggetto MQQueueManager , è possibile passare una tabella di hash delle proprietà contenente una LOCAL_ADDRESS_PROPERTY con il valore "192.0.2.0(2000,3000)"

In ognuno di questi esempi, quando l'applicazione si connette successivamente a un gestore code, l'applicazione si collega a un indirizzo IP locale e a un numero di porta compresi nell'intervallo tra 192.0.2.0(2000) e 192.0.2.0(3000).

In un sistema con più di un'interfaccia di rete, è anche possibile utilizzare il campo Impostazione localAddresso la proprietà di ambiente CMQC.LOCAL_ADDRESS_PROPERTY, per specificare quale interfaccia di rete deve essere utilizzata per una connessione.

Gli errori di connessione potrebbero verificarsi se si limita l'intervallo di porte. Se si verifica un errore, viene generata una MQException contenente il codice motivo di IBM WebSphere MQ MQRC_Q_MGR_NOT_AVAILABLE e il seguente messaggio:

```
Socket connection attempt refused due to LOCAL_ADDRESS_PROPERTY restrictions
```

Si potrebbe verificare un errore se tutte le porte nell'intervallo specificato sono in uso o se l'indirizzo IP, il nome host o il numero di porta specificati non sono validi (ad esempio, un numero di porta negativo).

Accesso a code, argomenti e processi nelle classi WebSphere MQ per Java

Per accedere a code, argomenti e processi, utilizzare i metodi della classe MQQueueManager . MQOD (object descriptor structure) è compresso nei parametri di questi metodi.

Code

Per aprire una coda è possibile utilizzare il metodo accessQueue della classe MQQueueManager . Ad esempio, su un gestore code denominato queueManager, utilizzare il seguente codice:

```
MQQueue queue = queueManager.accessQueue("qName",CMQC.MQOO_OUTPUT);
```

Il metodo accessQueue restituisce un nuovo oggetto della classe MQQueue.

Una volta terminato di utilizzare la coda, utilizzare il metodo close () per chiuderla, come nel seguente esempio:

```
queue.close();
```

È anche possibile creare una coda utilizzando il costruttore MQQueue. I parametri sono esattamente gli stessi del metodo accessQueue , con l'aggiunta di un parametro del gestore code. Ad esempio:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            CMQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserID");
```

È possibile specificare un certo numero di opzioni quando si creano code. Per i dettagli, consultare Class.com.ibm.mq.MQQueue. La creazione di un oggetto coda in questo modo consente di scrivere le proprie sottoclassi di MQQueue.

Argomenti

Allo stesso modo, è possibile aprire un topic utilizzando il metodo accessTopic della classe MQQueueManager . Ad esempio, su un gestore code denominato queueManager, utilizzare il seguente codice per creare un sottoscrittore e un publisher:

```
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);
```

Una volta terminato di utilizzare l'argomento, utilizzare il metodo close () per chiuderlo.

È anche possibile creare un argomento utilizzando il costruttore MQTopic. I parametri sono esattamente gli stessi del metodo accessTopic , con l'aggiunta di un parametro del gestore code. Ad esempio:

```
MQTopic subscriber = new
      MQTopic(queueManager,"TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

È possibile specificare un certo numero di opzioni quando si creano argomenti. Per i dettagli, consultare Class com.ibm.mq.MQTopic . La creazione di un oggetto argomento in questo modo consente di scrivere le proprie sottoclassi di MQTopic.

Un argomento deve essere aperto per la pubblicazione o per la sottoscrizione. La classe MQQueueManager ha otto metodi accessTopic e la classe Topic ha otto costruttori. In ogni caso, quattro di questi hanno un parametro **destination** e quattro hanno un parametro **subscriptionName** (inclusi due che hanno entrambi). Questi possono essere utilizzati solo per aprire l'argomento per le sottoscrizioni. I due metodi restanti hanno un parametro **openAs** e l'argomento può essere aperto per la pubblicazione o la sottoscrizione a seconda del valore del parametro **openAs** .

Per creare un argomento come sottoscrittore durevole, utilizzare un metodo accessTopic della classe MQQueueManager o un costruttore MQTopic che accetta un nome di sottoscrizione e, in entrambi i casi, impostare CMQC.MQSO_DURABLE .

Processi

Per accedere a un processo, utilizzare il metodo accessProcess di MQQueueManager. Ad esempio, su un gestore code denominato queueManager, utilizzare il codice riportato di seguito per creare un oggetto MQProcess:

```
MQProcess process =
      queueManager.accessProcess("PROCESSNAME",
      CMQC.MQOO_FAIL_IF_QUIESCING);
```

Per accedere a un processo, utilizzare il metodo accessProcess di MQQueueManager.

Il metodo accessProcess restituisce un nuovo oggetto della classe MQProcess.

Una volta terminato di utilizzare l'oggetto processo, utilizzare il metodo close () per chiuderlo, come nel seguente esempio:

```
process.close();
```

È anche possibile creare un processo utilizzando il costruttore MQProcess. I parametri sono esattamente gli stessi del metodo accessProcess , con l'aggiunta di un parametro del gestore code. Ad esempio:

```
MQProcess process =
     new MQProcess(queueManager,"PROCESSNAME",
     CMQC.MQOO_FAIL_IF_QUIESCING);
```

La costruzione di un oggetto processo in questo modo consente di scrivere le proprie sottoclassi di MQProcess.

Gestione dei messaggi nelle classi WebSphere MQ per Java

I messaggi sono rappresentati dalla classe MQMessage. I messaggi vengono inseriti e ricevuti utilizzando i metodi della classe MQDestination, che dispone di sottoclassi MQQueue e MQTopic.

Inserire i messaggi nelle code o negli argomenti utilizzando il metodo put () della classe MQDestination. I messaggi vengono ricevuti dalle code o dagli argomenti utilizzando il metodo get () della classe MQDestination. A differenza dell'interfaccia procedurale, in cui MQPUT e MQGET immettono e ottengono array di byte, il linguaggio di programmazione Java immette e richiama le istanze della classe MQMessage. La classe MQMessage incapsula il buffer di dati che contiene i dati del messaggio effettivi, insieme a tutti i parametri MQMD (descrittore del messaggio) e le proprietà del messaggio che descrivono tale messaggio.

Per generare un nuovo messaggio, creare una nuova istanza della classe MQMessage e utilizzare i metodi writeXXX per inserire i dati nel buffer del messaggio.

Quando viene creata la nuova istanza di messaggio, tutti i parametri MQMD vengono impostati automaticamente sui loro valori predefiniti, come definito in Valori iniziali e dichiarazioni della lingua per MQMD . Il metodo put () di MQDestination utilizza anche un'istanza della classe di opzioni MQPutMessagecome parametro. Questa classe rappresenta la struttura MQPMO. Il seguente esempio crea un messaggio e lo inserisce in una coda:

```
// 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);
```

Il metodo get () di MQDestination restituisce una nuova istanza di MQMessage, che rappresenta il messaggio appena preso dalla coda. Inoltre, utilizza un'istanza della classe di opzioni MQGetMessagecome parametro. Questa classe rappresenta la struttura MQGMO.

Non è necessario specificare una dimensione massima del messaggio, poiché il metodo get () regola automaticamente la dimensione del buffer interno in modo che si adatti al messaggio in entrata. Utilizzare i metodi readXXX della classe MQMessage per accedere ai dati nel messaggio restituito.

Il seguente esempio mostra come ottenere un messaggio da una coda:

```
// 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);
```

È possibile modificare il formato numerico utilizzato dai metodi di lettura e scrittura impostando la variabile membro *codifica* .

È possibile modificare la serie di caratteri da utilizzare per la lettura e la scrittura delle stringhe impostando la variabile membro *characterSet* .

Per ulteriori informazioni, fare riferimento a “Classe MQMessage” a pagina 1072.

Nota: Il metodo writeUTF() di MQMessage codifica automaticamente la lunghezza della stringa e i byte Unicode che contiene. Quando il messaggio viene letto da un altro programma Java (utilizzando readUTF()), questo è il metodo più semplice per inviare informazioni sulla stringa.

Miglioramento delle prestazioni dei messaggi non persistenti nelle classi WebSphere MQ per Java

Per migliorare le prestazioni durante l'esplorazione dei messaggi o l'utilizzo di messaggi non persistenti da un'applicazione client, è possibile utilizzare la *lettura anticipata*. Le applicazioni client che utilizzano MQGET o il consumo asincrono beneficeranno dei miglioramenti delle prestazioni durante l'esplorazione dei messaggi o l'utilizzo di messaggi non persistenti.

Per informazioni generali sulla funzione di lettura anticipata, consultare l'argomento correlato.

Nelle classi WebSphere MQ per Java, si utilizza CMQC.MQSO_READ_AHEAD e CMQC.MQSO_NO_READ_AHEAD di un oggetto MQQueue o MQTopic per stabilire se i consumer dei messaggi e i browser delle code possono utilizzare la lettura anticipata su tale oggetto.

Inserimento asincrono dei messaggi utilizzando le classi WebSphere MQ per Java

Per inserire un messaggio in modo asincrono, impostare MQPMO_ASYNC_RESPONSE.

I messaggi vengono inseriti nelle code o negli argomenti utilizzando il metodo put () della classe MQDestination. Per inserire un messaggio in modo asincrono, ossia consentendo il completamento dell'operazione senza attendere una risposta dal gestore code, è possibile impostare MQPMO_ASYNC_RESPONSE nel campo delle opzioni MQPutMessageOptions. Per determinare l'esito positivo o negativo delle operazioni di immissione asincrone, utilizzare la chiamata di stato MQQueueManager.getAsync.

Pubblica / sottoscrivi in classi WebSphere MQ per Java

Nelle classi WebSphere MQ per Java, l'argomento è rappresentato da una classe MQTopic e viene pubblicato utilizzando i metodi MQTopic.put().

Per informazioni generali sulla pubblicazione / sottoscrizione WebSphere MQ , consultare Introduzione a WebSphere MQ messaggistica di pubblicazione / sottoscrizione .

Gestione delle intestazioni dei messaggi WebSphere MQ con classi WebSphere MQ per Java

Vengono fornite classi Java che rappresentano diversi tipi di intestazione del messaggio. Sono fornite anche due classi helper.

Gli oggetti di intestazione sono descritti dall'interfaccia MQHeader, che fornisce metodi di uso generico per accedere ai campi di intestazione e per leggere e scrivere il contenuto del messaggio. Ogni tipo di intestazione ha la propria classe che implementa l'interfaccia MQHeader e aggiunge metodi getter e setter per i singoli campi. Ad esempio, il tipo di intestazione MQRFH2 è rappresentato dalla classe MQRFH2 ; il tipo di intestazione MQDLH dalla classe MQDLH e così via. Le classi di intestazione eseguono

automaticamente qualsiasi conversione di dati necessaria e possono leggere o scrivere i dati in qualsiasi CCSID (serie di caratteri o codifica numerica specificata).

Due classi helper, MQHeaderIterator e MQHeaderList, assistono nella lettura e decodifica (analisi) del contenuto dell'intestazione nei messaggi:

- La classe MQHeaderIterator funziona come un java.util.Iterator. Finché ci sono più intestazioni nel messaggio, il metodo next () restituisce true e il metodo nextHeader() o next () restituisce l'oggetto intestazione successivo.
- MQHeaderList funziona come un java.util.List. Come MQHeaderIterator, analizza il contenuto dell'intestazione, ma consente anche di ricercare intestazioni particolari, aggiungere nuove intestazione, rimuovere intestazioni esistenti, aggiornare i campi di intestazione e quindi scrivere il contenuto dell'intestazione in un messaggio. In alternativa, è possibile creare un MQHeaderListvuoto, quindi popolarlo con le istanze di intestazione e scriverlo in un messaggio una volta o ripetutamente.

Le classi MQHeaderIterator e MQHeaderList utilizzano le informazioni contenute in MQHeaderRegistry per sapere quali classi di intestazione WebSphere MQ sono associate a determinati tipi e formati di messaggi. Il MQHeaderRegistry è configurato con la conoscenza di tutti i formati e i tipi di intestazione WebSphere MQ correnti ed è possibile registrare anche i propri tipi di intestazione.

Il supporto viene fornito per le seguenti intestazioni Websphere MQ comunemente utilizzate

- MQRFH - Regole e intestazione di formattazione
- MQRFH2 - Come MQRFH, utilizzato per trasmettere messaggi a e da un broker di messaggi appartenente a WebSphere Message Broker. Utilizzato anche per contenere le proprietà del messaggio
- MQCIH - Bridge CICS
- MQDLH - Intestazione lettera non instradabile
- MQIIH - Intestazione informazioni IMS
- MQRMH - intestazione messaggio di riferimento
- MQSAPH - Intestazione SAP
- MQWIH - Intestazione informazioni di lavoro
- MQXQH - Intestazione Coda di trasmissione
- MQDH - Intestazione di distribuzione
- MQEPH - Intestazione PCF incapsulata

È anche possibile definire classi che rappresentano le proprie intestazioni.

Per utilizzare un MQHeaderIterator per richiamare un'intestazione RFH2 , impostare MQGMO_PROPERTIES_FORCE_MQRFH2 nelle opzioni GetMessageoppure impostare la proprietà della coda PROPCTL su FORCE.

Stampa di tutte le intestazioni in un messaggio utilizzando le classi WebSphere MQ per Java

In questo esempio, un'istanza di MQHeaderIterator analizza le intestazioni in un MQMessage ricevuto da una coda. Gli oggetti MQHeader restituiti dal metodo nextHeader() visualizzano la loro struttura e il loro contenuto quando viene richiamato il loro metodo 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);
}
```

Saltare le intestazioni in un messaggio utilizzando WebSphere MQ classes per Java

In questo esempio, il metodo skipHeaders() di MQHeaderIterator posiziona il cursore di lettura del messaggio immediatamente dopo l'ultima intestazione.

```
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 ();
```

Ricerca del codice di errore in un messaggio non instradabile utilizzando le classi di WebSphere MQ per Java

In questo esempio, il metodo read popola l'oggetto MQDLH leggendo dal messaggio. Dopo l'operazione di lettura, il cursore di lettura del messaggio viene posizionato immediatamente dopo il contenuto dell'intestazione MQDLH.

I messaggi nella coda di messaggi non instradabili del gestore code hanno come prefisso un'intestazione di messaggi non instradabili (MQDLH). Per decidere come gestire questi messaggi - ad esempio, per determinare se riprovare o eliminarli - un'applicazione di gestione dei messaggi non recapitabili deve esaminare il codice di errore contenuto in 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 ());
```

Tutte le classi di intestazione forniscono anche un costruttore di convenienza per inizializzarsi direttamente dal messaggio in un singolo passo. Quindi, il codice in questo esempio potrebbe essere semplificato come segue:

```
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 ());
```

Lettura e rimozione di MQDLH da un messaggio non instradabile utilizzando le classi WebSphere MQ per Java

In questo esempio, MQDLH viene utilizzato per eliminare l'intestazione da un messaggio non instradabile.

Un'applicazione di gestione dei messaggi non instradabili in genere reinoltra i messaggi che sono stati rifiutati se il relativo codice motivo indica un errore transitorio. Prima di inoltrare nuovamente il messaggio, è necessario rimuovere l'intestazione MQDLH.

Questo esempio esegue la seguente procedura (vedere i commenti nel codice di esempio):

1. MQHeaderList legge l'intero messaggio e ogni intestazione rilevata nel messaggio diventa un elemento nell'elenco.
2. I messaggi non instradabili contengono un MQDLH come prima intestazione, quindi è possibile trovarlo nella prima voce dell'elenco di intestazioni. MQDLH è già stato popolato dal messaggio quando viene creato MQHeaderList , quindi non è necessario richiamarne il metodo di lettura.
3. Il codice motivo viene estratto utilizzando il metodo getReason() fornito dalla classe MQDLH.
4. Il codice di errore è stato ispezionato e indica che è appropriato inoltrare nuovamente il messaggio. MQDLH viene rimosso utilizzando il metodo MQHeaderList remove ().

5. MQHeaderList scrive il contenuto rimanente in un nuovo oggetto messaggio. Il nuovo messaggio ora contiene tutto il contenuto del messaggio originale tranne MQDLH e può essere scritto in una coda. L'argomento **true** per il costruttore e per il metodo di scrittura indica che il corpo del messaggio deve essere contenuto in MQHeaderListe scritto di nuovo.
6. Il campo formato nel descrizione del messaggio del nuovo messaggio ora contiene il valore che era precedentemente nel campo formato MQDLH. I dati del messaggio corrispondono alla codifica numerica e CCSID impostati nel descrittore del messaggio.

```
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.
```

Stampa del contenuto di un messaggio utilizzando WebSphere MQ classes per Java

In questo esempio viene utilizzato MQHeaderList per stampare il contenuto di un messaggio, incluse le intestazioni.

L'output contiene una vista di tutto il contenuto dell'intestazione e del testo del messaggio. La classe MQHeaderList decodifica tutte le intestazioni in una sola volta, mentre MQHeaderIterator le espone una alla volta sotto il controllo dell'applicazione. È possibile utilizzare questa tecnica per fornire uno strumento di debug semplice durante la scrittura di applicazioni 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));
```

Questo esempio stampa anche i campi descrittore del messaggio, utilizzando la classe MQMD. Il metodo copyFrom() della classe com.ibm.mq.headers.MQMD popola l'oggetto intestazione dai campi del descrittore del messaggio di MQMessage piuttosto che leggendo il contenuto del messaggio.

```
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));
```

Ricerca di un tipo specifico di intestazione in un messaggio utilizzando WebSphere MQ classes per Java

Questo esempio utilizza il metodo indexOf(String) di MQHeaderList per trovare un'intestazione MQRFH2 in un messaggio, se presente.

```
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);
    ...
}
```

Analisi di un'intestazione MQRFH2 utilizzando WebSphere MQ classes per Java

Questo esempio mostra come accedere a un valore di campo noto in una cartella denominata utilizzando la classe MQRFH2 .

La classe MQRFH2 fornisce una serie di modi per accedere non solo ai campi nella parte fissa della struttura, ma anche al contenuto della cartella codificata XML contenuto nel campo NameValueData. Questo esempio mostra come accedere a un valore di campo noto in una cartella denominata - in questa istanza, il campo Rto nella cartella jms, che rappresenta il nome della coda di risposte in un messaggio JMS MQ .

```
MQRFH2 rfh = ...

String value = rfh.getStringFieldValue ("jms", "Rto");
```

Per scoprire il contenuto di MQRFH2 (invece di richiedere campi specifici direttamente), è possibile utilizzare il metodo getFolders per restituire un elenco di MQRFH2.Element, che rappresenta la struttura di una cartella che potrebbe contenere campi e altre cartelle. L'impostazione di un campo o di una cartella su null lo rimuove da MQRFH2. Quando si manipola il contenuto della cartella dati NameValuein questo modo, il campo StrucLength viene aggiornato automaticamente.

Lettura e scrittura di flussi di byte diversi dagli oggetti MQMessage utilizzando le classi WebSphere MQ per Java

Questi esempi utilizzano classi di intestazione per analizzare e manipolare il contenuto dell'intestazione WebSphere MQ quando l'origine dati non è un oggetto MQMessage.

È possibile utilizzare le classi di intestazione per analizzare e manipolare il contenuto dell'intestazione WebSphere MQ anche quando l'origine dati è diversa da un oggetto MQMessage. L'interfaccia MQHeader implementata da ogni classe di intestazione fornisce i metodi `int read (java.io.DataInput message, int encoding, int characterSet)` e `int write (java.io.DataOutput message, int encoding, int characterSet)`. La classe com.ibm.mq.MQMessage implementa le interfacce java.io.DataInput e java.io.DataOutput . Ciò significa che è possibile utilizzare i due metodi MQHeader per leggere e scrivere il contenuto di MQMessage, sostituendo la codifica e il CCSID specificati nel descrittore del messaggio. È utile per i messaggi che contengono una catena di intestazioni in codifiche differenti.

È inoltre possibile ottenere gli oggetti DataInput e DataOutput da altri flussi di dati, ad esempio flussi di file o socket o array di byte trasportati nei messaggi JMS. Le classi java.io.DataInputStream implementano DataInput e le classi java.io.DataOutputStream implementano DataOutput. Questo esempio legge il contenuto dell'intestazione WebSphere MQ da una matrice di byte:

```
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);
```

La riga che inizia con MQHeaderIterator potrebbe essere sostituita con

```
MQDLH dlh = new MQDLH (in, CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
// or any other header type
```

Questo esempio scrive in un array di byte utilizzando il flusso DataOutput:

```
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 ();
```

Quando si utilizzano gli stream in questo modo, fare attenzione a utilizzare i valori corretti per gli argomenti di codifica e characterSet . Durante la lettura delle intestazioni, specificare la codifica e il CCSID con cui il contenuto byte è stato originariamente scritto. Durante la scrittura delle intestazioni, specificare la codifica e il CCSID che si desidera produrre. La conversione dei dati viene eseguita automaticamente dalle classi di intestazione.

Creazione di classi per nuovi tipi di intestazione utilizzando le classi WebSphere MQ per Java

È possibile creare classi Java per tipi di intestazione non forniti con le classi WebSphere MQ per Java.

Per aggiungere una classe Java che rappresenta un nuovo tipo di intestazione che è possibile utilizzare allo stesso modo di qualsiasi classe di intestazione fornita con le classi WebSphere MQ per Java, creare una classe che implementi l'interfaccia MQHeader. L'approccio più semplice consiste nell'estendere la classe com.ibm.mq.headers.impl.Header . Questo esempio produce una classe completamente funzionale che rappresenta la struttura dell'intestazione MQTM. Non è necessario aggiungere singoli metodi getter e setter per ogni campo, ma si tratta di una convenienza utile per gli utenti della classe intestazione. I metodi getValue e setValue generici che utilizzano una stringa per il nome campo funzioneranno per tutti i campi definiti nel tipo di intestazione. I metodi di lettura, scrittura e dimensione ereditati consentono alle istanze del nuovo tipo di intestazione di essere lette e scritte e calcolano correttamente la dimensione dell'intestazione in base alla definizione del campo. La definizione del tipo viene creata solo una volta, tuttavia vengono create molte istanze di questa classe di intestazione.
Per rendere la nuova definizione di intestazione disponibile per la decodifica utilizzando le classi MQHeaderIterator o MQHeaderList , è necessario registrarla utilizzando MQHeaderRegistry. Notare, tuttavia, che la classe di intestazione MQTM è già fornita in questo pacchetto e registrata nel registro predefinito.

```
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.
}
```

Gestione dei messaggi PCF con classi WebSphere MQ per Java

Le classi Java sono fornite per creare e analizzare messaggi strutturati PCF e per facilitare l'invio di richieste PCF e la raccolta di risposte PCF.

Le classi PCFMessage & MQCFGR rappresentano schiere di strutture di parametri PCF. Forniscono metodi di convenienza per aggiungere e richiamare i parametri PCF.

Le strutture dei parametri PCF sono rappresentate dalle classi MQCFH, MQCFIN, MQCFIN64, MQCFST, MQCFBS, MQCFIL, MQCFIL64 MQCFSL e MQCFGR. Queste condividono le interfacce operative di base:

- Metodi per leggere e scrivere il contenuto del messaggio: read (), write () e size ()
- Metodi per manipolare i parametri getValue (), setValue (), getParameter () e altri
- Il metodo enumerator.nextParameter (), che analizza il contenuto PCF in un MQMessage

Il parametro di filtro PCF viene utilizzato nei comandi inquire per fornire una funzione di filtro. Esso è incapsulato nelle seguenti classi:

- MQCFIF - filtro numero intero
- MQCFSF - filtro stringa
- MQCFBF - filtro byte

Vengono fornite due classi di agent, PCFagent e PCFMessageAgent , per gestire la connessione a un gestore code, la coda del server dei comandi e una coda di risposta associata. PCFMessageAgent estende PCFAgent e deve essere normalmente utilizzato di preferenza. La classe PCFMessageAgent converte i messaggi MQMessage ricevuti e li restituisce al chiamante come un array PCFMessage. PCFAgent restituisce un array di messaggi MQMessage, che è necessario analizzare prima dell'utilizzo.

Gestione delle proprietà dei messaggi nelle classi WebSphere MQ per Java

Le chiamate di funzione per elaborare gli handle del messaggio non hanno equivalenti nelle classi WebSphere MQ per Java. Per impostare, restituire o eliminare le proprietà dell'handle del messaggio, utilizzare i metodi della classe MQMessage.

Per informazioni generali sulle proprietà dei messaggi, consultare “Nomi proprietà” a pagina 19.

Nelle classi WebSphere MQ per Java, l'accesso ai messaggi avviene mediante la classe MQMessage. Gli handle dei messaggi non vengono quindi forniti nell'ambiente Java e non esiste alcun equivalente alle chiamate della funzione WebSphere MQ MQCRTMH, MQDLTMH, MQMHBUF e MQBUFMH

Per impostare le proprietà della gestione messaggi nell'interfaccia procedurale, utilizzare la chiamata MQSETMP. Nelle classi WebSphere MQ per Java, utilizzare il metodo appropriato della classe MQMessage:

- Proprietà setBoolean
- Proprietà setByte
- Proprietà setBytes
- Proprietà setShort
- Proprietà setInt
- setInt2Property
- setInt4Property
- setInt8Property
- Proprietà setLong
- Proprietà setFloat
- Proprietà setDouble
- Proprietà setString
- Proprietà setObject

Talvolta si fa riferimento a tali metodi collettivamente come metodi *set*property* .

Per restituire il valore delle proprietà dell'handle del messaggio nell'interfaccia procedurale, utilizzare la chiamata MQINQMP. Nelle classi WebSphere MQ per Java, utilizzare il metodo appropriato della classe MQMessage:

- Proprietà getBoolean
- Proprietà getByte
- Proprietà getBytes
- Proprietà getShort
- Proprietà getInt
- getInt2Property
- getInt4Property
- getInt8Property
- Proprietà getLong
- Proprietà getFloat
- Proprietà getDouble
- Proprietà getString
- Proprietà getObject

Talvolta si fa riferimento a tali metodi collettivamente come ai metodi *get*property* .

Per eliminare il valore delle proprietà dell'handle del messaggio nell'interfaccia procedurale, utilizzare la chiamata MQDLTMP. In WebSphere MQ classes per Java, utilizzare il metodo deleteProperty della classe MQMessage.

Gestione degli errori nelle classi WebSphere MQ per Java

Gestire gli errori derivanti dalle classi WebSphere MQ per Java utilizzando i blocchi Java `try` e `catch` .

I metodi nell'interfaccia Java non restituiscono un codice di completamento e un codice motivo. Al contrario, generano un'eccezione ogni volta che il codice di completamento e il codice motivo risultanti da una chiamata WebSphere MQ non sono entrambi zero. Ciò semplifica la logica del programma in modo da non dover controllare i codici di ritorno dopo ogni chiamata a WebSphere MQ. È possibile decidere in quali punti del programma si desidera affrontare la possibilità di errore. In questi punti, puoi racchiudere il tuo codice con blocchi `try` e `catch` , come nel seguente esempio:

```
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() );
}
```

I codici motivo della chiamata WebSphere MQ riportati nelle eccezioni Java per z/OS sono documentati in Codici motivo per z/OS e Codici motivo per tutte le altre piattaforme.

Le eccezioni generate durante l'esecuzione di classi WebSphere MQ per l'applicazione Java vengono scritte anche nel log. Tuttavia, un'applicazione può richiamare il metodo MQException.logExclude() per impedire la registrazione delle eccezioni associate a uno specifico codice motivo. È possibile eseguire questa operazione in situazioni in cui si prevedono molte eccezioni associate a uno specifico codice di errore e non si desidera che il log venga riempito con tali eccezioni. Ad esempio, se l'applicazione tenta di richiamare un messaggio da una coda ogni volta che esegue l'iterazione di

un loop e, per la maggior parte di questi tentativi, si prevede che non vi sia alcun messaggio adatto nella coda, è possibile impedire la registrazione delle eccezioni associate al codice di errore MQRC_NO_MSG_AVAILABLE. Se un'applicazione ha precedentemente impedito la registrazione delle eccezioni associate a un codice motivo specifico, può consentire la registrazione di tali eccezioni richiamando il metodo MQException.logInclude().

A volte il codice di errore non trasmette tutti i dettagli associati all'errore. Per ogni eccezione generata, un'applicazione deve controllare l'eccezione collegata. L'eccezione collegata stessa potrebbe avere un'altra eccezione collegata, e quindi le eccezioni collegate formano una catena che riporta al problema sottostante originale. Un'eccezione collegata viene implementata utilizzando il meccanismo di eccezione concatenato della classe java.lang.Throwable , e un'applicazione ottiene un'eccezione collegata richiamando il metodo Throwable.getCause(). Da un'eccezione che è un'istanza MQException, MQException.getCause() richiama l'istanza sottostante di com.ibm.mq.jmqi.JmqiException e getCause da questa eccezione richiama l' java.lang.Exception sottostante che ha causato l'errore.

Per impostazione predefinita, la classe MQException trasmette automaticamente le eccezioni a System.err, che generalmente viene indirizzato alla console. Se si desidera interrompere la visualizzazione delle eccezioni sulla console, includere una riga nella propria applicazione per impostare MQException.log= null.

Richiamo e impostazione dei valori di attributo nelle classi WebSphere MQ per Java

I metodi getXXX() e setXXX() vengono forniti per molti attributi comuni. È possibile accedere ad altri utilizzando i metodi inquire () e set () generici.

Per molti degli attributi comuni, le classi MQManagedObject, MQDestination, MQQueue, MQTopic, MQProcess e MQQueueManager contengono metodi getXXX() e setXXX(). Questi metodi consentono di ottenere e impostare i loro valori di attributi. Tenere presente che per MQDestination, MQQueue e MQTopic, i metodi funzionano solo se si specificano gli indicatori inquire e set appropriati quando si apre l'oggetto.

Per gli attributi meno comuni, le classi MQQueueManager, MQDestination, MQQueue, MQTopic e MQProcess ereditano tutte da una classe denominata MQManagedObject. Questa classe definisce le interfacce inquire () e set ().

Quando si crea un nuovo oggetto gestore code utilizzando l'operatore *nuovo* , viene aperto automaticamente per l'interrogazione. Quando si utilizza il metodo accessProcess() per accedere a un oggetto processo, tale oggetto viene automaticamente aperto per l'interrogazione. Quando si utilizza il metodo accessQueue() per accedere ad un oggetto coda, tale oggetto *non* viene aperto automaticamente per le operazioni di interrogazione o di impostazione. Questo perché l'aggiunta automatica di queste opzioni può causare problemi con alcuni tipi di code remote. Per utilizzare i metodi inquire, set, getXXXe setXXX su una coda, è necessario specificare gli indicatori inquire e set appropriati nel parametro openOptions del metodo accessQueue(). Lo stesso vale per gli oggetti di destinazione e argomento.

I metodi inquire e set utilizzano tre parametri:

- array selettori
- array intAttrs
- Array charAttrs

Non sono necessari i parametri SelectorCount, IntAttrCount e CharAttrLength trovati in MQINQ, poiché la lunghezza di un array in Java è sempre nota. Il seguente esempio mostra come eseguire un'interrogazione su una coda:

```
// 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));
```

Programmi a più sottoprocessi in Java

Java Runtime Environment è intrinsecamente multithread. WebSphere Le classi MQ per Java consentono a un oggetto gestore code di essere condiviso da più thread, ma garantiscono che tutto l'accesso al gestore code di destinazione sia sincronizzato.

I programmi multithread sono difficili da evitare in Java. Considerare un programma semplice che si connette a un gestore code e apre una coda all'avvio. Il programma visualizza un singolo pulsante sullo schermo. Quando un utente fa clic su quel pulsante, il programma richiama un messaggio dalla coda.

Java Runtime Environment è intrinsecamente multithread. Pertanto, l'inizializzazione dell'applicazione avviene in un thread e il codice che viene eseguito in seguito alla pressione del pulsante viene eseguito in un thread separato (il thread dell'interfaccia utente).

Con il client MQI basato su C WebSphere MQ , ciò causerebbe un problema, poiché esistono limitazioni alla condivisione di handle da parte di più thread. WebSphere MQ classes for Java rilassa questo vincolo, consentendo a un oggetto gestore code (e ai relativi oggetti coda, argomento e processo associati) di essere condiviso da più thread.

L'implementazione delle classi WebSphere MQ per Java garantisce che, per una connessione particolare (istanza di oggettoMQQueueManager), tutti gli accessi al gestore code WebSphere MQ di destinazione siano sincronizzati. Un thread che desidera emettere una chiamata a un gestore code viene bloccato finché non vengono completate tutte le altre chiamate in corso per tale connessione. Se si richiede l'accesso simultaneo allo stesso gestore code da più thread all'interno del programma, creare un nuovo oggetto MQQueueManager per ogni thread che richiede l'accesso simultaneo. (Ciò equivale all'emissione di una chiamata MQCONN separata per ciascun thread.)

Nota: Le istanze della classe `com.ibm.mq.MQGetMessageOptions` non devono essere condivise tra i thread che richiedono messaggi contemporaneamente. Le istanze di questa classe vengono aggiornate con i dati durante la richiesta MQGET corrispondente, il che può causare conseguenze impreviste quando più thread operano contemporaneamente nella stessa istanza dell'oggetto.

Utilizzo delle uscite di canale nelle classi di WebSphere MQ per Java

Una panoramica su come utilizzare le uscite canale in un'applicazione utilizzando le classi WebSphere MQ per Java.

I seguenti argomenti descrivono come scrivere un'uscita di canale in Java, come assegnarla e come passare i dati ad essa. Descrivono quindi come utilizzare le uscite canale scritte in C e come utilizzare una sequenza di uscite canale.

L'applicazione deve disporre dell'autorizzazione di protezione corretta per caricare la classe di uscita canale.

Creazione di un'uscita di canale in classi WebSphere MQ per Java

È possibile fornire le proprie uscite canale definendo una classe Java che implementa un'interfaccia appropriata.

Per implementare un'uscita, definire una nuova classe Java che implementi l'interfaccia appropriata. Tre interfacce di uscita sono definite nel pacchetto com.ibm.mq.exits :

- WMQSendExit
- WMQReceiveExit
- WMQSecurityExit

Nota: Le uscite del canale sono supportate solo per connessioni client; non sono supportate per connessioni di bind. Non è possibile utilizzare un'uscita di canale Java all'esterno delle classi WebSphere MQ per Java, ad esempio se si utilizza un'applicazione client scritta in C.

Qualsiasi crittografia SSL definita per una connessione viene eseguita *dopo* che sono state richiamate le uscite di invio e di sicurezza. Analogamente, la decrittografia viene eseguita *prima che* vengano richiamate le uscite di sicurezza e di ricezione.

L'esempio riportato di seguito definisce una classe che implementa tutte e tre le interfacce:

```
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
  }
}
```

Ad ogni uscita viene passato un oggetto MQCXP e un oggetto MQCD. Questi oggetti rappresentano le strutture MQCXP e MQCD definite nell'interfaccia procedurale.

Qualsiasi classe di uscita scritta deve avere un costruttore. Può essere il costruttore predefinito o uno che utilizza un argomento stringa. Se richiede una stringa, i dati utente verranno passati nella classe di uscita quando vengono creati. Se la classe di uscita contiene sia un costruttore predefinito che un costruttore di argomento singolo, il costruttore di argomento singolo ha la priorità.

Per le uscite di invio e di sicurezza, il codice di uscita deve restituire i dati che si desidera inviare al server. Per un'uscita di ricezione, il codice di uscita deve restituire i dati modificati che devono essere interpretati da WebSphere MQ .

Il corpo di uscita più semplice possibile è:

```
{ return agentBuffer; }
```

Non chiudere il gestore code dall'interno di un'uscita canale.

Utilizzo delle classi di uscita canale esistenti

Nelle versioni di WebSphere MQ precedenti alla 7.0, è necessario implementare queste uscite utilizzando le interfacce MQSendExit, MQReceiveExite MQSecurityExit, come nel seguente esempio. Questo metodo rimane valido, ma il nuovo metodo è preferito per migliorare la funzionalità e le prestazioni.

```
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
  }
}
```

Assegnazione di un'uscita canale in IBM WebSphere MQ classes for Java

È possibile assegnare un'uscita di canale utilizzando IBM WebSphere MQ classes for Java.

Non esiste un equivalente diretto del canale IBM WebSphere MQ in IBM WebSphere MQ classes for Java. Le uscite dei canali vengono assegnate a un MQQueueManager. Ad esempio, dopo aver definito una classe che implementa l'interfaccia WMQSecurityExit , un'applicazione può utilizzare l'uscita di sicurezza in quattro modi:

- Assegnando un'istanza della classe al campo MQEnvironment.channelSecurityExit prima di creare un oggetto MQQueueManager
- Impostando il campo MQEnvironment.channelSecurityExit su una stringa che rappresenta la classe di uscita di sicurezza prima di creare un oggetto MQQueueManager
- Creando una coppia chiave / valore nell'hashtable delle proprietà passata a MQQueueManager con una chiave di CMQC.SECURITY_EXIT_PROPERTY
- Utilizzo di una tabella di definizione del canale client (CCDT)

Qualsiasi uscita assegnata impostando il campo MQEnvironment.channelSecurityExit su una stringa, creando una coppia chiave / valore nell'hashtable delle proprietà o utilizzando una CCDT, deve essere scritta con un costruttore predefinito. Un'uscita assegnata come istanza di una classe non necessita di un costruttore predefinito, a seconda dell'applicazione.

Un'applicazione può utilizzare un'uscita di invio o di ricezione in modo simile. Ad esempio, il seguente frammento di codice mostra come utilizzare le uscite di sicurezza, di invio e di ricezione implementate nella classe MyMQExits, definita in precedenza, utilizzando MQEnvironment:

```
MyMQExits myexits = new MyMQExits();
MQEnvironment.channelSecurityExit = myexits;
MQEnvironment.channelSendExit = myexits;
MQEnvironment.channelReceiveExit = myexits;
:
MQQueueManager jupiter = new MQQueueManager("JUPITER");
```

Se viene utilizzato più di un metodo per assegnare un'uscita canale, l'ordine di precedenza è il seguente:

1. Se l'URL di una CCDT viene passato a MQQueueManager, il contenuto della CCDT determina le uscite del canale da utilizzare e tutte le definizioni di uscita in MQEnvironment o nella tabella hash delle proprietà vengono ignorate.
2. Se non viene passato alcun URL CCDT, le definizioni di uscita da MQEnvironment e l'hashtable vengono unite
   - Se lo stesso tipo di uscita è definito sia in MQEnvironment che nella tabella hash, viene utilizzata la definizione nella tabella hash.
   - Se vengono specificati nuovi e vecchi tipi di uscita equivalenti (ad esempio, il campo sendExit , che può essere utilizzato solo per il tipo di uscita utilizzato nelle versioni di IBM WebSphere MQ

precedenti alla versione 7.0e il campo channelSendExit, che può essere utilizzato per qualsiasi uscita di invio), viene utilizzata la nuova uscita (channelSendExit) anziché la vecchia uscita.

Se è stata dichiarata un'uscita canale come stringa, è necessario abilitare IBM WebSphere MQ per individuare il programma di uscita canale. È possibile eseguire questa operazione in vari modi, a seconda dell'ambiente in cui l'applicazione è in esecuzione e della modalità di impacchettatura dei programmi di uscita del canale.

- Per un'applicazione in esecuzione in un application server, è necessario memorizzare i file nella directory mostrata in Tabella 88 a pagina 691 o impacchettati nei file JAR a cui fa riferimento **exitClasspath**.
- Per un'applicazione non in esecuzione in un application server, si applicano le seguenti regole:
  - Se le classi di uscita del canale sono compresse in file JAR separati, tali file JAR devono essere inclusi in **exitClasspath**.
  - Se le classi di uscita del canale non sono impacchettate in file JAR, i file di classe possono essere memorizzati nella directory mostrata in Tabella 88 a pagina 691 o in qualsiasi directory nel percorso di classe del sistema JVM o in **exitClasspath**.

La proprietà **exitClasspath** può essere specificata in quattro modi. In ordine di priorità, queste modalità sono le seguenti:

1. La proprietà di sistema com.ibm.mq.exitClasspath (definita sulla riga comandi utilizzando l'opzione `-D`)
2. La sezione exitPath del file `mqclient.ini`
3. Una voce hashtable con la chiave CMQC.EXIT_CLASSPATH_PROPERTY
4. La variabile MQEnvironment **exitClasspath**

Separare più percorsi utilizzando il carattere java.io.File.pathSeparator .

Tabella 88. La directory per i programmi di uscita del canale

Piattaforma	**Cartella**
AIX, HP-UX, Linuxe Solaris	`/var/mqm/exits` (programmi exit canale a 32 bit) `/var/mqm/exits64` (programmi exit canale a 64 bit)
Windows	*dir_dati_installazione*\exits
Nota: *install_data_dir* è la directory scelta per i file di dati IBM WebSphere MQ durante l'installazione. La directory predefinita è `C:\Program Files\IBM\WebSphere MQ`.	

Passaggio di dati alle uscite del canale in classi WebSphere MQ per Java

È possibile passare i dati alle uscite del canale e restituire i dati dalle uscite del canale all'applicazione.

Parametro agentBuffer

Per un'uscita di invio, il parametro *agentBuffer* contiene i dati che si sta per inviare. Per un'uscita di ricezione o di sicurezza, il parametro *agentBuffer* contiene i dati appena ricevuti. Non è necessario un parametro di lunghezza, in quanto l'espressione agentBuffer.limit () indica la lunghezza dell'array.

Per le uscite di invio e di sicurezza, il codice di uscita deve restituire i dati che si desidera inviare al server. Per un'uscita di ricezione, il codice di uscita deve restituire i dati modificati che devono essere interpretati da WebSphere MQ .

Il corpo di uscita più semplice possibile è:

```
{ return agentBuffer; }
```

Le uscite del canale vengono richiamate con un buffer che ha un array di supporto. Per prestazioni ottimali, l'uscita deve restituire un buffer con un array di backup.

Dati utente

Se un'applicazione si connette a un gestore code impostando channelSecurityExit, channelSendExit o channelReceiveExit, 32 byte di dati utente possono essere passati alla classe di uscita canale appropriata quando viene richiamata, utilizzando i dati channelSecurityExitUser, channelSendExitUserData o channelReceiveExitUserData. Questi dati sono disponibili per la classe di uscita del canale ma vengono aggiornati ogni volta che viene richiamata l'uscita. Tutte le modifiche apportate ai dati utente nell'uscita del canale andranno quindi perse. Se si desidera apportare modifiche persistenti ai dati in un'uscita del canale, utilizzare l'area MQCXP exitUser. I dati in questo campo vengono conservati tra i richiami dell'uscita.

Se l'applicazione imposta securityExit, sendExito receiveExit, non è possibile trasmettere dati utente a queste classi di uscita del canale.

Se un'applicazione utilizza una tabella di definizione del canale client (CCDT) per connettersi a un gestore code, tutti i dati utente specificati in una definizione del canale di connessione client vengono passati alle classi di uscita del canale quando vengono richiamati. Per ulteriori informazioni sull'utilizzo di una tabella di definizione del canale client, consultare “Utilizzo di una tabella di definizione di canale client con IBM WebSphere MQ classes for Java” a pagina 674.

Utilizzo di uscite di canale non scritte in Java con classi WebSphere MQ per Java

Come utilizzare i programmi di uscita del canale scritti in C da un'applicazione Java.

In WebSphere MQ Versione 7.0, è possibile specificare il nome di un programma di uscita del canale scritto in C come stringa passata ai campi channelSecurityExit, channelSendExit o channelReceiveExit nell'oggetto MQEnvironment o proprietà Hashtable. Tuttavia, non è possibile utilizzare un'uscita canale scritta in Java in un'applicazione scritta in un altro linguaggio.

Specificare il nome del programma di uscita nel formato `library(function)` e assicurarsi che l'ubicazione del programma di uscita sia inclusa nella variabile di ambiente del percorso.

Per informazioni su come scrivere un'uscita canale in C, consultare “Programmi di uscita canale per canali di messaggistica” a pagina 399.

Utilizzo di classi di uscita esterne

Nelle versioni di WebSphere MQ precedenti alla versione 7.0, sono state fornite tre classi per consentire l'utilizzo di uscite di canale scritte in linguaggi diversi da Java:

- MQExternalSecurityExit, che implementa l'interfaccia MQSecurityExit
- MQExternalSendExit, che implementa l'interfaccia MQSendExit
- MQExternalReceiveExit, che implementa l'interfaccia MQReceiveExit

L'utilizzo di queste classi rimane valido ma si preferisce il nuovo metodo.

Per utilizzare un'uscita di protezione non scritta in Java, un'applicazione ha prima dovuto creare un oggetto di uscita MQExternalSecurity. L'applicazione ha specificato, come parametri sul costruttore dell'uscita MQExternalSecurity, il nome della libreria contenente l'uscita di protezione, il nome del punto di ingresso per l'uscita di sicurezza e i dati utente da trasmettere all'uscita di sicurezza quando è stata richiamata. I programmi di uscita del canale non scritti in Java sono memorizzati nella directory mostrata in Tabella 88 a pagina 691.

Utilizzo di una sequenza di uscite di invio o ricezione del canale nelle classi WebSphere MQ per Java

Le classi WebSphere MQ per l'applicazione Java possono utilizzare una sequenza di uscite di invio o ricezione del canale eseguite in successione.

Per utilizzare una sequenza di uscite di invio, un'applicazione può creare un elenco o una stringa contenente le uscite di invio. Se viene utilizzato un elenco, ogni elemento dell'elenco può essere uno dei seguenti:

- Un'istanza di una classe definita dall'utente che implementa l'interfaccia WMQSendExit

- Un'istanza di una classe definita dall'utente che implementa l'interfaccia di MQSendExit (per un'uscita di invio scritta in Java)
- Un'istanza della classe di uscita MQExternalSend(per un'uscita di invio non scritta in Java)
- Un'istanza della classe Chain MQSendExit
- Un'istanza della classe String

Un elenco non può contenere un altro elenco.

L'applicazione può utilizzare una sequenza di uscite di ricezione in modo simile.

Se viene utilizzata una stringa, deve essere composta da una o più definizioni di uscita separate da virgole, ognuna delle quali può essere il nome di una classe Java o un programma C nel formato `library(function)`.

L'applicazione quindi assegna l'oggetto List o String al campo MQEnvironment.channelSendExit prima di creare un oggetto MQQueueManager .

Il contesto delle informazioni trasmesse alle uscite si trova esclusivamente all'interno del dominio delle uscite. Ad esempio, se un'uscita Java e un'uscita C sono concatenate, la presenza dell'uscita Java non ha alcun effetto sull'uscita C.

Utilizzo delle classi della catena di uscita

Nelle versioni di WebSphere MQ precedenti alla Versione 7.0, sono state fornite due classi per consentire sequenze di uscite:

- Catena MQSendExit, che implementa l'interfaccia MQSendExit
- Catena MQReceiveExit, che implementa l'interfaccia MQReceiveExit

L'utilizzo di queste classi rimane valido ma si preferisce il nuovo metodo. Utilizzando le classi WebSphere MQ per le interfacce Java, l'applicazione ha ancora una dipendenza su `com.ibm.mq.jar` Se viene utilizzata la nuova serie di interfacce nel pacchetto `com.ibm.mq.exits` , non vi è alcuna dipendenza su `com.ibm.mq.jar`.

Per utilizzare una sequenza di uscite di invio, un'applicazione ha creato un elenco di oggetti, dove ogni oggetto era uno dei seguenti:

- Un'istanza di una classe definita dall'utente che implementa l'interfaccia di MQSendExit (per un'uscita di invio scritta in Java)
- Un'istanza della classe di uscita MQExternalSend(per un'uscita di invio non scritta in Java)
- Un'istanza della classe Chain MQSendExit

L'applicazione ha creato un oggetto della catena MQSendExitinoltrando questo elenco di oggetti come parametro sul costruttore. L'applicazione avrebbe quindi assegnato l'oggetto catena MQSendExital campo MQEnvironment.sendExit prima di creare un oggetto MQQueueManager .

Compressione canale nelle classi WebSphere MQ per Java

La compressione dei dati che fluiscono su un canale può migliorare le prestazioni del canale e ridurre il traffico di rete. IBM WebSphere MQ classes for Java utilizzare la funzione di compressione integrata in IBM WebSphere MQ.

Utilizzando la funzione fornita con IBM WebSphere MQ, è possibile comprimere i dati che fluiscono sui canali di messaggi e MQI e, su entrambi i tipi di canale, è possibile comprimere i dati di intestazione e i dati di messaggio indipendentemente l'uno dall'altro. Per impostazione predefinita, non viene compresso alcun dato su un canale. Per una descrizione completa della compressione del canale, inclusa la modalità di implementazione in IBM WebSphere MQ, consultare Data compression (COMPMSG) e Header compression (COMPHDR).

Un'applicazione IBM WebSphere MQ classes for Java specifica le tecniche che possono essere utilizzate per comprimere i dati di intestazione o messaggio su una connessione client creando un oggetto java.util.Collection . Ogni tecnica di compressione è un oggetto intero nella raccolta e l'ordine in

cui l'applicazione aggiunge le tecniche di compressione alla raccolta è l'ordine in cui le tecniche di compressione vengono negoziate con il gestore code quando viene avviata la connessione client. L'applicazione può quindi assegnare la raccolta al campo hdrCompList, per i dati di intestazione, o al campo msgCompList, per i dati del messaggio, nella classe MQEnvironment. Quando l'applicazione è pronta, può avviare la connessione client creando un oggetto MQQueueManager .

I seguenti frammenti di codice illustreranno l'approccio descritto. Il primo frammento di codice mostra come implementare la compressione dei dati di intestazione:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(CMQXC.MQCOMPRESS_SYSTEM));
:
MQEnvironment.hdrCompList = headerComp;
:
MQQueueManager qMgr = new MQQueueManager(QM);
```

Il secondo frammento di codice mostra come implementare la compressione dei dati del messaggio:

```
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);
```

Nel secondo esempio, le tecniche di compressione vengono negoziate nell'ordine RLE, quindi ZLIBHIGH, all'avvio della connessione client. La tecnica di compressione selezionata non può essere modificata durante la durata dell'oggetto MQQueueManager .

Le tecniche di compressione per i dati dell'intestazione e del messaggio supportati sia dal client che dal gestore code su una connessione client vengono inoltrati a un'uscita del canale come raccolte nei campi Elenco hdrCompe Elenco msgCompdi un oggetto MQChannelDefinition . Le tecniche effettive attualmente utilizzate per comprimere i dati dell'intestazione e del messaggio su una connessione client vengono inoltrate a un'uscita del canale nei campi di compressione CurHdre CurMsgdi un oggetto MQChannelExit .

Se la compressione viene utilizzata su una connessione client, i dati vengono compressi prima che le uscite di invio del canale vengano elaborate ed estratte dopo l'elaborazione delle uscite di ricezione del canale. I dati passati per inviare e ricevere le uscite sono quindi in uno stato compresso.

Per ulteriori informazioni sulla specifica delle tecniche di compressione e sulle tecniche di compressione disponibili, consultare Class com.ibm.mq.MQEnvironment e Interface com.ibm.mq.MQC .

Condivisione di una connessione TCP/IP in IBM WebSphere MQ classes for Java

È possibile creare più istanze di un canale MQI per condividere una connessione TCP/IP singola.

In IBM WebSphere MQ classes for Java, utilizzare la variabile MQEnvironment.sharingConversations per controllare il numero di conversazioni che possono condividere una singola connessione TCP/IP.

L'attributo SHARECNV è un approccio ottimale alla condivisione delle connessioni. Pertanto, quando un valore SHARECNV maggiore di 0 viene utilizzato con IBM WebSphere MQ classes for Java , non è garantito che una nuova richiesta di connessione condivida sempre una connessione già stabilita.

Pool di connessioni nelle classi WebSphere MQ per Java

WebSphere MQ classes per Java consente di raggruppare in lotti le connessioni di riserva da riutilizzare.

WebSphere MQ classes per Java fornisce ulteriore supporto per le applicazioni che gestiscono più connessioni ai gestori code WebSphere MQ . Quando una connessione non è più necessaria, invece di eliminarla, può essere raggruppata e successivamente riutilizzata. Ciò può fornire un miglioramento sostanziale delle prestazioni per applicazioni e middleware che si connettono in modo seriale a gestori code arbitrari.

WebSphere MQ fornisce un pool di connessione predefinito. Le applicazioni possono attivare o disattivare questo pool di connessioni registrando e annullando la registrazione dei token tramite la classe MQEnvironment. Se il pool è attivo quando WebSphere MQ classes for Java crea un oggetto MQQueueManager , ricerca questo pool predefinito e riutilizza qualsiasi connessione appropriata. Quando si verifica una chiamata MQQueueManager.disconnect (), la connessione sottostante viene restituita al lotto.

In alternativa, le applicazioni possono creare un pool di connessioni MQSimpleConnectionManager per un utilizzo particolare. Quindi, l'applicazione può specificare tale pool durante la creazione di un oggetto MQQueueManager oppure passare tale pool a MQEnvironment per utilizzarlo come pool di connessioni predefinito.

Per evitare che le connessioni utilizzino una quantità eccessiva di risorse, è possibile limitare il numero totale di connessioni che un oggetto gestore MQSimpleConnectionpuò gestire ed è possibile limitare la dimensione del lotto connessioni. L'impostazione dei limiti è utile se vi sono richieste in conflitto per le connessioni all'interno di una JVM.

Per impostazione predefinita, il metodo getMaxConnections () restituisce il valore zero, che indica che non esiste alcun limite al numero di connessioni che l'oggetto MQSimpleConnectionManager può gestire. È possibile impostare un limite utilizzando il metodo setMaxConnections (). Se si imposta un limite e il limite viene raggiunto, una richiesta per un'altra connessione potrebbe causare la generazione di un'eccezione MQException, con un codice motivo di MQRC_MAX_CONNS_LIMIT_REACHED.

Controllo del pool di connessioni predefinito nelle classi WebSphere MQ per Java

Questo esempio mostra come utilizzare il pool di connessione predefinito.

Considerare la seguente applicazione di esempio, 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();
         }
      }
}
```

MQApp1 prende un elenco di gestori code locali dalla riga comandi, si connette a ciascuno di essi ed esegue alcune operazioni. Tuttavia, quando la riga comandi elenca lo stesso gestore code più volte, è più efficiente connettersi una sola volta e riutilizzare tale connessione più volte.

WebSphere MQ classes for Java fornisce un pool di connessioni predefinito che è possibile utilizzare a tale scopo. Per abilitare il pool, utilizzare uno dei metodi MQEnvironment.addConnectionPoolToken(). Per disabilitare il pool, utilizzare MQEnvironment.removeConnectionPoolToken().

La seguente applicazione di esempio, MQApp2, è funzionalmente identica a MQApp1, ma si connette solo una volta a ciascun gestore code.

```
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);

        }
}
```

La prima riga in grassetto attiva il pool di connessione predefinito registrando un oggetto MQPoolToken con MQEnvironment.

Il costruttore MQQueueManager ora ricerca in questo lotto una connessione appropriata e crea una connessione al gestore code solo se non riesce a trovare una connessione esistente. La chiamata qmgr.disconnect() restituisce la connessione al lotto per un riutilizzo successivo. Queste chiamate API sono le stesse dell'applicazione di esempio MQApp1.

La seconda riga evidenziata disattiva il pool di connessione predefinito, che elimina tutte le connessioni del gestore code memorizzate nel pool. Ciò è importante perché altrimenti l'applicazione terminerebbe con un numero di connessioni gestore code attive nel pool. Questa situazione potrebbe causare errori che vengono visualizzati nei log del gestore code.

Se un'applicazione utilizza una tabella di definizione di canale client (CCDT) per connettersi a un gestore code, il costruttore MQQueueManager ricerca prima nella tabella una definizione di canale di connessione client adatta. Se ne viene trovato uno, il costruttore ricerca il pool di connessione predefinito per una connessione che può essere utilizzata per il canale. Se il costruttore non riesce a trovare una connessione adatta nel pool, ricerca nella tabella di definizione del canale client la successiva definizione di canale di connessione client adatta e procede nel modo descritto in precedenza. Se il costruttore completa la ricerca della tabella di definizione del canale client e non riesce a trovare alcuna connessione adatta nel lotto, il costruttore avvia una seconda ricerca della tabella. Durante questa ricerca, il costruttore tenta di creare una nuova connessione per ogni definizione di canale di connessione client adatta e utilizza la prima connessione che riesce a creare.

Il pool di connessione predefinito memorizza un massimo di dieci connessioni inutilizzate e mantiene attive le connessioni inutilizzate per un massimo di cinque minuti. L'applicazione può modificarlo (per i dettagli, consultare “Fornitura di un pool di connessione differente nelle classi WebSphere MQ per Java” a pagina 697).

Invece di utilizzare MQEnvironment per fornire un MQPoolToken, l'applicazione può crearne uno proprio:

```
    MQPoolToken token=new MQPoolToken();
    MQEnvironment.addConnectionPoolToken(token);
```

Alcuni fornitori di applicazioni o middleware forniscono sottoclassi di MQPoolToken per trasmettere le informazioni a un pool di connessioni personalizzato. Possono essere creati e passati a addConnectionPoolToken() in questo modo in modo che ulteriori informazioni possano essere trasmesse al pool di connessioni.

Il pool di connessioni predefinito e più componenti nelle classi WebSphere MQ per Java

Questo esempio mostra come aggiungere o rimuovere MQPoolTokens da una serie statica di oggetti MQPoolToken registrati.

MQEnvironment contiene una serie statica di oggetti MQPoolToken registrati. Per aggiungere o rimuovere MQPoolTokens da questa serie, utilizzare i seguenti metodi:

- MQEnvironment.addConnectionPoolToken()
- MQEnvironment.removeConnectionPoolToken()

Un'applicazione potrebbe essere composta da molti componenti che esistono in modo indipendente e che eseguono il lavoro utilizzando un gestore code. In un'applicazione di questo tipo, ogni componente deve aggiungere un MQPoolToken alla serie MQEnvironment per la sua durata.

Ad esempio, l'applicazione di esempio MQApp3 crea dieci thread e li avvia. Ogni thread registra il proprio MQPoolToken, attende per un periodo di tempo e si connette al gestore code. Dopo la disconnessione del thread, rimuove il proprio MQPoolToken.

Il pool di connessione predefinito rimane attivo mentre è presente almeno un token nella serie di MQPoolTokens, quindi rimarrà attivo per la durata di questa applicazione. L'applicazione non ha bisogno di mantenere un oggetto master nel controllo generale dei thread.

```
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);
      }
}
```

Fornitura di un pool di connessione differente nelle classi WebSphere MQ per Java

Questo esempio mostra come utilizzare la classe **com.ibm.mq.MQSimpleConnectionManager** per fornire un pool di connessione differente.

Questa classe fornisce funzioni di base per il pool di connessioni e le applicazioni possono utilizzare questa classe per personalizzare il comportamento del pool.

Una volta creata, è possibile specificare un gestore MQSimpleConnectionsul costruttore MQQueueManager . Il gestore MQSimpleConnectiongestisce quindi la connessione alla base del MQQueueManagercreato. Se il gestore MQSimpleConnectioncontiene una connessione in pool adatta, tale connessione viene riutilizzata e restituita al gestore MQSimpleConnectiondopo una chiamata MQQueueManager.disconnect ().

Il seguente frammento di codice dimostra questo comportamento:

```
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);
```

La connessione forgiata durante il primo costruttore MQQueueManager viene memorizzata in myConnMan dopo la chiamata qmgr.disconnect(). La connessione viene quindi riutilizzata durante la seconda chiamata al costruttore MQQueueManager .

La seconda riga abilita il gestore MQSimpleConnection. L'ultima riga disabilita MQSimpleConnectionManager, eliminando tutte le connessioni presenti nel lotto. Un gestore MQSimpleConnectionè, per impostazione predefinita, in MODE_AUTO, descritto più avanti in questa sezione.

Un gestore MQSimpleConnectionassegna le connessioni in base all'utilizzo più recente e distrugge le connessioni in base all'utilizzo meno recente. Per impostazione predefinita, una connessione viene eliminata se non è stata utilizzata per cinque minuti o se nel pool sono presenti più di dieci connessioni non utilizzate. È possibile modificare questi valori richiamando MQSimpleConnectionManager.setTimeout().

È inoltre possibile impostare un gestore MQSimpleConnectionda utilizzare come pool di connessioni predefinito, da utilizzare quando non viene fornito un gestore connessioni sul costruttore di MQQueueManager .

La seguente applicazione lo dimostra:

```
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);
      }
}
```

Le righe in grassetto creano e configurano un oggetto MQSimpleConnectionManager. La configurazione effettua quanto segue:

- Termina le connessioni non utilizzate per un'ora
- Limita il numero di connessioni gestite da myConnMan a 75
- Limita il numero di connessioni non utilizzate nel pool a 50
- Imposta MODE_AUTO, che è il valore predefinito. Ciò significa che il pool è attivo solo se è il gestore connessioni predefinito e che è presente almeno un token nella serie di MQPoolTokens detenuti da MQEnvironment.

Il nuovo gestore MQSimpleConnectionviene quindi impostato come gestore connessioni predefinito.

Nell'ultima riga, l'applicazione richiama MQApp3.main(). Viene eseguito un numero di thread, dove ogni thread utilizza WebSphere MQ in modo indipendente. Questi thread utilizzano myConnMan quando creano connessioni.

Fornitura delle proprie classi ConnectionManager per WebSphere MQ per Java

WebSphere MQ classes per Java fornisce un'implementazione parziale dell'architettura del connettore Java EE , consentendo l'utilizzo delle implementazioni di javax.resource.spi.ConnectionManager .

I provider middleware e delle applicazioni possono fornire implementazioni alternative dei pool di connessione. WebSphere MQ classes per Java fornisce un'implementazione parziale di Java EE Connector Architecture. Le implementazioni di **javax.resource.spi.ConnectionManager** possono essere utilizzate come Connection Manager predefinito o specificate nel costruttore MQQueueManager .

WebSphere MQ classes per Java è conforme al contratto di gestione della connessione di Java EE Connector Architecture. Leggere questa sezione insieme al contratto di gestione della connessione di Java EE Connector Architecture (consultare il sito Web Java di Sun all'indirizzo https://java.sun.com).

L'interfaccia ConnectionManager definisce solo un metodo:

```
package javax.resource.spi;
public interface ConnectionManager {
      Object allocateConnection(ManagedConnectionFactory mcf,
                                ConnectionRequestInfo cxRequestInfo);
}
```

Il costruttore MQQueueManager richiama allocateConnection sul ConnectionManagerappropriato. Trasmette le implementazioni appropriate di ManagedConnectionFactory e ConnectionRequestInfo come parametri per descrivere la connessione richiesta.

Il ConnectionManager ricerca nel proprio pool un oggetto javax.resource.spi.ManagedConnection creato con oggetti ManagedConnectionFactory e ConnectionRequestInfo identici. Se ConnectionManager trova oggetti ManagedConnection adatti, crea un java.util.Set che contiene il candidato ManagedConnections. Quindi, il ConnectionManager richiama quanto segue:

```
ManagedConnection mc=mcf.matchManagedConnections(connectionSet, subject,
cxRequestInfo);
```

L'implementazione WebSphere MQ di ManagedConnectionFactory ignora il parametro dell'oggetto. Questo metodo seleziona e restituisce una ManagedConnection adatta dalla serie o restituisce null se non trova una ManagedConnectionadatta. Se non esiste una ManagedConnection adatta nel pool, ConnectionManager può crearne una utilizzando:

```
ManagedConnection mc=mcf.createManagedConnection(subject, cxRequestInfo);
```

Di nuovo, il parametro oggetto viene ignorato. Questo metodo si connette a un gestore code WebSphere MQ e restituisce un'implementazione di javax.resource.spi.ManagedConnection che rappresenta la nuova connessione forgiata. Una volta che ConnectionManager ha ottenuto una ManagedConnection (dal pool o appena creato), crea un handle di connessione utilizzando:

```
Object handle=mc.getConnection(subject, cxRequestInfo);
```

Questo handle di connessione può essere restituito da allocateConnection().

Un ConnectionManager deve registrare un interesse per ManagedConnection tramite:

```
mc.addConnectionEventListener()
```

Il listener ConnectionEventriceve una notifica se si verifica un errore grave sulla connessione o quando viene richiamato MQQueueManager.disconnect (). Quando si chiama MQQueueManager.disconnect (), il listener ConnectionEventpuò effettuare una delle seguenti operazioni:

- Reimpostare ManagedConnection utilizzando la chiamata mc.cleanup(), quindi restituire ManagedConnection al lotto
- Eliminare la ManagedConnection utilizzando la chiamata mc.destroy()

Se ConnectionManager è il ConnectionManagerpredefinito, può anche registrare un interesse nello stato della serie gestita da MQEnvironment di MQPoolTokens. A tale scopo, creare prima un oggetto MQPoolServices , quindi registrare un oggettoEventListener di MQPoolServicescon l'oggetto MQPoolServices :

```
MQPoolServices mqps=new MQPoolServices();
mqps.addMQPoolServicesEventListener(listener);
```

Il listener riceve una notifica quando un MQPoolToken viene aggiunto o rimosso dalla serie o quando il ConnectionManager predefinito viene modificato. L'oggetto MQPoolServices fornisce inoltre un modo per interrogare la dimensione corrente della serie di MQPoolTokens.

Coordinazione JTA/JDBC utilizzando le classi WebSphere MQ per Java

WebSphere Le classi MQ per Java supportano il metodo MQQueueManager.begin (), che consente a WebSphere MQ di agire come coordinatore per un database che fornisce un driver compatibile JDBC tipo 2 o JDBC tipo 4.

Questo supporto non è disponibile su tutte le piattaforme. Per verificare quali piattaforme supportano il coordinamento JDBC , consultare https://www.ibm.com/software/integration/wmq/requirements/.

Per utilizzare il supporto XA - JTA, è necessario utilizzare la libreria di switch JTA speciale. Il metodo per utilizzare questa libreria varia a seconda che si stia utilizzando Windows o una delle altre piattaforme.

Configurazione del coordinamento JTA/JDBC su Windows

La libreria XA viene fornita come DLL con un nome del formato `jdbcxxx.dll`.

V 7.5.0.7 `jdbcora12.dll` fornito fornisce la compatibilità con Oracle 12C, per un'installazione del server IBM WebSphere MQ Windows .

Su sistemi Windows , la libreria XA viene fornita come una DLL completa. Il nome di questa DLL è `jdbcxxx.dll` dove `xxx` indica il database per cui è stata compilata la libreria switch. Questa libreria si trova nella directory `java\lib\jdbc` o `java\lib64\jdbc` delle proprie classi IBM WebSphere MQ per l'installazione di Java . È necessario dichiarare la libreria XA, descritta anche come file di caricamento switch, al gestore code. Utilizzare IBM WebSphere MQ Explorer. Specificare i dettagli del file di caricamento switch nel pannello delle proprietà del gestore code, nel gestore risorse XA. È necessario fornire solo il nome della libreria. Ad esempio:

Per un database Db2 impostare il campo SwitchFile su `dbcdb2`

Per un database Oracle impostare il campo SwitchFile su `jdbcora`

Configurazione del coordinamento JTA/JDBC su piattaforme diverse da Windows

Vengono forniti i file oggetto. Collegare quello appropriato utilizzando il makefile fornito e dichiararlo al gestore code utilizzando il file di configurazione.

Per ogni sistema di gestione del database, WebSphere MQ fornisce due file oggetto. È necessario collegare un file oggetto per creare una libreria di switch a 32 bit e collegare l'altro file oggetto per creare una libreria di switch a 64 bit. Per DB2, il nome di ciascun file di oggetto è jdbcdb2.o e, per Oracle, il nome di ogni file di oggetto è jdbcora.o.

È necessario collegare ciascun file oggetto utilizzando il makefile appropriato fornito con WebSphere MQ. Una libreria switch richiede altre librerie, che potrebbero essere memorizzate in ubicazioni differenti su sistemi differenti. Tuttavia, una libreria switch non può utilizzare la variabile di ambiente del percorso libreria per individuare queste librerie poiché la libreria switch viene caricata dal gestore code, che viene eseguito in un ambiente setuid. Il makefile fornito garantisce quindi che una libreria switch contenga i nomi percorso completi di tali librerie.

Per creare una libreria switch, immettere un comando **make** con il seguente formato. Per creare una libreria switch a 32 bit, immettere il comando nella directory /java/lib/jdbc dell'installazione di WebSphere MQ . Per creare una libreria switch a 64 bit, immettere il comando nella directory /java/lib64/jdbc .

```
make DBMS
```

dove *DBMS* è il sistema di gestione database per cui si sta creando la libreria switch. I valori validi sono db2 per DB2 e `oracle` per Oracle.

Di seguito è riportato un esempio di comando **make** :

```
make db2
```

Tenere presente i seguenti aspetti:

- Per eseguire applicazioni a 32 bit, è necessario creare una libreria di switch a 32 bit e a 64 bit per ciascun sistema di gestione database che si sta utilizzando. Per eseguire applicazioni a 64 bit, è

necessario creare solo una libreria di switch a 64 bit. Per DB2, il nome di ciascuna libreria di switch è jdbcdb2 e, per Oracle, il nome di ogni libreria di switch è jdbcora. I makefile garantiscono che le librerie di switch a 32 bit e a 64 bit siano archiviate in directory WebSphere MQ differenti. Una libreria switch a 32 bit è memorizzata nella directory /java/lib/jdbc e una libreria switch a 64 bit è memorizzata nella directory /java/lib64/jdbc .

- Poiché è possibile installare Oracle ovunque su un sistema, i makefile utilizzano la variabile di ambiente ORACLE_HOME per individuare dove è installato Oracle .

Dopo aver creato le librerie di switch per DB2, Oracleo entrambe, è necessario dichiararle al gestore code. Se il file di configurazione del gestore code (qm.ini) contiene già stanze XAResourceManager per database DB2 o Oracle , è necessario sostituire la voce SwitchFile in ogni stanza con una delle seguenti:

Per un database DB2

```
SwitchFile=jdbcdb2
```

Per un database Oracle

```
SwitchFile=jdbcora
```

Non specificare il nome percorso completo della libreria di commutazione a 32 bit o a 64 bit. Specificare solo il nome della libreria.

Se il file di configurazione del gestore code non contiene già le stanze XAResourceManager per i database DB2 o Oracle o se si desidera aggiungere ulteriori stanze XAResourceManager , consultare Amministrazione per informazioni su come creare una stanza XAResourceManager . Tuttavia, ogni voce SwitchFile in una nuova stanza XAResourceManager deve essere esattamente come descritto in precedenza per un database DB2 o Oracle . È necessario includere anche la voce `ThreadOfControl=PROCESS`.

Dopo aver aggiornato il file di configurazione del gestore code e essersi assicurati che siano state impostate tutte le variabili di ambiente del database appropriate, è possibile riavviare il gestore code.

Utilizzo del coordinamento JTA/JDBC

Codificare le chiamate API come nell'esempio fornito.

La sequenza di base delle chiamate API per un'applicazione utente è:

```
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` nella chiamata getJDBCConnection è un'implementazione specifica del database dell'interfaccia XADataSource , che definisce i dettagli del database a cui connettersi. Consultare la documentazione per il proprio database per determinare come creare un oggetto XADataSource appropriato da passare a getJDBCConnection.

È inoltre necessario aggiornare il percorso di classe con i file jar specifici del database appropriati per eseguire il lavoro JDBC .

Se è necessario connettersi a più database, è necessario richiamare getJDBCConnection più volte per eseguire la transazione su diverse connessioni.

Esistono due forme di getJDBCConnection, che riflettono le due forme di 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
```

Questi metodi dichiarano l'eccezione nelle relative clausole throws per evitare problemi con il programma di verifica JVM per i clienti che non utilizzano le funzioni JTA. L'eccezione effettiva generata è javax.transaction.xa.XAException che richiede l'aggiunta del file jta.jar al percorso di classe per i programmi che non lo richiedevano in precedenza.

Per utilizzare il supporto JTA/JDBC , è necessario includere la seguente istruzione nell'applicazione:

```
MQEnvironment.properties.put(CMQC.THREAD_AFFINITY_PROPERTY, new Boolean(true));
```

Problemi noti e limitazioni con il coordinamento di JTA/JDBC

Esistono alcuni problemi e limitazioni del supporto JTA/JDBC , alcuni in base al sistema di gestione del database in uso.

Poiché questo supporto effettua chiamate ai driver JDBC , l'implementazione di tali driver JDBC può avere un effetto significativo sul funzionamento del sistema. In particolare, i driver JDBC verificati si comportano in modo diverso quando il database viene chiuso mentre un'applicazione è in esecuzione. **Sempre** evitare la chiusura improvvisa di un database mentre vi sono applicazioni che detengono connessioni aperte.

Più stanze XAResourceManager

L'utilizzo di più di una stanza XAResourceManager in un file di configurazione del gestore code, qm.ini, non è supportato. Qualsiasi stanza XAResourceManager diversa dalla prima viene ignorata.

DB2

Talvolta DB2 restituisce un errore SQL0805N . Questo problema può essere risolto con il seguente comando CLP:

```
DB2 bind @db2cli.lst blocking all grant public
```

Per ulteriori informazioni, fare riferimento alla documentazione di DB2 .

La stanza XAResourceManager deve essere configurata per utilizzare ThreadOfControl = PROCESS. Per DB2 versione 8.1 e versioni successive non corrisponde al thread predefinito dell'impostazione di controllo per DB2, quindi toc=p deve essere specificato in XA Open String. Un esempio di stanza XAResourceManager per il coordinamento DB2 con JTA/JDBC è il seguente:

```
XAResourceManager:
     Name=jdbcdb2
     SwitchFile=jdbcdb2
     XAOpenString=uid=userid,db=dbalias,pwd=password,toc=p
     ThreadOfControl=PROCESS
```

Ciò non impedisce alle applicazioni Java che utilizzano la coordinazione JTA/JDBC di essere multithread.

Oracle

Richiamando il metodo JDBC Connection.close() dopo che MQQueueManager.disconnect () ha generato una SQLException. Richiamare Connection.close() prima di MQQueueManager.disconnect () oppure omettere la chiamata a Connection.close().

Supporto SSL (Secure Sockets Layer) nelle classi WebSphere MQ per Java

Le classi WebSphere MQ per le applicazioni client Java supportano la codifica SSL (Secure Sockets Layer). È necessario un fornitore JSSE per utilizzare la codifica SSL.

WebSphere MQ per applicazioni client Java che utilizzano TRANSPORT (CLIENT) supportano la codifica SSL (Secure Sockets Layer). SSL fornisce la codifica di comunicazione, l'autenticazione e l'integrità del messaggio. Generalmente viene utilizzato per proteggere le comunicazioni tra due peer su Internet o all'interno di una intranet.

WebSphere MQ classes per Java utilizza JSSE (Java Secure Socket Extension) per gestire la crittografia SSL e quindi richiede un fornitore JSSE. Le JVM JSE v1.4 hanno un provider JSSE integrato. I dettagli su come gestire e memorizzare i certificati possono variare da fornitore a fornitore. Per informazioni, fare riferimento alla documentazione del provider JSSE.

Questa sezione presuppone che il provider JSSE sia installato e configurato correttamente e che i certificati appropriati siano stati installati e resi disponibili per il proprio provider JSSE.

Se le classi WebSphere MQ per l'applicazione client Java utilizzano una CCDT (client channel definition table) per connettersi a un gestore code, consultare “Utilizzo di una tabella di definizione di canale client con IBM WebSphere MQ classes for Java” a pagina 674.

Abilitazione di SSL in IBM WebSphere MQ classes for Java

Per abilitare SSL, specificare una CipherSuite. Esistono due metodi per specificare una CipherSuite.

SSL è supportato solo per le connessioni client. Per abilitare SSL, è necessario specificare la CipherSuite da utilizzare durante la comunicazione con il gestore code e questa CipherSuite deve corrispondere alla CipherSpec impostata nel canale di destinazione. Inoltre, la CipherSuite denominata deve essere supportata dal provider JSSE. Tuttavia, le CipherSuites sono distinte da CipherSpecs e hanno nomi differenti. “SSL CipherSpecs e CipherSuites in classi WebSphere MQ per Java” a pagina 707 contiene una tabella che associa i CipherSpecs supportati da IBM WebSphere MQ ai CipherSuites equivalenti noti a JSSE.

Per abilitare SSL, specificare CipherSuite utilizzando la variabile del membro statico della suite sslCipherdi MQEnvironment. Il seguente esempio si collega a un canale SVRCONN denominato SECURE.SVRCONN.CHANNEL, che è stato impostato per richiedere SSL con un CipherSpec di 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");
```

Sebbene il canale abbia una CipherSpec di RC4_MD5_EXPORT, l'applicazione Java deve specificare una CipherSuite di SSL_RSA_EXPORT_WITH_RC4_40_MD5. Consultare “SSL CipherSpecs e CipherSuites in classi WebSphere MQ per Java” a pagina 707 per un elenco di associazioni tra CipherSpecs e CipherSuites.

Un'applicazione può anche specificare una CipherSuite impostando la proprietà di ambiente CMQC.SSL_CIPHER_SUITE_PROPERTY.

In alternativa, utilizzare CCDT (Client Channel Definition Table). Per ulteriori informazioni, vedi “Utilizzo di una tabella di definizione di canale client con IBM WebSphere MQ classes for Java” a pagina 674

Se si richiede una connessione client per utilizzare una CipherSuite supportata dal provider IBM Java JSSE FIPS (IBMJSSEFIPS), un'applicazione può impostare il campo obbligatorio sslFipsnella classe MQEnvironment su `true`. In alternativa, l'applicazione può impostare la proprietà di ambiente CMQC.SSL_FIPS_REQUIRED_PROPERTY. Il valore predefinito è `false`, il che significa che una connessione client può utilizzare qualsiasi CipherSuite supportato da IBM WebSphere MQ.

Se un'applicazione utilizza più di una connessione client, il valore del campo sslFipsObbligatorio utilizzato quando l'applicazione crea la prima connessione client determina il valore utilizzato quando l'applicazione crea una connessione client successiva. Pertanto, quando l'applicazione crea una successiva connessione client, il valore del campo sslFipsObbligatorio viene ignorato. È necessario riavviare l'applicazione se si desidera utilizzare un valore diverso per il campo sslFipsObbligatorio.

Per connettersi correttamente utilizzando SSL, il truststore JSSE deve essere configurato con certificati root dell'autorità di certificazione da cui è possibile autenticare il certificato presentato dal gestore code. Allo stesso modo, se SSLClientAuth sul canale SVRCONN è stato impostato su MQSSL_CLIENT_AUTH_REQUIRED, il keystore JSSE deve contenere un certificato di identificazione ritenuto attendibile dal gestore code.

Riferimenti correlati

Federal Information Processing Standards (FIPS) per UNIX, Linux e Windows

Utilizzo del DN del gestore code in IBM WebSphere MQ classes for Java

Il gestore code si identifica utilizzando un certificato SSL, che contiene un DN (distinguished name). Un'applicazione client IBM WebSphere MQ classes for Java può utilizzare questo DN per garantire che stia comunicando con il gestore code corretto.

Un modello DN viene specificato utilizzando la variabile nome sslPeerdi MQEnvironment. Ad esempio, l'impostazione:

```
MQEnvironment.sslPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

consente la riuscita della connessione solo se il gestore code presenta un certificato con un nome comune che inizia con QMGR., e almeno due nomi di unità organizzative, il primo dei quali deve essere IBM e il secondo WebSphere.

Se è impostato il nome sslPeer, le connessioni hanno esito positivo solo se sono impostate su un modello valido e il gestore code presenta un certificato corrispondente.

Un'applicazione può anche specificare il DN (Distinguished Name) del gestore code impostando la proprietà di ambiente CMQC.SSL_PEER_NAME_PROPERTY. Per ulteriori informazioni sui DN (distinguished name), consultare DN (distinguished name).

Utilizzo degli elenchi di revoca certificati in IBM WebSphere MQ classes for Java

Specificare gli elenchi di revoca certificati da utilizzare tramite la classe java.security.cert.CertStore . IBM WebSphere MQ classes for Java quindi controlla i certificati rispetto al CRL specificato.

Un CRL (Certificate Revocation List) è una serie di certificati che sono stati revocati, dall'autorità di certificazione emittente o dall'organizzazione locale. I CRL sono generalmente ospitati su server LDAP. Con Java 2 v1.4, un server CRL può essere specificato in fase di connessione e il certificato presentato dal gestore code viene controllato rispetto al CRL prima che la connessione sia consentita. Per ulteriori informazioni sui CRL (Certificate Revocation List) e IBM WebSphere MQ, consultare Utilizzo dei CRL (Certificate Revocation List) e degli ARL (Authority Revocation List) e Accesso a CRL e ARL con le classi WebSphere MQ per Java e WebSphere MQ per JMS.

Nota: Per utilizzare correttamente un CertStore con un CRL ospitato su un server LDAP, assicurati che il tuo SDK (Software Development Kit) Java sia compatibile con il CRL. Alcuni SDK richiedono che il CRL sia conforme a RFC 2587, che definisce uno schema per LDAP v2. La maggior parte dei server LDAP v3 utilizza invece RFC 2256.

I CRL da utilizzare vengono specificati tramite la classe java.security.cert.CertStore . Fare riferimento alla documentazione su questa classe per i dettagli completi su come ottenere le istanze di CertStore. Per creare un CertStore basato su un server LDAP, creare prima un'istanza di parametri LDAPCertStore, inizializzata con le impostazioni server e porta da utilizzare. Ad esempio:

```
import java.security.cert.*;
CertStoreParameters csp = new LDAPCertStoreParameters("crl_server", 389);
```

Dopo aver creato un'istanza dei parametri CertStore, utilizzare il costruttore statico su CertStore per creare un CertStore di tipo LDAP:

```
CertStore cs = CertStore.getInstance("LDAP", csp);
```

Sono supportati anche altri tipi CertStore (ad esempio, Collection). Di solito ci sono diversi server CRL configurati con informazioni CRL identiche per fornire ridondanza. Quando si dispone di un oggetto CertStore per ciascuno di questi server CRL, collocarli tutti in una raccolta adatta. Il seguente esempio mostra gli oggetti CertStore inseriti in un ArrayList:

```
import java.util.ArrayList;
Collection crls = new ArrayList();
crls.add(cs);
```

Questa raccolta può essere impostata nella variabile statica MQEnvironment, sslCertStores, prima della connessione per abilitare il controllo CRL:

```
MQEnvironment.sslCertStores = crls;
```

Il certificato presentato dal gestore code durante l'impostazione di una connessione viene convalidato nel modo seguente:

1. Il primo oggetto CertStore nella raccolta identificata da sslCertStores viene utilizzato per identificare un server CRL.
2. È stato effettuato un tentativo di contattare il server CRL.
3. Se il tentativo ha esito positivo, il server viene ricercato per una corrispondenza per il certificato.
   a. Se viene rilevato che il certificato è stato revocato, il processo di ricerca è stato completato e la richiesta di connessione ha esito negativo con codice motivo MQRC_SSL_CERTIFICATE_REVOKED.
   b. Se il certificato non viene trovato, il processo di ricerca è stato completato e la connessione può continuare.
4. Se il tentativo di contattare il server non ha esito positivo, il successivo oggetto CertStore viene utilizzato per identificare un server CRL e il processo si ripete dal passo 2.

   Se si tratta dell'ultima CertStore nella raccolta o se la raccolta non contiene oggetti CertStore , il processo di ricerca non è riuscito e la richiesta di connessione ha esito negativo con codice motivo MQRC_SSL_CERT_STORE_ERROR.

L'oggetto Raccolta determina l'ordine in cui vengono utilizzati i CertStores .

La raccolta di CertStores può essere impostata anche utilizzando CMQC.SSL_CERT_STORE_PROPERTY. Per comodità, questa proprietà consente anche di specificare un singolo CertStore senza essere membro di una raccolta.

Se sslCertStores è impostato su null, non viene eseguito alcun controllo CRL. Questa proprietà viene ignorata se la suite sslCiphernon è impostata.

Rinegoziazione della chiave segreta nelle classi WebSphere MQ per Java

Le classi WebSphere MQ per l'applicazione client Java possono controllare quando la chiave segreta utilizzata per la codifica su una connessione client viene rinegoziata, in termini di numero totale di byte inviati e ricevuti.

L'applicazione può eseguire questa operazione in uno dei modi seguenti: se l'applicazione utilizza più di uno di questi modi, si applicano le solite regole di precedenza.

- Impostando il campo sslResetCount nella classe MQEnvironment.
- Impostando la proprietà di ambiente MQC.SSL_RESET_COUNT_PROPERTY in un oggetto Hashtable. L'applicazione, quindi, assegna l'hashtable al campo `properties` nella classe MQEnvironment o passa l'hashtable a un oggetto MQQueueManager sul relativo costruttore.

Il valore del campo sslReseto della proprietà di ambiente MQC.SSL_RESET_COUNT_PROPERTY rappresenta il numero totale di byte inviati e ricevuti dalle classi WebSphere MQ per il codice client Java prima che la chiave segreta venga rinegoziata. Il numero di byte inviati è il numero prima della codifica e il numero di byte ricevuti è il numero dopo la decodifica. Il numero di byte include anche le informazioni di controllo inviate e ricevute dal client WebSphere MQ .

Se il conteggio di reimpostazione è zero, che è il valore predefinito, la chiave segreta non viene mai rinegoziata. Il conteggio di reimpostazioni viene ignorato se non viene specificato alcun CipherSuite .

Fornitura di un SSLSocketFactory personalizzato in IBM WebSphere MQ classes for Java

Se si utilizza un factory di socket JSSE personalizzato, impostare MQEnvironment.sslSocketFactory sull'oggetto factory personalizzato. I dettagli variano tra le diverse implementazioni JSSE.

Diverse implementazioni JSSE possono fornire funzioni differenti. Ad esempio, un'implementazione JSSE specializzata potrebbe consentire la configurazione di un particolare modello di hardware di codifica. Inoltre, alcuni provider JSSE consentono la personalizzazione dei keystore e dei truststore in base al programma o consentono di modificare la scelta del certificato di identità dal keystore. In JSSE, tutte queste personalizzazioni vengono astratte in una classe factory, javax.net.ssl.SSLSocketFactory.

Consultare la documentazione JSSE per i dettagli su come creare un'implementazione SSLSocketFactory personalizzata. I dettagli variano da fornitore a provider, ma una tipica sequenza di passi potrebbe essere:

1. Creazione di un oggetto SSLContext utilizzando un metodo statico su SSLContext
2. Inizializzare questo SSLContext con le implementazioni KeyManager e TrustManager appropriate (create dalle proprie classi factory)
3. Crea un SSLSocketFactory da SSLContext

Quando si dispone di un oggetto SSLSocketFactory , impostare MQEnvironment.sslSocketFactory sull'oggetto factory personalizzato. Ad esempio:

```
javax.net.ssl.SSLSocketFactory sf = sslContext.getSocketFactory();
MQEnvironment.sslSocketFactory = sf;
```

IBM WebSphere MQ classes for Java utilizzare questo SSLSocketFactory per connettersi al gestore code IBM WebSphere MQ . Questa proprietà può essere impostata anche utilizzando CMQC.SSL_SOCKET_FACTORY_PROPERTY. Se sslSocketFactory è impostato su null, viene utilizzato il valore predefinito SSLSocketFactory della JVM. Questa proprietà viene ignorata se la suite sslCiphernon è impostata.

Quando si utilizzano SSLSocketFactoriespersonalizzati, considerare l'effetto della condivisione della connessione TCP/IP. Se la condivisione della connessione è possibile, non viene richiesto un nuovo socket del SSLSocketFactory fornito, anche se il socket prodotto sarebbe diverso in qualche modo nel contesto di una successiva richiesta di connessione. Ad esempio, se un certificato client differente deve essere presentato su una connessione successiva, la condivisione della connessione non deve essere consentita.

Esecuzione di modifiche al keystore o al truststore JSSE in WebSphere MQ classes per Java

Se si modifica il keystore o il truststore JSSE, è necessario eseguire alcune azioni per rendere effettive le modifiche.

Se si modifica il contenuto del keystore o del truststore JSSE o si modifica l'ubicazione del keystore o del file truststore, le classi WebSphere MQ per le applicazioni Java in esecuzione al momento non acquisiscono automaticamente le modifiche. Per rendere effettive le modifiche, è necessario eseguire le seguenti operazioni:

- Le applicazioni devono chiudere tutte le relative connessioni ed eliminare tutte le connessioni inutilizzate nei pool di connessioni.
- Se il provider JSSE memorizza nella cache le informazioni dal keystore e dal truststore, tali informazioni devono essere aggiornate.

Una volta eseguite queste azioni, le applicazioni possono ricreare le connessioni.

A seconda della modalità di progettazione delle applicazioni e della funzione fornita dal provider JSSE, potrebbe essere possibile eseguire queste azioni senza arrestare e riavviare le applicazioni. Tuttavia, l'arresto e il riavvio delle applicazioni potrebbe essere la soluzione più semplice.

Gestione degli errori quando si utilizza SSL con classi WebSphere MQ per Java

È possibile emettere un numero di codici di errore dalle classi WebSphere MQ per Java durante la connessione a un gestore code mediante SSL.

Questi sono spiegati nel seguente elenco:

MQRC_SSL_NOT_ALLOWED
La proprietà sslCipherSuite è stata impostata, ma è stata utilizzata la connessione dei bind. Solo la connessione client supporta SSL.

ERRORE MQRC_JSSE
Il fornitore JSSE ha riportato un errore che non può essere gestito da WebSphere MQ. Ciò potrebbe essere causato da un problema di configurazione con JSSE o perché non è stato possibile convalidare il certificato presentato dal gestore code. L'eccezione prodotta da JSSE può essere richiamata utilizzando il metodo getCause() su MQException.

ERRORE MQRC_SSL_INITIALIZATION_ERROR
È stata emessa una chiamata MQCONN o MQCONNX con le opzioni di configurazione SSL specificate, ma si è verificato un errore durante l'inizializzazione dell'ambiente SSL.

MQRC_SSL_PEER_NAME_MISMATCH
Il pattern DN specificato nella proprietà Nome sslPeernon corrisponde al DN presentato dal gestore code.

ERRORE MQRC_SSL_PEER_NAME_ERROR
Il modello DN specificato nella proprietà Nome sslPeernon è valido.

MQRC_UNSUPPORT_CIPHER_SUITE
La CipherSuite indicata nella suite sslCiphernon è stata riconosciuta dal fornitore JSSE. Un elenco completo di CipherSuites supportati dal provider JSSE può essere ottenuto da un programma utilizzando il metodo SSLSocketFactory.getSupportedCipherSuites(). Un elenco di CipherSuites che possono essere utilizzati per comunicare con WebSphere MQ è disponibile in “SSL CipherSpecs e CipherSuites in classi WebSphere MQ per Java” a pagina 707.

MQRC_SSL_CERTIFICATE_REVOKED
Il certificato presentato dal gestore code è stato trovato in un CRL specificato con la proprietà sslCertStores. Aggiornare il gestore code per utilizzare i certificati attendibili.

ERRORE - CERT_STORE_MQRC_SSL_CERT_
Non è stato possibile ricercare in nessuno dei CertStores forniti il certificato presentato dal gestore code. Il metodo MQException.getCause() restituisce l'errore che si è verificato durante la ricerca del primo CertStore tentato. Se l'eccezione causale è NoSuchElementException, ClassCastException o NullPointerException, verificare che la raccolta specificata nella proprietà sslCertStores contenga almeno un oggetto CertStore valido.

SSL CipherSpecs e CipherSuites in classi WebSphere MQ per Java

Se un'applicazione IBM WebSphere MQ classes for Java può stabilire una connessione a un gestore code dipende dalla CipherSpec specificata all'estremità server del canale MQI e dalla CipherSuite specificata all'estremità client.

Per ogni combinazione di CipherSpec e CipherSuite, se un'applicazione di IBM WebSphere MQ classes for Java può connettersi a un gestore code dipende dal valore del campo sslFipsObbligatorio nella classe MQEnvironment o dal valore della proprietà di ambiente CMQC.SSL_FIPS_REQUIRED_PROPERTY.

All'estremità del server di un canale MQI, il nome di un CipherSpec può essere specificato come valore del parametro SSLCIPH in un comando DEFINE CHANNEL CHLTYPE (SVRCONN). All'estremità client di un canale MQI, un'applicazione IBM WebSphere MQ classes for Java può impostare il campo sslCipherSuite nella classe MQEnvironment oppure impostare la proprietà di ambiente CMQC.SSL_CIPHER_SUITE_PROPERTY.

Configurazione della tua applicazione per utilizzare le associazioni IBM Java o Oracle Java CipherSuite

Da IBM WebSphere MQ Version 7.5.0, Fix Pack 5, è possibile configurare se l'applicazione utilizza le associazioni IBM Java CipherSuite a WebSphere MQ CipherSpec predefinite oppure Oracle CipherSuite a WebSphere MQ CipherSpec . Pertanto, puoi utilizzare TLS CipherSuites se la tua applicazione utilizza un JRE IBM o un JRE Oracle . La proprietà di sistema Java `com.ibm.mq.cfg.useIBMCipherMappings` controlla quali associazioni vengono utilizzate. La proprietà può essere uno dei seguenti valori:

vero
Utilizzare le associazioni IBM Java CipherSuite per WebSphere MQ CipherSpec .

Questo è il valore predefinito.

falso
Utilizzare le associazioni Oracle CipherSuite a WebSphere MQ CipherSpec .

La seguente tabella elenca i CipherSpecs supportati da IBM WebSphere MQ e i relativi CipherSuitesequivalenti. La tabella indica anche se un'applicazione IBM WebSphere MQ classes for Java può stabilire una connessione a un gestore code se viene specificato CipherSpec all'estremità server del canale MQI e l'equivalente CipherSuite viene specificato all'estremità client.

Tabella 89. CipherSpecs supportati da WebSphere MQ e relativi CipherSuites equivalenti

CipherSpec	CipherSuite equivalente	Connessione possibile se SFIPS [1] è impostato su YES?
NULL_MD5	SSL_RSA_WITH_NULL_MD5	No
NULL_SHA	SSL_RSA_WITH_NULL_SHA	No
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5 (JREIBM) Nessun equivalente per Oracle JRE.	No
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	No
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA (JREIBM) Nessun equivalente per Oracle JRE.	No
RC2_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (IBM JRE) SSL_RSA_EXPORT_WITH_RC4_40_MD5 (JREOracle)	No
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA (JREIBM) Nessun equivalente per Oracle JRE.	No
RC4_56_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA (JREIBM) Nessun equivalente per Oracle JRE.	No
DES_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA (JREIBM) Nessun equivalente per Oracle JRE.	No

Tabella 89. CipherSpecs supportati da WebSphere MQ e relativi CipherSuites equivalenti (Continua)		
CipherSpec	**CipherSuite equivalente**	**Connessione possibile se SFIPS [1] è impostato su YES?**
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA (JREIBM) Nessun equivalente per Oracle JRE.	No
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256 (IBM JRE) TLS_RSA_WITH_NULL_SHA256 (Oracle JRE)	Nessun [7]
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA (JREIBM) TLS_RSA_WITH_AES_128_CBC_SHA (JREOracle)	Sì [5] [7]
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256 (JREIBM) TLS_RSA_WITH_AES_128_CBC_SHA256 (Oracle JRE)	Sì [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA (JREIBM) TLS_RSA_WITH_AES_256_CBC_SHA (JREOracle)	Sì [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256 (JREIBM) TLS_RSA_WITH_AES_256_CBC_SHA256 (JREOracle)	Sì [5] [7]
AES_SHA_IT[2]		
TLS_RSA_WITH_DES_CBC_SHA[8]	SSL_RSA_WITH_DES_CBC_SHA	No[3]
TLS_RSA_WITH_3DES_EDE_CBC_SHA[8] [9]	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Sì
FIPS_WITH_DES_CBC_SHA	SSL_RSA_FIPS_WITH_DES_CBC_SHA (JREIBM) Nessun equivalente per Oracle JRE.	Nessun[4]
FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (JREIBM) Nessun equivalente per Oracle JRE.	Nessun[6]

Note:

1. In un'applicazione IBM WebSphere MQ classes for Java , indicare che solo gli algoritmi certificati FIPS devono essere utilizzati impostando il campo sslFipsObbligatorio nella classe MQEnvironment su `true` e indicare che gli algoritmi non certificati FIPS possono essere utilizzati anche impostando il campo sslFipsObbligatorio su `false`. In alternativa, impostare la proprietà dell'ambiente CMQC.SSL_FIPS_REQUIRED_PROPERTY.
2. Questo CipherSpec non ha un CipherSuiteequivalente.
3. Questa CipherSpec era certificata FIPS 140-2 prima del 19th maggio 2007.
4. Questa CipherSpec era certificata FIPS 140-2 prima del 19th maggio 2007. Il nome FIPS_WITH_DES_CBC_SHA è storico e riflette il fatto che questo CipherSpec era precedentemente (ma non più) conforme a FIPS. Questa CipherSpec è obsoleta e non se ne consiglia l'utilizzo.

5. Questi 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) non possono essere utilizzati per proteggere una connessione da WebSphere MQ Explorer a un gestore code a meno che non vengano applicati i file JRE senza limitazioni appropriati.

   Consultare Informazioni sulla sicurezza per ulteriori informazioni sui file delle politiche.
6. Il nome FIPS_WITH_3DES_EDE_CBC_SHA è storico e riflette il fatto che questo CipherSpec era precedentemente (ma non è più) compatibile con FIPS. Questa CipherSpec è obsoleta e non se ne consiglia l'utilizzo.
7. Questi 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) richiedono IBM JRE 6.0 SR13 FP2 , 7.0 SR4 FP2 o versioni successive.
8. Questi CipherSpecs (TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_DES_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA256) possono utilizzare SSLv3 o TLS. Per impostazione predefinita, quando FIPS non è abilitato, si utilizza SSLv3 . Per utilizzare TLS, impostare la proprietà di sistema Java **`com.ibm.mq.cfg.preferTLS`** su `true`.
9. Questa CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA è obsoleta. Tuttavia, può essere ancora utilizzato per trasferire fino a 32 GB di dati prima che la connessione venga terminata con l'errore AMQ9288. Per evitare questo errore, è necessario evitare di utilizzare il triplo DES o abilitare la reimpostazione della chiave segreta quando si utilizza questo CipherSpec.

Informazioni correlate

Specifica che solo i CipherSpecs certificati FIPS vengono utilizzati al runtime sul client MQI

Federal Information Processing Standards (FIPS) per UNIX, Linux e Windows

Blog MQdev: MQ Java, TLS Ciphers, Non-IBM JREs & APARs IT06775, IV66840, IT09423, IT10837

Blog MQdev: la relazione tra MQ CipherSpecs e Java Cipher Suites

Esecuzione delle classi WebSphere MQ per applicazioni Java

Se si scrive un'applicazione (una classe che contiene un metodo main ()), utilizzando la modalità client o bind, eseguire il programma utilizzando l'interprete Java.

Utilizzare il comando:

```
java -Djava.library.path=library_path MyClass
```

dove *percorso libreria* è il percorso delle classi WebSphere MQ per le librerie Java (consultare WebSphere MQ classes per le librerie Java).

WebSphere MQ per il comportamento dipendente dall'ambiente Java

Le classi WebSphere MQ per Java consentono di creare applicazioni che possono essere eseguite su versioni differenti di WebSphere MQ. Questa raccolta di argomenti descrive il funzionamento delle classi Java che dipendono da queste differenti versioni.

WebSphere MQ classes per Java fornisce un nucleo di classi che forniscono funzioni e comportamenti coerenti in tutti gli ambienti. Le funzioni esterne a questo core dipendono dalla capacità del gestore code a cui è connessa l'applicazione.

Tranne dove indicato qui, il comportamento mostrato è quello descritto in Application Programming Reference appropriato al gestore code.

Classi principali nelle classi WebSphere MQ per Java

WebSphere MQ classes per Java contiene una serie di classi principali, che possono essere utilizzate in tutti gli ambienti.

La seguente serie di classi è considerata classi principali e può essere utilizzata in tutti gli ambienti con solo le variazioni minori elencate in “Limitazioni e variazioni per le classi principali delle classi WebSphere MQ per Java” a pagina 712.

- Ambiente MQ
- Eccezione MQException
- Opzioni MQGetMessage

  Escluso:

  – MatchOptions
  – GroupStatus
  – SegmentStatus
  – Segmentazione
- MQManagedObject

  Escluso:

  – inquire ()
  – set ()
- Messaggio MQT

  Escluso:

  – groupId
  – messageFlags
  – Numero messageSequence
  – offset
  – originalLength
- MQPoolServices
- Eventi MQPoolServices
- MQPoolServicesEventListener
- MQPoolToken
- Opzioni MQPutMessage

  Escluso:

  – KnownDestCount
  – UnknownDestCount
  – InvalidDestCount
  – recordFields
- Processo MQ
- MQQUEUE
- MQQueueManager

  Escluso:

  – begin ()
  – Elenco accessDistribution()
- Gestore MQSimpleConnection
- MQArgomento
- MQC

Nota:

1. Alcune costanti non sono incluse nel core (per i dettagli, consultare “Limitazioni e variazioni per le classi principali delle classi WebSphere MQ per Java” a pagina 712); non utilizzarle in programmi completamente portatili.
2. Alcune piattaforme non supportano tutte le modalità di connessione. Su queste piatteforme, è possibile utilizzare solo le classi principali e le opzioni relative alle modalità supportate. (Consultare “Opzioni di connessione per WebSphere MQ classes for Java” a pagina 655.)

Limitazioni e variazioni per le classi principali delle classi WebSphere MQ per Java

Le classi principali generalmente si comportano in modo congruente in tutti gli ambienti, anche se le chiamate MQI equivalenti normalmente presentano differenze di ambiente. Il comportamento è come se fosse utilizzato un gestore code Windows, UNIX o Linux WebSphere MQ , ad eccezione delle seguenti limitazioni e variazioni minori.

*Limitazioni per i valori MQGMO_ * nelle classi WebSphere MQ per Java*
Alcuni valori MQGMO_ * non sono supportati da tutti i gestori code.

L'utilizzo dei seguenti valori MQGMO_ * potrebbe causare una MQException generata da MQQueue.get():

MQGMO_SYNCPOINT_IF_PERSISTENT
MQGMO_MARK_SKIP_BACKOUT
MQGMO_BROWSE_MSG_UNDER_CURSOR
LOCK_MQGMO
MQGMO_UNLOCK
ORDER LOGICAL_MQGMO_
MESSAGGIO_COMPLETAMENTO_MQGM
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

Inoltre, MQGMO_SET_SIGNAL non è supportato quando utilizzato da Java.

*Limitazioni per i valori MQPMRF_ * nelle classi WebSphere MQ per Java*
Questi sono utilizzati solo quando si inseriscono i messaggi in un elenco di distribuzione e sono supportati solo dai gestori code che supportano gli elenchi di distribuzione. Ad esempio, i gestori code z/OS non supportano gli elenchi di distribuzione.

*Limitazioni per i valori MQPMO_ * nelle classi WebSphere MQ per Java*
Alcuni valori MQPMO_ * non sono supportati da tutti i gestori code

L'utilizzo dei seguenti valori MQPMO_ * potrebbe causare un'eccezione MQException generata da MQQueue.put() o da MQQueueManager.put ():

ORDER MQPMO_LOGICAL_
ID_CORREL_NEW_MQPMO_
ID MQPMO_NEW_MESSAGE_ID
MQPMO_RESOLVE_LOCAL_Q

*Limitazioni e variazioni per i valori MQCNO_ * in WebSphere MQ classes per Java*
Alcuni valori `MQCNO_*` non sono supportati.

- La riconnessione automatica del client non è supportata dalle classi WebSphere MQ per Java. Indipendentemente dal valore `MQCNO_RECONNECT_*` impostato, la connessione continua a funzionare come se si impostasse `MQCNO_RECONNECT_DISABLED`.

- MQCNO_FASTPATH viene ignorato sui gestori code che non supportano MQCNO_FASTPATH. Viene anche ignorato dalle connessioni client.

*Limitazioni per i valori MQRO_ * nelle classi WebSphere MQ per Java*
È possibile impostare le seguenti opzioni del report.

MQRO_EXCEPTION_WITH_FULL_DATA
MQRO_EXPIRATION_WITH_FULL_DATA
DATI_COA_WITH_MQRO_FULL_DATA
DAD_COD MQRO_WITH_FULL_DATA
MQRO_DISCARD_MSG
MQRO_PASS_DISCARD_E_SCADENZA

Per ulteriori informazioni, consultare Report.

Funzioni al di fuori delle classi principali delle classi WebSphere MQ per Java

Le classi WebSphere MQ per Java contengono alcune funzioni specificamente progettate per utilizzare estensioni API non supportate da tutti i gestori code. Questa raccolta di argomenti descrive come si comportano quando si utilizza un gestore code che *non* li supporta.

Variazioni nell'opzione del costruttore MQQueueManager
Alcuni dei costruttori MQQueueManager includono un argomento numero intero facoltativo. Alcuni valori di questo argomento non sono accettati su tutte le piattaforme.

Quando un costruttore MQQueueManager include un argomento integer facoltativo, viene associato al campo delle opzioni MQCNO dell'MQI e viene utilizzato per passare da una connessione normale a una connessione fast path. Questo formato esteso del costruttore viene accettato in tutti gli ambienti, se le uniche opzioni utilizzate sono MQCNO_STANDARD_BINDING o MQCNO_FASTPATH_BINDING. Qualsiasi altra opzione causa l'errore del costruttore con MQRC_OPTIONS_ERROR. L'opzione di percorso rapido CMQC.MQCNO_FASTPATH_BINDING viene rispettato solo con una connessione di bind a un gestore code che lo supporta. In altri ambienti, viene ignorato.

Restrizioni sul metodo MQQueueManager.begin ()
Questo metodo può essere utilizzato solo rispetto a un gestore code WebSphere MQ su sistemi UNIX, Linuxo Windows in modalità bind. Altrimenti, ha esito negativo con MQRC_ENVIRONMENT_ERROR.

Per ulteriori dettagli, vedere “Coordinazione JTA/JDBC utilizzando le classi WebSphere MQ per Java” a pagina 700.

Variazioni nei campi Opzioni di MQGetMessage
Alcuni gestori code non supportano la struttura MQGMO Versione 2, pertanto è necessario impostare alcuni campi sui valori predefiniti.

Quando si utilizza un gestore code che non supporta la struttura MQGMO versione 2, lasciare impostati i seguenti campi sui valori predefiniti:

GroupStatus
SegmentStatus
Segmentazione

Inoltre, il campo MatchOptions supporta solo MQMO_MATCH_MSG_ID e MQMO_MATCH_CORREL_ID. Se si inseriscono valori non supportati in questi campi, il successivo MQDestination.get() ha esito negativo con MQRC_GMO_ERROR. Se il gestore code non supporta la struttura MQGMO Versione 2, questi campi non vengono aggiornati dopo un esito positivo di MQDestination.get().

Limitazioni negli elenchi di distribuzione nelle classi WebSphere MQ per Java
Non tutti i gestori code consentono di aprire un MQDistributionList.

Le seguenti classi vengono utilizzate per creare elenchi di distribuzione:

MQDistributionList

MQDistributionListElemento
MQMessageTracker

È possibile creare e popolare gli elementi MQDistributionLists e MQDistributionListin qualsiasi ambiente, ma non tutti i gestori code consentono di aprire un MQDistributionList. In particolare, i gestori code z/OS non supportano gli elenchi di distribuzione. Il tentativo di apertura di un MQDistributionList quando si utilizza un gestore code di questo tipo risulta in MQRC_OD_ERROR.

Variazioni nei campi Opzioni di MQPutMessage
Se un gestore code non supporta gli elenchi di distribuzione, alcuni campi MQPMO vengono trattati in modo diverso.

Quattro campi in MQPMO vengono rappresentati come le seguenti variabili membro nella classe di opzioni MQPutMessage:

KnownDestCount
UnknownDestCount
InvalidDestCount
recordFields

Questi campi sono principalmente destinati ad essere utilizzati con gli elenchi di distribuzione. Tuttavia, un gestore code che supporta gli elenchi di distribuzione compila i campi DestCount dopo un MQPUT in una singola coda. Ad esempio, se la coda si risolve in una coda locale, knownDestCount è impostato su 1 e gli altri due campi di conteggio sono impostati su 0.

Se il gestore code non supporta gli elenchi di distribuzione, questi valori vengono simulati nel modo seguente:

- Se l'operazione put () ha esito positivo, unknownDestCount è impostato su 1 e gli altri sono impostati su 0.
- Se l'operazione put () ha esito negativo, invalidDestCount è impostato su 1 e gli altri sono impostati su 0.

la variabile recordFields viene utilizzata con gli elenchi di distribuzione. Un valore può essere scritto in recordFields in qualsiasi momento, indipendentemente dall'ambiente. Viene ignorato se l'oggetto MQPutMessageOptions viene utilizzato su un successivo MQDestination.put() o MQQueueManager.put (), piuttosto che MQDistributionList.put ().

Limitazioni nei campi MQMD con le classi WebSphere MQ per Java
Alcuni campi MQMD relativi alla segmentazione dei messaggi devono essere lasciati al loro valore predefinito quando si utilizza un gestore code che non supporta la segmentazione.

I seguenti campi MQMD sono principalmente interessati alla segmentazione dei messaggi:

GroupId
MsgSeqNumber
Offset
MsgFlags
OriginalLength

Se un'applicazione imposta uno qualsiasi di questi campi MQMD su valori diversi da quelli predefiniti, e quindi esegue un comando put () o get () su un gestore code che non li supporta, il comando put () o get () genera un'eccezione MQException con MQRC_MD_ERROR. Un put () o get () con un gestore code di questo tipo lascia sempre i campi MQMD impostati sui valori predefiniti. Non inviare un messaggio raggruppato o segmentato a un'applicazione Java che viene eseguita rispetto a un gestore code che non supporta il raggruppamento e la segmentazione dei messaggi.

Se un'applicazione Java tenta di richiamare () un messaggio da un gestore code che non supporta questi campi e il messaggio fisico da richiamare fa parte di un gruppo di messaggi segmentati (vale a dire, ha valori non predefiniti per i campi MQMD), viene richiamato senza errori. Tuttavia, i campi MQMD in MQMessage non vengono aggiornati, la proprietà del formato MQMessage è impostata su MQFMT_MD_EXTENSION e i dati del messaggio true sono preceduti da una struttura MQMDE che contiene i valori per i nuovi campi.

Restrizioni per le classi WebSphere MQ per Java in CICS Transaction Server

Nell'ambiente CICS Transaction Server for z/OS , solo al thread principale (primo) è consentito emettere chiamate CICS o WebSphere MQ .

Tenere presente che le classi JMS WebSphere MQ non sono supportate per l'utilizzo in un'applicazione CICS Java.

Pertanto, non è possibile condividere gli oggetti MQQueueManager o MQQueue tra i thread in questo ambiente o creare un nuovo MQQueueManager su un thread secondario.

Esecuzione di classi IBM WebSphere MQ per applicazioni Java nella piattaforma Java Enterprise Edition

Vi sono alcune limitazioni e considerazioni di progettazione che devono essere prese in considerazione prima di utilizzare le classi IBM WebSphere MQ per Java in Java EE

Le classi di IBM WebSphere MQ per Java hanno delle limitazioni quando vengono utilizzate in un ambiente Java EE . Ci sono anche ulteriori considerazioni che devono essere prese in considerazione quando si progettano, implementano e gestiscono classi IBM WebSphere MQ per l'applicazione Java che vengono eseguite in un ambiente Java EE . Queste limitazioni e considerazioni sono descritte nelle seguenti sezioni.

Limitazioni transazioni JTA

L'unico gestore transazioni supportato per applicazioni che utilizzano classi IBM WebSphere MQ per Java è lo stesso IBM WebSphere MQ . Anche se un'applicazione sotto il controllo JTA può utilizzare le classi IBM WebSphere MQ per Java, qualsiasi lavoro eseguito attraverso queste classi non è controllato dalle unità di lavoro JTA. Formano invece unità di lavoro locali separate da quelle gestite dal server delle applicazioni tramite le interfacce JTA. In particolare, qualsiasi rollback della transazione JTA non risulta in un rollback dei messaggi inviati o ricevuti. Questa limitazione si applica alle transazioni gestite dall'applicazione o dal bean e alle transazioni gestite dal contenitore e a tutti i contenitori Java EE . Per eseguire la messaggistica direttamente con IBM WebSphere MQ all'interno delle transazioni coordinate del server delle applicazioni, è necessario utilizzare le classi IBM WebSphere MQ per JMS.

Creazione thread

IBM WebSphere MQ classes per Java crea i thread internamente per varie operazioni. Ad esempio, quando si utilizza la modalità BINDINGS per richiamare direttamente un gestore code locale, le chiamate vengono effettuate su un thread 'worker' creato internamente dalle classi IBM WebSphere MQ per Java. Altri thread possono essere creati internamente, ad esempio per cancellare le connessioni inutilizzate da un pool di connessioni o per rimuovere le sottoscrizioni per le applicazioni di pubblicazione / sottoscrizione terminate.

Alcune applicazioni Java EE (ad esempio quelle in esecuzione nei contenitori EJB e Web) non devono creare nuovi thread. Invece, tutto il lavoro deve essere eseguito sui thread dell'applicazione principale gestiti dal server delle applicazioni. Quando le applicazioni utilizzano le classi IBM WebSphere MQ per Java, il server delle applicazioni potrebbe non essere in grado di distinguere tra il codice dell'applicazione e le classi IBM WebSphere MQ per il codice Java , in modo che i thread precedentemente descritti non siano conformi alla specifica del contenitore. IBM WebSphere MQ classes per JMS non interrompe queste specifiche Java EE e può quindi essere utilizzato.

Limitazioni di sicurezza

Le politiche di sicurezza implementate da un server delle applicazioni potrebbero impedire alcune operazioni eseguite dalle classi IBM WebSphere MQ per l'API Java , come la creazione e l'utilizzo di nuovi thread di controllo (come descritto nelle sezioni precedenti).

Ad esempio, i server delle applicazioni generalmente vengono eseguiti con la sicurezza Java disabilitata per impostazione predefinita e ne consentono l'abilitazione tramite una configurazione specifica del server delle applicazioni (alcuni server delle applicazioni consentono anche una configurazione più dettagliata delle politiche utilizzate in Java Security). Quando è abilitata la sicurezza Java , le classi

IBM WebSphere MQ per Java potrebbero interrompere le regole di thread della politica di sicurezza di Java definite per il server delle applicazioni e l'API potrebbe non essere in grado di creare tutti i thread necessari per il funzionamento. Per evitare problemi con la gestione dei thread, l'utilizzo delle classi IBM WebSphere MQ per Java non è supportato in ambienti in cui è abilitata la sicurezza Java .

Considerazioni sull'isolamento dell'applicazione

Un vantaggio previsto dell'esecuzione di applicazioni in un ambiente Java EE è l'isolamento dell'applicazione. La progettazione e l'implementazione delle classi IBM WebSphere MQ per Java precedano l'ambiente Java EE . IBM WebSphere MQ le classi per Java possono essere utilizzate in una modalità che non supporta il concetto di isolamento dell'applicazione. Esempi specifici di considerazioni in questo settore sono:

- L'utilizzo delle impostazioni statiche (a livello di processo JVM) all'interno della classe MQEnvironment, come:
  - l'ID utente e la password da utilizzare per l'identificazione e l'autenticazione della connessione
  - il nome host, la porta e il canale utilizzati per le connessioni client
  - Configurazione SSL per le connessioni client protette

  La modifica delle proprietà MQEnvironment a vantaggio di un'applicazione influisce anche su altre applicazioni che utilizzano le stesse proprietà. Quando viene eseguita in un ambiente a più applicazioni, ad esempio Java EE, ciascuna applicazione deve utilizzare la propria configurazione distinta mediante la creazione di oggetti MQQueueManager con una serie specifica di proprietà, anziché utilizzare le proprietà configurate nella classe MQEnvironment a livello di processo.
- La classe MQEnvironment introduce un certo numero di metodi statici che agiscono globalmente su tutte le applicazioni che utilizzano le classi IBM WebSphere MQ per Java all'interno dello stesso processo JVM e non c'è modo di sovrascrivere questo comportamento per particolari applicazioni. Alcuni esempi comprendono:
  - configurazione delle proprietà SSL, come ad esempio l'ubicazione del keystore
  - configurazione uscite canale client
  - abilitazione o disabilitazione della traccia diagnostica
  - gestione del pool di connessioni predefinito utilizzato per ottimizzare l'uso delle connessioni ai gestori code

  Il richiamo di tali metodi influisce sulle applicazioni in esecuzione nello stesso ambiente Java EE .
- Il pool di connessioni è abilitato per ottimizzare il processo di creazione di più connessioni allo stesso gestore code. Il gestore pool di connessioni predefinito è a livello di processo e condiviso da più applicazioni. Le modifiche alla configurazione del pool di connessione, ad esempio la sostituzione del gestore connessioni predefinito per un'applicazione utilizzando il metodo MQEnvironment.setDefaultConnectionManager(), influiscono sulle altre applicazioni in esecuzione nello stesso server delle applicazioni Java EE .
- SSL è configurato per le applicazioni che utilizzano le classi IBM WebSphere MQ per Java utilizzando la classe MQEnvironment e le proprietà oggetto MQQueueManager . Non è integrato con la configurazione della sicurezza gestita del server delle applicazioni stesso. È necessario assicurarsi di configurare le classi IBM WebSphere MQ per Java in modo appropriato per fornire il livello di sicurezza richiesto e non utilizzare la configurazione del server delle applicazioni.

Limitazioni della modalità di bind

IBM WebSphere MQ e WebSphere Application Server possono essere installati sulla stessa macchina in modo che le versioni principali del gestore code e dell'adattatore risorse IBM WebSphere MQ forniti in WebSphere Application Server siano differenti. Ad esempio, WebSphere Application Server Version 7.0, che fornisce un IBM WebSphere MQ livello RA 7.0.1, può essere installata sulla stessa macchina di un gestore code Version 6.0 .

Se le versioni principali del gestore code e dell'adattatore di risorse sono differenti, non è possibile utilizzare le connessioni di bind. Tutte le connessioni da WebSphere Application Server al gestore code che utilizzano l'adattatore di risorse devono utilizzare connessioni di tipo client. Le connessioni dei collegamenti possono essere utilizzate se le versioni sono le stesse.

Utilizzo delle classi WebSphere MQ per JMS

WebSphere MQ classes for Java Message Service (WebSphere MQ classes for JMS) è il fornitore JMS fornito con WebSphere MQ. Oltre ad implementare le interfacce definite nel pacchetto javax.jms , WebSphere MQ classes per JMS fornisce due serie di estensioni all'API JMS.

La specificazione JMS definisce una serie di interfacce che le applicazioni possono utilizzare per eseguire operazioni di messaggistica. Il pacchetto javax.jms definisce le interfacce JMS e un fornitore JMS implementa tali interfacce per un prodotto di messaggistica specifico. WebSphere MQ versione 7.5 utilizza attualmente la specifica JMS 1.1 . WebSphere MQ classes per JMS è un provider JMS che implementa le interfaccia JMS per WebSphere MQ.

La specifica JMS prevede che gli oggetti ConnectionFactory e Destination siano oggetti gestiti. Un amministratore crea e gestisce oggetti gestiti in un repository centrale e un'applicazione JMS richiama tali oggetti utilizzando JNDI (Java Naming and Directory Interface). WebSphere MQ classes per JMS supporta l'uso di oggetti amministrati, e un amministratore può utilizzare lo strumento di amministrazione JMS WebSphere MQ o WebSphere MQ Explorer per creare e gestire oggetti amministrati.

WebSphere MQ classes for JMS fornisce inoltre due serie di estensioni all'API JMS. Il focus principale di queste estensioni riguarda la creazione e la configurazione di factory di connessione e destinazioni in modo dinamico al runtime, ma le estensioni forniscono anche funzioni che non sono direttamente correlate alla messaggistica, come la funzione per la determinazione dei problemi.

Le estensioni JMS WebSphere MQ
I rilasci precedenti di WebSphere MQ classes for JMS contengono estensioni implementate in oggetti quali MQConnectionFactory, MQQueue e MQTopic. Questi oggetti hanno proprietà e metodi specifici per WebSphere MQ. Gli oggetti possono essere gestiti oppure un'applicazione può creare gli oggetti in modo dinamico al runtime. Questa release delle classi WebSphere MQ per JMS conserva queste estensioni, ora note come estensioni JMS WebSphere MQ . È possibile continuare a utilizzare, senza modifiche, tutte le applicazioni che utilizzano queste estensioni.

Le estensioni JMS IBM
Questa release delle classi WebSphere MQ per JMS fornisce una serie più generica di estensioni all'API JMS, che non sono specifiche di WebSphere MQ come sistema di messaggistica. Queste estensioni sono note come estensioni JMS IBM e hanno i seguenti obiettivi generali:

- Per fornire un livello maggiore di coerenza tra i provider JMS IBM
- Per semplificare la scrittura di un'applicazione bridge tra due sistemi di messaggistica IBM
- Per semplificare la porta di un'applicazione da un fornitore JMS IBM a un altro

Le estensioni forniscono funzioni simili a quelle fornite in Message Service Client for C/C++ e Message Service Client for .NET.

Perché utilizzare WebSphere MQ classes per JMS?

L'utilizzo di WebSphere MQ classes for JMS presenta i seguenti vantaggi:

- È possibile riutilizzare le capacità JMS.

  WebSphere MQ classes for JMS è un provider JMS che implementa interfacce JMS per WebSphere MQ come sistema di messaggistica. Se la tua organizzazione non ha dimestichezza con WebSphere MQ, ma ha già capacità di sviluppo di applicazioni JMS, potresti trovare più semplice utilizzare l'API JMS familiare per accedere a risorse WebSphere MQ piuttosto che a una delle altre API fornite con WebSphere MQ.
- JMS è parte integrante di Java Platform, Enterprise Edition (Java EE).

JMS è l'API naturale da utilizzare per la messaggistica sulla piattaforma Java EE . Ogni server delle applicazioni compatibile con Java EE deve includere un provider JMS. Ô possibile utilizzare JMS in client delle applicazioni, servlet, JSP (JavaServer pages), EJB (enterprise Java beans) e MDB (message driven beans). Notare in particolare che le applicazioni Java EE utilizzano gli MDB per elaborare i messaggi in modo asincrono e tutti i messaggi vengono consegnati agli MDB come messaggi JMS.

- Un amministratore può creare e gestire oggetti amministrati JMS in un repository centrale e WebSphere MQ classes for JMS applications può recuperare questi oggetti utilizzando JNDI (Java Naming and Directory Interface).

  Le destinazioni e i factory di connessione JMS incapsulano WebSphere MQ informazioni specifiche quali i nomi dei gestori code, i nomi dei canali, le opzioni di connessione, i nomi delle code e i nomi degli argomenti. Se le factory di connessione e le destinazioni sono memorizzate come oggetti gestiti, queste informazioni non sono codificate in modo permanente in un'applicazione. Questa disposizione fornisce quindi all'applicazione un grado di indipendenza dalla configurazione sottostante di WebSphere MQ .
- JMS è un'API standard del settore che può fornire la portabilità dell'applicazione.

  Un'applicazione JMS può utilizzare JNDI per richiamare factory di connessione e destinazioni memorizzate come oggetti amministrati e utilizzare solo le interfacce definite nel pacchetto javax.jms per eseguire operazioni di messaggistica. L'applicazione è quindi completamente indipendente da qualsiasi provider JMS, come WebSphere MQ classes for JMS, e può essere trasferita da un provider JMS all'altro senza alcuna modifica all'applicazione.

  Se JNDI non è disponibile in un particolare ambiente dell'applicazione, un WebSphere MQ classes for JMS application può utilizzare le estensioni all'API JMS per creare e configurare le factory di connessione e le destinazioni in modo dinamico durante il runtime. L'applicazione è quindi completamente autonoma, ma è collegata alle classi WebSphere MQ per JMS come provider JMS.
- Le applicazioni bridge potrebbero essere più semplici da scrivere utilizzando JMS.

  Un'applicazione bridge è un'applicazione che riceve i messaggi da un sistema di messaggistica e li invia a un altro sistema di messaggistica. La scrittura di un'applicazione bridge può essere complicata utilizzando API e formati di messaggio specifici del prodotto. È invece possibile scrivere un'applicazione bridge utilizzando due provider JMS, uno per ciascun sistema di messaggistica. L'applicazione, quindi, utilizza solo un'API, l'API JMS ed elabora solo i messaggi JMS.

Introduzione a WebSphere MQ classes for JMS

Questo argomento fornisce una panoramica delle classi WebSphere MQ per JMS e indica le informazioni necessarie prima di utilizzare le classi WebSphere MQ per JMS.

Prerequisiti per le classi WebSphere MQ per JMS

Per sviluppare ed eseguire classi WebSphere MQ per applicazioni JMS, è necessario disporre di determinati componenti software come prerequisiti.

Per informazioni più aggiornate sui prerequisiti per WebSphere MQ classes for JMS, consultare il file readme WebSphere MQ .

Per sviluppare le classi WebSphere MQ per le applicazioni JMS, è necessario un SDK (Software Development Kit) Java 2. I dettagli dei JDK supportati con il proprio sistema operativo sono disponibili nella pagina dei requisiti di sistema di WebSphere MQ . Consultare Requisiti di WebSphere MQ.

Per eseguire le classi WebSphere MQ per applicazioni JMS, sono necessari i seguenti componenti software:

- Un gestore code WebSphere MQ
- Un JRE (Java Runtime Environment), per ogni sistema su cui si eseguono le applicazioni

Se si richiedono connessioni SSL per utilizzare moduli crittografici certificati FIPS 140-2, è necessario il provider IBM Java JSSE FIPS (IBMJSSEFIPS). Ogni IBM Java 2 SDK e JRE versione 5 o successiva contiene IBMJSSEFIPS.

È possibile utilizzare gli indirizzi Internet Protocol Versione 6 (IPv6) nelle classi WebSphere MQ per applicazioni JMS, purché gli indirizzi IPv6 siano supportati dalla JVM (Java virtual machine) e dall'implementazione TCP/IP sul proprio sistema operativo. Lo strumento di amministrazione JMS WebSphere MQ (consultare “Utilizzo dello strumento di amministrazione JMS WebSphere MQ” a pagina 936) accetta anche indirizzi IPv6 .

Lo strumento di gestione JMS WebSphere MQ e WebSphere MQ Explorer utilizzano JNDI (Java Naming and Directory Interface) per accedere a un servizio di directory, che memorizza gli oggetti gestiti. Le classi WebSphere MQ per le applicazioni JMS possono utilizzare JNDI anche per richiamare gli oggetti gestiti da un servizio directory. Un fornitore di servizi è un codice che fornisce l'accesso a un servizio directory associando le chiamate JNDI alle chiamate al servizio directory. I seguenti provider di servizi vengono forniti con WebSphere MQ classes per JMS:

- Un fornitore di servizi LDAP (Lightweight Directory Access Protocol) nei file ldap.jar e providerutil.jar. Il provider del servizio LDAP fornisce l'accesso a un servizio di directory basato su un server LDAP.
- Un provider del servizio file system nei file fscontext.jar e providerutil.jar. Il provider di servizi del file system fornisce l'accesso a un servizio di directory basato sul file system locale.

Se si intende utilizzare un servizio di directory basato su un server LDAP, è necessario installare e configurare un server LDAP o avere accesso a un server LDAP esistente. In particolare, è necessario configurare il server LDAP per memorizzare gli oggetti Java. Per informazioni su come installare e configurare il server LDAP, consultare la documentazione fornita con il server.

Preparazione dei programmi JMS per il client IBM WebSphere MQ per HP Integrity NonStop Server

Questo argomento spiega cosa è necessario sapere prima di sviluppare ed eseguire programmi JMS per il client IBM WebSphere MQ per HP Integrity NonStop Server.

Le classi IBM WebSphere MQ per JMS vengono installate come parte dell'installazione del client IBM WebSphere MQ per HP Integrity NonStop Server . Per i dettagli di un riepilogo del contenuto dell'installazione, consultare File system.

Alcuni aspetti della funzionalità del client sono specifici per il sistema operativo host. Per ulteriori informazioni sulle funzioni supportate per il client IBM WebSphere MQ per HP Integrity NonStop Server, consultare IBM WebSphere MQ client for HP Integrity NonStop Server supported environments and features.

Prerequisiti

Per creare ed eseguire le applicazioni JMS, il componente *HP Integrity NonStop Server for Java* deve essere installato e disponibile.

Configura

Per informazioni sull'impostazione dell'ambiente per eseguire e creare applicazioni in cui è possibile utilizzare le classi IBM WebSphere MQ per JMS, consultare “Variabili di ambiente utilizzate dalle classi IBM WebSphere MQ per JMS” a pagina 724.

Per informazioni sulla procedura richiesta per configurare un gestore code per accettare le connessioni dalle applicazioni client, consultare “Impostazione post - installazione per le classi WebSphere MQ per le applicazioni JMS” a pagina 772.

Per informazioni sulla convalida delle classi IBM WebSphere MQ per l'ambiente JMS, consultare “Il test di verifica dell'installazione point-to-point per WebSphere MQ classes per JMS” a pagina 776.

Scrittura di applicazioni

Per ulteriori informazioni sulla scrittura delle applicazioni JMS, consultare “Scrittura delle classi WebSphere MQ per le applicazioni JMS” a pagina 806.

Per ulteriori informazioni sull'utilizzo dello strumento di amministrazione JMS IBM WebSphere MQ , consultare “Utilizzo dello strumento di amministrazione JMS WebSphere MQ” a pagina 936.

Esempi

Le applicazioni di esempio vengono fornite nella seguente sottodirectory dell'installazione: `opt/mqm/samp/jms`.

Per ulteriori informazioni sui passi di configurazione richiesti per eseguire gli esempi, consultare “Preparazione ed esecuzione dei programmi di esempio” a pagina 110.

Risoluzione dei problemi

Per informazioni sulla risoluzione dei problemi, consultare “Risoluzione dei problemi con le classi IBM WebSphere MQ per JMS” a pagina 797.

Installazione e configurazione delle classi WebSphere MQ per JMS

Questa sezione descrive le directory e i file che vengono creati quando si installa WebSphere MQ classes for JMS e indica come configurare WebSphere MQ classes for JMS dopo l'installazione.

Concetti correlati

“Cosa è installato per le classi IBM WebSphere MQ per JMS” a pagina 721
Una serie di file e directory vengono creati quando si installano le classi IBM WebSphere MQ per JMS. Su Windows , alcune configurazioni vengono eseguite durante l'installazione impostando automaticamente le variabili di ambiente. Su altre piattaforme e in determinati ambienti Windows, è necessario impostare le variabili di ambiente prima di poter eseguire le classi IBM WebSphere MQ per applicazioni JMS.

“Esecuzione delle classi WebSphere MQ per applicazioni JMS in Java Security Manager” a pagina 730
WebSphere MQ classes for JMS può essere eseguito con il gestore sicurezza Java abilitato. Per eseguire correttamente le applicazioni con il gestore della protezione abilitato, è necessario configurare la JVM (Java virtual machine) con un file di configurazione della politica adatto.

“L'adattatore di risorse IBM WebSphere MQ” a pagina 734
L'adattatore risorse consente alle applicazioni in esecuzione in un application server di accedere alle risorse IBM WebSphere MQ . Supporta la comunicazione in entrata e in uscita.

“Impostazione post - installazione per le classi WebSphere MQ per le applicazioni JMS” a pagina 772
Questo argomento indica le autorizzazioni WebSphere MQ per le applicazioni JMS necessarie per accedere alle risorse di un gestore code. Inoltre, introduce le modalità di connessione e descrive come configurare un gestore code in modo che le applicazioni possano connettersi in modalità client.

“Il test di verifica dell'installazione point-to-point per WebSphere MQ classes per JMS” a pagina 776
Un programma IVT (point - to - point installation verification test) viene fornito con WebSphere MQ classes per JMS. Il programma si connette a un gestore code in modalità bind o client, invia un messaggio alla coda denominata SYSTEM.DEFAULT.LOCAL.QUEUEe riceve il messaggio dalla coda. Il programma può creare e configurare tutti gli oggetti richiesti in modo dinamico al runtime oppure può utilizzare JNDI per richiamare gli oggetti gestiti da un servizio di directory.

“Il test di verifica dell'installazione di pubblicazione / sottoscrizione per WebSphere MQ classes per JMS” a pagina 779
Un programma IVT (publish/subscribe installation verification test) viene fornito con WebSphere MQ classes per JMS. Il programma si connette a un gestore code in modalità bind o client, effettua la sottoscrizione a un argomento, pubblica un messaggio sull'argomento e riceve il messaggio appena pubblicato. Il programma può creare e configurare tutti gli oggetti richiesti in modo dinamico al runtime oppure può utilizzare JNDI per richiamare gli oggetti gestiti da un servizio di directory.

“Il programma di verifica dell'installazione per l'adattatore di risorse WebSphere MQ” a pagina 783
Il programma IVT viene fornito come file EAR. Per utilizzare il programma, è necessario distribuirlo e definire alcuni oggetti come risorse JCA.

“Configurazione dell'adattatore di risorse per la comunicazione in uscita” a pagina 753

Per configurare la comunicazione in uscita, definire le proprietà di un oggetto ConnectionFactory e di un oggetto di destinazione gestito.

“Supporto per OSGi” a pagina 796
OSGi fornisce un framework che supporta la distribuzione delle applicazioni come bundle. Nove bundle OSGi vengono forniti come parte di IBM WebSphere MQ classes for JMS .

“Risoluzione dei problemi con le classi IBM WebSphere MQ per JMS” a pagina 797
È possibile esaminare i problemi eseguendo i programmi di verifica dell'installazione e utilizzando le funzionalità di traccia e di log.

Attività correlate

“Installazione e verifica dell'adattatore di risorse MQ in WAS CE” a pagina 786
Installazione dell'adattatore di risorse IBM WebSphere MQ e esecuzione dell'applicazione IVT (Installation Verification Test) in WebSphere Application Server CE.

“Distribuzione dell'applicazione IVT su WAS CE con un ambiente MQ personalizzato” a pagina 788
Se si desidera utilizzare una coda, un gestore code, una porta, un host, un canale o utilizzare la modalità di bind anziché la modalità client, è necessario modificare l'applicazione IVT e gli script associati in WebSphere Application Server CE prima di distribuire l'adattatore risorse o l'applicazione IVT.

“Distribuzione dell'applicazione IVT in JBoss con un ambiente IBM WebSphere MQ personalizzato” a pagina 791
Quando si installa l'adattatore di risorse IBM WebSphere MQ in JBoss, se si desidera utilizzare una coda, un gestore code, una porta, un host, un canale o utilizzare la modalità di bind invece della modalità client, è necessario prima modificare l'applicazione IVT e gli script associati in JBoss prima di distribuire l'adattatore di risorse o l'applicazione IVT.

Determinazione dei problemi per l'adattatore di risorse IBM WebSphere MQ

Riferimenti correlati

“Script forniti con le classi WebSphere MQ per JMS” a pagina 795
Viene fornito un certo numero di script per assistere le attività comuni che devono essere eseguite quando si utilizzano le classi WebSphere MQ per JMS.

Cosa è installato per le classi IBM WebSphere MQ per JMS

Una serie di file e directory vengono creati quando si installano le classi IBM WebSphere MQ per JMS. Su Windows , alcune configurazioni vengono eseguite durante l'installazione impostando automaticamente le variabili di ambiente. Su altre piattaforme e in determinati ambienti Windows, è necessario impostare le variabili di ambiente prima di poter eseguire le classi IBM WebSphere MQ per applicazioni JMS.

Per la maggior parte dei sistemi operativi, le classi IBM WebSphere MQ per JMS vengono installate come un componente facoltativo quando si installa IBM WebSphere MQ. Per IBM WebSphere MQ client per HP Integrity NonStop Server, le classi IBM WebSphere MQ per JMS sono installate per impostazione predefinita. Per ulteriori informazioni sull'installazione di IBM WebSphere MQ, consultare:

- Installazione di un server WebSphere MQ
- Installazione di un client IBM WebSphere MQ

Tabella 90 a pagina 721 mostra dove sono installate le classi IBM WebSphere MQ per i file JMS su ciascuna piattaforma.

Tabella 90. Classi IBM WebSphere MQ per directory di installazione JMS

Piattaforma	**Cartella**
AIX	*MQ_INSTALLATION_PATH*/java
HP Integrity NonStop Server	*MQ_INSTALLATION_PATH*/opt/mqm/java
HP-UX, Linuxe Solaris	*MQ_INSTALLATION_PATH*/java
Windows	*MQ_INSTALLATION_PATH*\java

Tabella 90. Classi IBM WebSphere MQ per directory di installazione JMS (Continua)	
Piattaforma	**Cartella**
MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato IBM WebSphere MQ .	

La directory di installazione comprende:

- Le classi IBM WebSphere MQ per i file JAR JMS, che si trovano nella directory *MQ_INSTALLATION_PATH*\java\lib.
- Le librerie native di IBM WebSphere MQ , utilizzate dalle applicazioni che utilizzano Java Native Interface.

  Le librerie native a 32 bit sono installate nella directory *MQ_INSTALLATION_PATH*\java\lib e le librerie native a 64 bit sono disponibili nella directory *MQ_INSTALLATION_PATH*\java\lib64 .

  Per ulteriori informazioni sulle librerie native IBM WebSphere MQ , consultare “Configurazione delle librerie JNI (Java Native Interface)” a pagina 726.
- Ulteriori script descritti in “Script forniti con le classi WebSphere MQ per JMS” a pagina 795. Questi script si trovano nella directory *MQ_INSTALLATION_PATH*\java\bin.
- Le specifiche delle classi IBM WebSphere MQ per l'API JMS. Lo strumento Javadoc è stato utilizzato per generare le pagine HTML contenenti le specifiche dell'API.

  Le pagine HTML si trovano nella directory *MQ_INSTALLATION_PATH*\java\doc\WMQJMSClasses.

  Su sistemi UNIX, Linuxe Windows, questa directory secondaria contiene le singole pagine HTML.
- Supporto per OGSi. I bundle OSGi sono installati nella directory java \lib\OSGi e descritti in “Supporto per OSGi” a pagina 796.
- L'adattatore di risorse IBM WebSphere MQ , che può essere distribuito in qualsiasi server delle applicazioni compatibile con JCA 1.5 (o versioni successive).

  L'adattatore di risorse IBM WebSphere MQ si trova nella directory *MQ_INSTALLATION_PATH*\java\lib\jca; per ulteriori informazioni, consultare “L'adattatore di risorse IBM WebSphere MQ” a pagina 734
- Su Windows, i simboli che possono essere utilizzati per il debug sono installati nella directory *MQ_INSTALLATION_PATH*\java\lib\symbol.

La directory di installazione include anche alcuni file che appartengono ad altri componenti di IBM WebSphere MQ . Queste directory sono le seguenti:

- Il trasporto IBM WebSphere MQ per SOAP, che fornisce un trasporto JMS per SOAP, è installato nella directory *MQ_INSTALLATION_PATH*\java\lib\soap. Per ulteriori informazioni sul trasporto IBM WebSphere MQ per SOAP, consultare la sezione del centro informazioni che descrive “Trasporto WebSphere MQ per SOAP” a pagina 949.
- Su piattaforme distribuite, IBM WebSphere MQ Bridge for HTTP è installato nella directory *MQ_INSTALLATION_PATH*\java\lib\http. Per ulteriori informazioni sul bridge IBM WebSphere MQ per HTTP, consultare la sezione del centro informazioni che descrive “WebSphere MQ Bridge per HTTP” a pagina 1025

Alcune applicazioni di esempio vengono fornite con le classi IBM WebSphere MQ per JMS. Tabella 91 a pagina 722 mostra dove sono installate le applicazioni di esempio su ciascuna piattaforma.

Tabella 91. Directory di esempi	
Piattaforma	**Cartella**
AIX	*MQ_INSTALLATION_PATH*/samp/jms
HP Integrity NonStop Server	*MQ_INSTALLATION_PATH*/opt/mqm/samp/jms
HP-UX, Linuxe Solaris	*MQ_INSTALLATION_PATH*/samp/jms

Tabella 91. Directory di esempi (Continua)	
Piattaforma	**Cartella**
Windows	*MQ_INSTALLATION_PATH*\tools\jms
MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato IBM WebSphere MQ .	

Dopo l'installazione, potrebbe essere necessario eseguire alcune attività di configurazione per compilare ed eseguire le applicazioni.

"Variabili di ambiente utilizzate dalle classi IBM WebSphere MQ per JMS" a pagina 724 descrive il percorso classi richiesto per eseguire classi IBM WebSphere MQ semplici per le applicazioni JMS. Questo argomento descrive anche i file JAR aggiuntivi a cui è necessario fare riferimento in circostanze speciali e le variabili di ambiente che è necessario impostare per eseguire gli script forniti con le classi IBM WebSphere MQ per JMS.

Se è necessario che le classi IBM WebSphere MQ per l'applicazione JMS si colleghino al codice scritto in linguaggi diversi da Java (ad esempio, per utilizzare il trasporto dei collegamenti durante la connessione a un gestore code), "Configurazione delle librerie JNI (Java Native Interface)" a pagina 726 spiega dove trovare l'ubicazione delle librerie JNI (Java Native Interface) da specificare come parametro del comando Java.

Per controllare le proprietà, come la traccia e la registrazione di un'applicazione, è necessario fornire un file delle proprietà di configurazione. Il file delle proprietà di configurazione delle classi IBM WebSphere MQ per JMS è descritto in "Il file di configurazione IBM WebSphere MQ classes for JMS" a pagina 728.

Installazione e aggiornamento delle classi WebSphere MQ per i file JAR JMS

L'unico modo supportato per ottenere i file JAR di IBM WebSphere MQ per JMS su un sistema è installare il prodotto IBM WebSphere MQ o i client WebSphere MQ V7.5 SupportPac- MQC75o utilizzando uno strumento di gestione software come Apache Maven, per ulteriori informazioni consultare "IBM WebSphere MQ classes for JMS e strumenti di gestione software" a pagina 729.

Non spostare o copiare le classi IBM WebSphere MQ per i file JAR JMS o le librerie native in altre macchine o in un'ubicazione diversa su una macchina in cui sono state installate le classi IBM WebSphere MQ per JMS, a meno che non si utilizzi uno strumento di gestione software.

- I fix pack non possono essere applicati a una "installazione" in cui i file JAR sono stati copiati da un'altra macchina, perché ciò rende molto più difficile garantire che tutti i file JAR siano tenuti al passo tra loro e siano a livelli compatibili.
- La copia delle classi IBM WebSphere MQ per i file JAR JMS tra le macchine può anche risultare in più copie dei file che risiedono sulla stessa macchina, il che può causare problemi di manutenzione del codice e problemi di debug.
- Il comando dspmqver , utilizzato per visualizzare le informazioni sulla versione da una installazione di IBM WebSphere MQ , visualizza solo le informazioni sulla versione per le classi IBM WebSphere MQ per JMS installate nella directory `\java\lib` .

  Se più copie dei file si trovano sulla stessa macchina, l'esecuzione di **dspmqver** potrebbe non fornire informazioni precise sulla versione delle classi IBM WebSphere MQ per JMS utilizzate da un'applicazione.

Non includere le classi IBM WebSphere MQ per i file JAR JMS all'interno degli archivi dell'applicazione (come gli archivi dell'applicazione enterprise o i file EAR).

- Gli aggiornamenti alle classi IBM WebSphere MQ per JMS non possono essere applicati utilizzando un fix pack IBM WebSphere MQ .
- Non è possibile per il supporto IBM determinare facilmente la versione delle classi IBM WebSphere MQ per JMS utilizzate dall'applicazione.
- I problemi possono verificarsi se più applicazioni in esecuzione all'interno dello stesso JRE (Java Runtime Environment) includono versioni differenti delle classi IBM WebSphere MQ per JMS, poiché

più versioni delle classi IBM WebSphere MQ per JMS vengono caricate contemporaneamente in JRE (Java Runtime Environment).

Esempi di questi problemi includono le seguenti eccezioni:

```
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
```

- Se un'applicazione utilizza il trasporto BINDINGS per connettersi a un gestore code, qualsiasi aggiornamento principale al gestore code richiede anche che l'applicazione venga aggiornata per includere il livello corrispondente delle classi IBM WebSphere MQ per JMS.

  Ad esempio, se un gestore code viene aggiornato al livello IBM WebSphere MQ Versione 7.5 , anche le applicazioni che si connettono al gestore code utilizzando il trasporto BINDINGS devono essere aggiornate per includere le classi IBM WebSphere MQ Versione 7.5 per JMS.

Variabili di ambiente utilizzate dalle classi IBM WebSphere MQ per JMS

Prima di poter compilare ed eseguire le classi IBM WebSphere MQ per applicazioni JMS, l'impostazione per la variabile di ambiente CLASSPATH deve includere il file JAR (Java archive) IBM WebSphere MQ classes per JMS. A seconda dei requisiti, potrebbe essere necessario aggiungere altri file JAR al percorso di classe. Per eseguire gli script forniti con le classi di IBM WebSphere MQ per JMS, è necessario impostare altre variabili di ambiente.

Per compilare ed eseguire le classi IBM WebSphere MQ per le applicazioni JMS, utilizzare l'impostazione CLASSPATH per la piattaforma, come mostrato in Tabella 92 a pagina 724. L'impostazione include la directory degli esempi, in modo da poter compilare ed eseguire le classi IBM WebSphere MQ per le applicazioni di esempio JMS. In alternativa, è possibile specificare il percorso classe nel comando **java** invece di utilizzare la variabile di ambiente.

Tabella 92. Impostazione CLASSPATH per compilare ed eseguire classi IBM WebSphere MQ per applicazioni JMS, incluse le applicazioni di esempio

Piattaforma	**impostazione 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, Linuxe 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\ms;
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 rappresenta la directory di alto livello in cui è installato IBM WebSphere MQ .	

Il file manifest del file JAR com.ibm.mqjms.jar contiene riferimenti alla maggior parte degli altri file JAR richiesti dalle classi IBM WebSphere MQ per le applicazioni JMS, quindi non è necessario aggiungere tali file JAR al percorso di classe. Questi file JAR includono quelli richiesti dalle applicazioni che utilizzano JNDI (Java Naming and Directory Interface) per richiamare gli oggetti gestiti da un servizio di directory e dalle applicazioni che utilizzano JTA (Java Transaction API).

Tuttavia, è necessario includere ulteriori file JAR nel percorso di classe nelle seguenti circostanze:

- Se si utilizzano classi di uscita canale che implementano le interfacce di uscita canale definite nel package com.ibm.mq , invece di quelle definite nel package com.ibm.mq.exits , è necessario aggiungere le classi IBM WebSphere MQ per il file JAR Java, com.ibm.mq.jar, al percorso classe.
- Se si compila il codice Java utilizzando un Java 2 SDK (Software Development Kit) alla versione 1.4.2, è necessario aggiungere i seguenti file JAR al percorso di classe:
  - jms.jar
  - com.ibm.mq.jmqi.jar

  Inoltre, se l'applicazione utilizza JNDI per richiamare gli oggetti gestiti da un servizio directory, è necessario aggiungere anche i seguenti file JAR al percorso di classe:
  - fscontext.jar
  - jndi.jar
  - ldap.jar
  - providerutil.jar

  E se la tua applicazione utilizza JTA, devi anche aggiungere jta.jar al tuo percorso di classe.

  Notare che questi file JAR aggiuntivi sono richiesti solo per la compilazione delle proprie applicazioni, non per la relativa esecuzione.

Se si esegue la compilazione utilizzando l'opzione -Xlint, potrebbe essere visualizzato un messaggio che indica che com.ibm.mq.ese.jar non è presente. È possibile ignorare l'avvertenza. Questo file è presente solo se è stata installata Extended Security Edition.

Gli script forniti con le classi IBM WebSphere MQ per JMS utilizzano le variabili di ambiente riportate di seguito:

MQ_JAVA_DATA_PATH
Questa variabile di ambiente specifica la directory per l'output di log e traccia.

PERCORSO_INSTALL_JAVA_MQ_
Questa variabile di ambiente specifica la directory in cui è installato WebSphere MQ classes per JMS.

MQ_JAVA_LIB_PATH
Questa variabile di ambiente specifica la directory in cui sono memorizzate le librerie WebSphere MQ per JMS, come mostrato in Tabella 93 a pagina 726.

In Windows, tutte le variabili di ambiente vengono impostati automaticamente durante l'installazione. Su qualsiasi altra piattaforma, è necessario impostarli da soli.

Per impostare le variabili di ambiente se si utilizza una JVM a 32 bit su sistemi UNIX, HP Integrity NonStop Server o Linux , è possibile utilizzare lo script setjmsenv. Per impostare le variabili di ambiente se si utilizza una JVM a 64 bit su un sistema UNIX o Linux , è possibile utilizzare lo script setjmsenv64. Questi script si trovano nella directory *MQ_INSTALLATION_PATH*`/java/bin` , dove *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato IBM WebSphere MQ .

È possibile utilizzare lo script setjmsenv o setjmsenv64 in vari modi: è possibile utilizzarlo come base per l'impostazione delle variabili di ambiente richieste, come mostrato nella tabella, oppure aggiungerle a `.profile` utilizzando un editor di testo. Se si dispone di una configurazione non tipica, modificare il contenuto dello script come necessario. In alternativa, è possibile eseguire lo script in ogni sessione da cui devono essere eseguiti gli script di avvio JMS. Se si sceglie questa opzione, è necessario eseguire lo script in ogni finestra della shell avviata, durante il processo di verifica JMS immettendo `. ./setjmsenv` o `. ./setjmsenv64`

Configurazione delle librerie JNI (Java Native Interface)

Le classi IBM WebSphere MQ per le applicazioni JMS, che si connettono a un gestore code utilizzando il trasporto dei collegamenti o che si connettono a un gestore code utilizzando il trasporto client e utilizzano i programmi di uscita del canale scritti in linguaggi diversi da Java, devono essere eseguite in un ambiente che accede alle librerie JNI (Java Native Interface).

Informazioni su questa attività

Per impostare questo ambiente, è necessario configurare il percorso della libreria dell'ambiente in modo che la JVM (Java virtual machine) possa caricare la libreria mqjbnd prima di avviare le classi IBM WebSphere MQ per l'applicazione JMS.

IBM WebSphere MQ fornisce due librerie JNI (Java Native Interface):

mqjbnd
Questa libreria è utilizzata da applicazioni che si connettono a un gestore code utilizzando il trasporto dei bind. Fornisce l'interfaccia tra le classi IBM WebSphere MQ per JMS e il gestore code. La libreria mqjbnd installata con IBM WebSphere MQ Versione 7.5 può essere utilizzata per connettersi a qualsiasi gestore code IBM WebSphere MQ Versione 7.5 (o precedente).

mqjexitstub02
La libreria mqjexitstub02 viene caricata dalle classi di IBM WebSphere MQ per JMS quando un'applicazione si connette a un gestore code utilizzando il trasporto client e utilizza un programma di uscita del canale scritto in un linguaggio diverso da Java.

Su alcune piattaforme, IBM WebSphere MQ installa le versioni a 32 bit e a 64 bit di tali librerie JNI. L'ubicazione delle librerie per ciascuna piattaforma è mostrata nella Tabella 1

Tabella 93. L'ubicazione delle librerie IBM WebSphere MQ classes per JMS per ciascuna piattaforma

Piattaforma	**Directory contenente le librerie di IBM WebSphere MQ classes per JMS**
AIX	*MQ_INSTALLATION_PATH*/java/lib (librerie a 32 bit) *MQ_INSTALLATION_PATH*/java/lib64 (librerie a 64 bit)
HP-UX	*MQ_INSTALLATION_PATH*/java/lib (librerie a 32 bit) *MQ_INSTALLATION_PATH*/java/lib64 (librerie a 64 bit)
Linux (POTENZA , x86-64 e zSeries s390x piattaforme)	*MQ_INSTALLATION_PATH*/java/lib (librerie a 32 bit) *MQ_INSTALLATION_PATH*/java/lib64 (librerie a 64 bit)
Linux (piattaforma x86) Linux (zSeries piattaforma)	*MQ_INSTALLATION_PATH* /java/lib
Solaris (piattaformex86-64 e SPARC)	*MQ_INSTALLATION_PATH*/java/lib (librerie a 32 bit) *MQ_INSTALLATION_PATH*/java/lib64 (librerie a 64 bit)
Windows	*MQ_INSTALLATION_PATH*\java\lib (librerie a 32 bit) *MQ_INSTALLATION_PATH*\java\lib64 (librerie a 64 bit)
MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato IBM WebSphere MQ .	

Procedura

1. Configurare la proprietà **java.library.path** della JVM, che può essere eseguita in due modi:
   - Specificando l'argomento JVM come mostrato nel seguente esempio:

     ```
     -Djava.library.path=<path_to_library_directory>
     ```

     Linux Ad esempio, per una JVM a 64 bit su Linux per un'installazione di ubicazione predefinita, specificare:

     ```
     -Djava.library.path=/opt/mqm/java/lib64
     ```

   - Configurando l'ambiente della shell in modo che la JVM configuri il proprio `java.library.path`. Questo percorso varia in base alla piattaforma e all'ubicazione in cui è installato IBM WebSphere MQ. Ad esempio, per una JVM a 64 bit e un'ubicazione di installazione IBM WebSphere MQ predefinita, è possibile utilizzare le seguenti impostazioni:

     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%
     ```

   Di seguito è riportato un esempio dello stack delle eccezioni che viene visualizzato quando l'ambiente non è stato configurato correttamente:

   ```
   Causato da: com.ibm.mq.jmqi.local.LocalMQ$4: CC=2;RC=2495;
   AMQ8598: Impossibile caricare la libreria JNI nativa di WebSphere MQ : 'mqjbnd'.
       all'indirizzo com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1268)
       all'indirizzo com.ibm.mq.jmqi.local.LocalMQ$1.run(LocalMQ.java:309)
       in java.security.AccessController.doPrivileged(AccessController.java:400)
       all'indirizzo com.ibm.mq.jmqi.local.LocalMQ.initialise_inner(LocalMQ.java:259)
       all'ubicazione com.ibm.mq.jmqi.local.LocalMQ.initialise(LocalMQ.java:221)
       all'indirizzo com.ibm.mq.jmqi.local.LocalMQ. < init> (LocalMQ.java:1350)
       all'indirizzo com.ibm.mq.jmqi.local.LocalServer. < init> (LocalServer.java:230)
       in sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native nativo)
       in
   sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:86)

       all'indirizzo
   sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.jav
   a:58)
       all'indirizzo java.lang.reflect.Constructor.newInstance(Constructor.java:542)
       all'indirizzo com.ibm.mq.jmqi.JmqiEnvironment.getInstance(JmqiEnvironment.java:706)
       all'indirizzo com.ibm.mq.jmqi.JmqiEnvironment.getMQI(JmqiEnvironment.java:640)
       all'indirizzo
   com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createV7ProviderConnection(WMQConnectionF
   actory.java:8437)
       ... 7 more
   Causato da: java.lang.UnsatisfiedLinkError: mqjbnd (non trovato in java.library.path)
       all'indirizzo java.lang.ClassLoader.loadLibraryWithPath(ClassLoader.java:1235)
       all'indirizzo java.lang.ClassLoader.loadLibraryWithClassLoader(ClassLoader.java:1205)
       in java.lang.System.loadLibrary(System.java:534)
       all'indirizzo com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1240)
       ... Altri 20
   ```

2. Una volta impostato l'ambiente a 32 bit o a 64 bit, avviare le classi IBM WebSphere MQ per l'applicazione JMS utilizzando il comando:

   ```
   java application-name
   ```

   dove *nome - applicazione* è il nome delle classi IBM WebSphere MQ per l'applicazione JMS da eseguire.

   Un'eccezione contenente il codice di errore IBM WebSphere MQ 2495 (MQRC_MODULE_NOT_FOUND) viene generata dalle classi IBM WebSphere MQ per JMS se:

- Le classi IBM WebSphere MQ per l'applicazione JMS vengono eseguite in un ambiente di runtime Java a 32 bit e un ambiente a 64 bit è stato impostato per le classi IBM WebSphere MQ per JMS, poiché l'ambiente di runtime Java a 32 bit non è in grado di caricare la libreria nativa Java a 64 bit.
- Le classi IBM WebSphere MQ per l'applicazione JMS vengono eseguite in un ambiente di runtime Java a 64 bit e un ambiente a 32 bit è stato impostato per le classi IBM WebSphere MQ per JMS, poiché l'ambiente di runtime Java a 64 bit non è in grado di caricare la libreria nativa Java a 32 bit.

Il file di configurazione IBM WebSphere MQ classes for JMS

Un file di configurazione delle classi WebSphere MQ per JMS specifica le proprietà utilizzate per configurare le classi WebSphere MQ per JMS.

Il formato di un file di configurazione di WebSphere MQ classes per JMS è quello di un file delle proprietà Java standard. Un file di configurazione di esempio denominato jms.config viene fornito nella sottodirectory `bin` della directory di installazione di WebSphere MQ classes per JMS. Questo file documenta tutte le proprietà supportate e i relativi valori predefiniti.

È possibile scegliere nome e ubicazione di un file di configurazione WebSphere MQ classes per JMS. Quando si avvia l'applicazione, utilizzare un comando **java** con il seguente formato:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

Nel comando, *config_file_url* è un URL (uniform resource locator) che specifica il nome e l'ubicazione del file di configurazione di WebSphere MQ classes per JMS. Sono supportati URL dei seguenti tipi: http, file, ftp e jar.

Di seguito è riportato un esempio di comando **java** :

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config MyAppClass
```

Questo comando identifica il file di configurazione di WebSphere MQ classes for JMS come file D:\mydir\mjms.config sul sistema Windows locale.

Quando un'applicazione viene avviata, WebSphere MQ classes for JMS legge il contenuto del file di configurazione e memorizza le proprietà specificate in un archivio delle proprietà interno. Se il comando **java** non identifica un file di configurazione o se non è possibile trovare il file di configurazione, WebSphere MQ classes per JMS utilizza i valori predefiniti per tutte le proprietà. Se necessario, è possibile sovrascrivere qualsiasi proprietà nel file di configurazione specificandola come proprietà di sistema nel comando **java** .

Un file di configurazione di WebSphere MQ classes for JMS può essere utilizzato con uno qualsiasi dei trasporti supportati tra un'applicazione e un gestore code o broker.

Si noti che non è possibile specificare la traccia di avvio impostando una proprietà nel file di configurazione di WebSphere MQ classes per JMS. È possibile specificare la traccia di avvio solo impostando una proprietà di sistema nel comando **Java** , come mostrato nel seguente esempio:

```
java -Dcom.ibm.msg.client.commonservices.trace.startup=true
     -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config
      MyAppClass
```

Sovrascrittura delle proprietà specificate in un file di configurazione client WebSphere MQ MQI

Un file di configurazione client WebSphere MQ MQI può anche specificare le proprietà utilizzate per configurare le classi WebSphere MQ per JMS. Tuttavia, le proprietà specificate in un WebSphere MQ file di configurazione del client MQI si applicano solo quando un'applicazione si connette a un gestore code in modalità client.

Se necessario, è possibile sovrascrivere qualsiasi attributo in un file di configurazione del client WebSphere MQ MQI specificandolo come proprietà in un file di configurazione WebSphere MQ classes

per JMS. Per sovrascrivere un attributo in un file di configurazione client WebSphere MQ , utilizzare una voce con il formato seguente nel file di configurazione di WebSphere MQ classes per JMS:

```
com.ibm.mq.cfg.stanza.propName=propValue
```

Le variabili nella voce hanno i seguenti significati:

stanza
Il nome della sezione nel file di configurazione del client WebSphere MQ MQI che contiene l'attributo

propName
Il nome dell'attributo come specificato nel file di configurazione del client WebSphere MQ

propValue
Il valore della proprietà che sovrascrive il valore dell'attributo specificato nel file di configurazione del client MQI WebSphere MQ

In alternativa, è possibile sovrascrivere un attributo in un file di configurazione client WebSphere MQ MQI specificando la proprietà come proprietà di sistema nel comando **java** . Utilizzare il formato precedente per specificare la proprietà come proprietà di sistema.

Solo i seguenti attributi in un file di configurazione client WebSphere MQ MQI sono rilevanti per le classi WebSphere MQ per JMS. Se si specificano o si sovrascrivono altri attributi, non ha alcun effetto.

stanza	Attributo
ClientExitStanza del percorso del file di configurazione client	ExitsDefaultPath
ClientExitStanza del percorso del file di configurazione client	ExitsDefaultPath64
ClientExitStanza del percorso del file di configurazione client	Percorso classi JavaExits
Stanza MessageBuffer del file di configurazione client	MaximumSize
Stanza MessageBuffer del file di configurazione client	PurgeTime
Stanza MessageBuffer del file di configurazione client	UpdatePercentage
Stanza TCP del file di configurazione client	ClntRcvBufSize
Stanza TCP del file di configurazione client	ClntSndBufSize
Stanza TCP del file di configurazione client	Timeout connessione
Stanza TCP del file di configurazione client	KeepAlive

IBM WebSphere MQ classes for JMS e strumenti di gestione software

Gli strumenti di gestione software come Apache Maven possono essere utilizzati con IBM WebSphere MQ classes for JMS.

Molte grandi organizzazioni di sviluppo utilizzano questi strumenti per gestire centralmente i repository di librerie di terze parti.

I IBM WebSphere MQ classes for JMS sono composti da un numero di file JAR. Quando si sviluppano applicazioni in linguaggio Java utilizzando questa API, è richiesta un'installazione di un IBM WebSphere MQ Server, Client o Client SupportPac sulla macchina su cui è in fase di sviluppo l'applicazione.

Se si desidera utilizzare tale strumento e aggiungere i file JAR che costituiscono IBM WebSphere MQ classes for JMS a un repository gestito centralmente, è necessario osservare i seguenti punti:

- Un repository o un contenitore deve essere reso disponibile solo agli sviluppatori all'interno della propria organizzazione. Non è consentita alcuna distribuzione al di fuori dell'organizzazione.
- Il repository deve contenere una serie completa e coerente di file JAR da una singola release IBM WebSphere MQ o Fix Pack.
- Sei responsabile dell'aggiornamento del repository con qualsiasi manutenzione fornita dal supporto IBM .

Per IBM WebSphere MQ Version 7.5, i seguenti file JAR devono essere installati nel repository:

- com.ibm.mqjms.jar.
- com.ibm.mq.jar
- com.ibm.mq.jmqi.jar
- com.ibm.mq.pcf.jar
- com.ibm.mq.headers.jar
- CL3Export.jar è obbligatorio se si utilizza IBM WebSphere MQ classes for JMS.
- CL3Nonexport.jar è obbligatorio se si utilizza IBM WebSphere MQ classes for JMS.
- jndi.jar è richiesto se si utilizza IBM WebSphere MQ classes for JMS.
- ldap.jar è richiesto se si utilizza IBM WebSphere MQ classes for JMS.
- rmm.jar è richiesto se si sta utilizzando IBM WebSphere MQ classes for JMS.
- dhbcore.jar è richiesto se si utilizza IBM WebSphere MQ classes for JMS.
- jms.jar è richiesto se si utilizza IBM WebSphere MQ classes for JMS.
- fscontext.jar è richiesto se si utilizza IBM WebSphere MQ classes for JMS e si accede agli oggetti amministrati JMS memorizzati in un contesto JNDI del file system.
- providerutil.jar se si sta utilizzando IBM WebSphere MQ classes for JMS e si accede agli oggetti amministrati JMS memorizzati in un contesto JNDI del file system.

Esecuzione delle classi WebSphere MQ per applicazioni JMS in Java Security Manager

WebSphere MQ classes for JMS può essere eseguito con il gestore sicurezza Java abilitato. Per eseguire correttamente le applicazioni con il gestore della protezione abilitato, è necessario configurare la JVM (Java virtual machine) con un file di configurazione della politica adatto.

Il modo più semplice per eseguire questa operazione è modificare il file di configurazione della politica fornito con il JRE. Sulla maggior parte dei sistemi, questo file è memorizzato nel percorso `lib/security/java.policy`, relativo alla directory JRE. È possibile modificare il file di configurazione della politica utilizzando l'editor preferito o il programma `policytool` fornito con JRE.

Importante: V 7.5.0.8 Ovunque possibile, il termine *allowlist* ha sostituito il termine *whitelist*. Un'eccezione è rappresentata dai seguenti nomi di proprietà di sistema Java .

Se si utilizza il meccanismo del gestore della sicurezza Java con l'applicazione, è necessario concedere le seguenti autorizzazioni:

- FilePermission su qualsiasi file allowlist utilizzato, con autorizzazione di lettura per la modalità ENFORCEMENT, autorizzazione di scrittura per la modalità DISCOVER.
- PropertyPermission (read) nelle proprietà com.ibm.mq.jms.whitelist, com.ibm.mq.jms.whitelist.discovere com.ibm.mq.jms.whitelist.mode .

ClassName allowlisting è supportato con APAR IT14385 e IBM WebSphere MQ Version 7.5.0, Fix Pack 8. Per ulteriori informazioni, consultare “ClassName allowlisting in JMS ObjectMessage” a pagina 731.

Di seguito è riportato un esempio di due voci in un file di configurazione della politica che consentono alle classi WebSphere MQ per JMS di essere eseguite correttamente con il gestore di sicurezza predefinito:

```
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";
};
```

Note:

- *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ .
- La prima istruzione `grant` contiene le autorizzazioni richieste dalle classi WebSphere MQ per JMS e la seconda istruzione `grant` contiene le autorizzazioni richieste da un'applicazione WebSphere MQ .
- Per consentire alle classi WebSphere MQ per JMS di accedere a file JAR (Java archive) di un'applicazione, aggiungere la seguente autorizzazione alla prima istruzione `grant` :

  ```
  permission java.io.FilePermission "/path_to_your_app/-", "read";
  ```

- Per utilizzare queste istruzioni `grant` nel file di configurazione della politica, potrebbe essere necessario modificare i nomi percorso in base a dove sono state installate le classi WebSphere MQ per JMS e dove sono state memorizzate le applicazioni.
- Le applicazioni di esempio fornite con WebSphere MQ classes for JMS, e gli script per eseguirle, non abilitano il gestore della sicurezza.

V 7.5.0.8 ClassName allowlisting in JMS ObjectMessage

V 7.5.0.8 Nelle classi WebSphere MQ per JMS, il supporto per consentire l'elenco delle classi nell'implementazione dell'interfaccia ObjectMessage JMS fornisce una potenziale attenuazione rispetto ad alcuni dei rischi di sicurezza potenzialmente correlati al meccanismo di serializzazione e deserializzazione degli oggetti Java.

Nota: Ovunque possibile, il termine *allowlist* ha sostituito il termine *whitelist*. Fanno eccezione i nomi di proprietà del sistema Java indicati in questo argomento.

Il meccanismo di serializzazione e deserializzazione degli oggetti Java è stato identificato come un potenziale rischio per la sicurezza perché la deserializzazione crea un'istanza di oggetti Java arbitrari,

dove è possibile che i dati inviati in modo dannoso causino vari problemi. Un'applicazione notevole della serializzazione si trova in ObjectMessages di Java Message Service (JMS) che utilizzano la serializzazione per incapsulare e trasferire oggetti arbitrari.

La serializzazione allowlisting è una potenziale attenuazione rispetto ad alcuni dei rischi che la serializzazione comporta. Specificando esplicitamente quali classi possono essere incapsulate ed estratte da ObjectMessages, allowlisting fornisce una protezione contro alcuni rischi di serializzazione.

Elenco consentito nelle classi WebSphere MQ per JMS

Con APAR IT14385 e IBM WebSphere MQ Version 7.5.0, Fix Pack 8, WebSphere le classi di MQ per JMS supportano l'elenco delle classi nell'implementazione dell'interfaccia ObjectMessage JMS. Allowlist definisce quali classi Java possono essere serializzate con ObjectMessage.setObject() e deserializzate con ObjectMessage.getObject().

I tentativi di serializzazione o deserializzazione di un'istanza di una classe non inclusa nella allowlist con ObjectMessage causano l'emissione di una javax.jms.MessageFormatException , con una java.io.InvalidClassException come causa.

Produzione della allowlist

Importante: Le classi WebSphere MQ per JMS non possono essere distribuite con un allowlist. La scelta delle classi da trasferire utilizzando ObjectMessages è una scelta di progettazione dell'applicazione e IBM WebSphere MQ non può anticipare tale scelta.

Per questo motivo, il meccanismo di elenco consentito consente due modalità di funzionamento:

Rilevamento
In questa modalità, il meccanismo produce un elenco di nomi classe completi, riportando tutte le classi che sono state serializzate o deserializzate in ObjectMessages.

Implementazione
In questa modalità, il meccanismo applica allowlisting, rifiutando i tentativi di serializzazione o deserializzazione delle classi che non sono nell'allowlist.

Se si desidera utilizzare questo meccanismo, è necessario eseguire inizialmente in modalità DISCOVERY per raccogliere l'elenco delle classi attualmente serializzate e deserializzate, rivedere l'elenco e utilizzarlo come base per l'elenco consentito. Potrebbe anche essere appropriato utilizzare l'elenco invariato, ma l'elenco deve essere rivisto prima di decidere di farlo.

Controllo del meccanismo allowlisting

Sono disponibili tre proprietà di sistema per controllare il meccanismo allowlisting:

com.ibm.mq.jms.whitelist
Questa proprietà può essere specificata in uno dei seguenti modi:

- Il nome del percorso del file che contiene l'elenco consentiti, in formato URI del file (che inizia con `file:`). In modalità DISCOVERY, questo file viene scritto dal meccanismo allowlisting. Il file non deve esistere. Se il file esiste, il meccanismo genera un'eccezione invece di sovrascriverla. In modalità ENFORCEMENT, questo file viene letto dal meccanismo allowlisting.
- Una virgola separata di nomi di classe completi che costituiscono l'elenco consentiti.

Se questa proprietà non è impostata, il meccanismo allowlist è inattivo.

Se si utilizza un gestore della sicurezza Java , è necessario assicurarsi che i file JAR di WebSphere MQ per JMS abbiano accesso in lettura e scrittura a questo file.

com.ibm.mq.jms.whitelist.discover

- Se questa proprietà non è impostata o è impostata su false, il meccanismo allowlist viene eseguito in modalità ENFORCEMENT.

- Se questa proprietà è impostata su true e l'allowlist è stato specificato come URI del file, il meccanismo allowlist viene eseguito in modalità DISCOVERY.
- Se questa proprietà è impostata su true e l'allowlist è stato specificato come un elenco di nomi classe, il meccanismo allowlist genera un'eccezione appropriata.
- Se questa proprietà è impostata su true e l'allowlist non è stata specificata utilizzando la proprietàcom.ibm.mq.jms.whitelist, il meccanismo allowlist non è attivo.
- Se questa proprietà è impostata su true e il file allowlist esiste già, il meccanismo allowlist genera una java.io.InvalidClassException e le voci non vengono aggiunte al file.

com.ibm.mq.jms.whitelist.mode
Questa proprietà stringa può essere specificata in uno dei seguenti tre modi:

- Se questa proprietà è impostata su SERIALIZE, la modalità ENFORCEMENT esegue la convalida allowlist solo sul metodo ObjectMessage.setObject().
- Se questa proprietà è impostata su DESERIALIZE, la modalità ENFORCEMENT esegue la convalida allowlist solo sul metodo ObjectMessage.getObject().
- Se questa proprietà non è impostata o è impostata su qualsiasi altro valore, la modalità ENFORCEMENT esegue la convalida allowlist sia sui metodi ObjectMessage.getObject() che ObjectMessage.setObject().

Formato del file allowlist

Queste sono le caratteristiche principali del formato del file allowlist:

- Il file allowlist si trova nella codifica del file della piattaforma predefinita con le terminazioni di riga appropriate per la piattaforma.

  Nota: Il file potrebbe richiedere la conversione se lo si sposta tra sistemi eterogenei.
- Ogni riga non vuota contiene un nome classe completo. Le righe vuote vengono ignorate.
- I commenti possono essere inclusi - tutto ciò che segue un carattere '#', alla fine della riga, viene ignorato.
- Esiste un meccanismo di caratteri jolly di base:
  - '*' può essere l'ultimo elemento di un nome classe.
  - '*' corrisponde a un elemento **singolo** di un nome classe, vale a dire, la classe, ma non fa parte del package.

  Quindi, `com.ibm.mq.*` corrisponderebbe `com.ibm.mq.MQMessage` ma non `com.ibm.mq.jmqi.remote.api.RemoteFAP`.

  Il carattere jolly non funziona per le classi nel pacchetto predefinito che è per le classi senza un nome pacchetto esplicito, quindi un nome classe di "*" viene rifiutato.
- I file allowlist formattati in modo non corretto, ad esempio i file che contengono una voce come `com.ibm.mq.*.Message`, in cui il carattere jolly non è l'ultimo elemento, generano una java.lang.IllegalArgumentException .
- Un file allowlist vuoto ha l'effetto di disabilitare completamente l'utilizzo di ObjectMessage.

Formato dell'elenco consentiti come elenco separato da virgole

Lo stesso meccanismo di caratteri jolly è disponibile per un allowlist come elenco separato da virgole.

- Il carattere '*' può essere espanso dal sistema operativo se specificato su una riga comandi o in uno script shell o in un file batch, quindi potrebbe richiedere una gestione speciale.
- Il carattere di commento '#' è applicabile solo quando viene specificato un file. Se l'allowlist viene specificato come un elenco separato da virgole di nomi classe, supponendo che il sistema operativo o la shell non lo elaborino, poiché si tratta del carattere di commento predefinito in molte shell UNIX o Linux , viene considerato come un carattere normale.

Quando avviene la creazione di un elenco?

Allowlisting viene avviato quando l'applicazione esegue per la prima volta un metodo ObjectMessage setMessage() o getMessage().

Le proprietà di sistema vengono valutate, il file allowlist viene aperto e, in modalità ENFORCEMENT, l'elenco delle classi consentite viene caricato quando il meccanismo viene inizializzato. A questo punto, viene scritta una voce nel file di log JMS IBM WebSphere MQ per l'applicazione.

Quando il meccanismo viene inizializzato, i relativi parametri potrebbero non essere modificati. Poiché il tempo di inizializzazione non è facilmente previsto in quanto dipende dal funzionamento dell'applicazione. Le impostazioni delle proprietà di sistema e il contenuto del file allowlist devono quindi essere considerati fissi dal momento in cui viene avviata l'applicazione. Non modificare le proprietà o il contenuto del file allowlist mentre l'applicazione è in esecuzione, poiché i risultati non sono garantiti.

Punti da considerare

L'approccio migliore per mitigare i rischi intrinseci al meccanismo di serializzazione Java consiste nell'esplorare approcci alternativi al trasferimento dei dati, ad esempio l'utilizzo di JSON invece di ObjectMessage. Utilizzando i meccanismi AMS (IBM WebSphere MQ Advanced Message Security) è possibile aggiungere ulteriore sicurezza assicurando che i messaggi provengano da origini attendibili.

Se si utilizza il meccanismo del gestore della sicurezza Java con l'applicazione, è necessario concedere le seguenti autorizzazioni:

- FilePermission su qualsiasi file allowlist utilizzato, con autorizzazione di lettura per la modalità ENFORCEMENT, autorizzazione di scrittura per la modalità DISCOVER.
- PropertyPermission (read) nelle proprietà com.ibm.mq.jms.whitelist, com.ibm.mq.jms.whitelist.discovere com.ibm.mq.jms.whitelist.mode .

Concetti correlati

“Esecuzione delle classi WebSphere MQ per applicazioni JMS in Java Security Manager” a pagina 730
WebSphere MQ classes for JMS può essere eseguito con il gestore sicurezza Java abilitato. Per eseguire correttamente le applicazioni con il gestore della protezione abilitato, è necessario configurare la JVM (Java virtual machine) con un file di configurazione della politica adatto.

L'adattatore di risorse IBM WebSphere MQ

L'adattatore risorse consente alle applicazioni in esecuzione in un application server di accedere alle risorse IBM WebSphere MQ . Supporta la comunicazione in entrata e in uscita.

Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) fornisce un modo standard per connettere le applicazioni in esecuzione in un ambiente Java EE a un EIS (Enterprise Information System) come IBM WebSphere MQ o Db2. L'adattatore di risorse IBM WebSphere MQ implementa le interfacce JCA 1.5 e contiene IBM WebSphere MQ classes for JMS. Consente alle applicazioni JMS e agli MDB (message driven bean), in esecuzione su un server delle applicazioni, di accedere alle risorse di un gestore code IBM WebSphere MQ . L'adattatore risorse supporta sia il dominio point-to-point che il dominio di pubblicazione / sottoscrizione.

L'adattatore di risorse IBM WebSphere MQ supporta due tipi di comunicazione tra un'applicazione e un gestore code:

Comunicazione in uscita
Un'applicazione avvia una connessione a un gestore code, quindi invia messaggi JMS alle destinazioni JMS e riceve messaggi JMS dalle destinazioni JMS in modo sincrono.

Comunicazione in ingresso
Un messaggio JMS che arriva a una destinazione JMS viene consegnato a un MDB, che elabora il messaggio in modo asincrono.

Per ulteriori informazioni su IBM WebSphere MQ classes for JMS, consultare “Utilizzo delle classi WebSphere MQ per JMS” a pagina 717.

L'adattatore di risorse contiene anche IBM WebSphere MQ classes for Java. Le classi sono automaticamente disponibili per le applicazioni in esecuzione in un application server in cui è stato distribuito l'adattatore di risorse e consentono alle applicazioni in esecuzione in tale application server di utilizzare l'API IBM WebSphere MQ classes for Java durante l'accesso alle risorse di un gestore code IBM WebSphere MQ . Per ulteriori informazioni su IBM WebSphere MQ classes for Java, consultare “Utilizzo di classi WebSphere MQ per Java” a pagina 654.

L'utilizzo di IBM WebSphere MQ classes for Java in ambiente Java EE è supportato con limitazioni. Per informazioni su queste restrizioni, consultare “Esecuzione di classi IBM WebSphere MQ per applicazioni Java nella piattaforma Java Enterprise Edition” a pagina 715.

Altra documentazione richiesta per supportare un adattatore di risorse JCA

Fare riferimento alla documentazione del server delle applicazioni per informazioni su come configurare un adattatore risorse JCA.

Ogni server delle applicazioni fornisce la propria serie di interfacce di amministrazione. Alcuni server delle applicazioni forniscono interfacce utente grafiche per definire le risorse JCA, ma altri richiedono che l'amministratore scriva piani di distribuzione XML. Pertanto, è al di là dell'ambito di questa documentazione per fornire informazioni su come configurare l'adattatore di risorse WebSphere MQ per ciascun server delle applicazioni. Questa documentazione si concentra solo su ciò che è necessario configurare. Fare riferimento alla documentazione fornita con il server delle applicazioni per informazioni su come configurare un adattatore risorse JCA.

Per comprendere questa documentazione, è necessario conoscere le classi JMS e WebSphere MQ per JMS. Molte delle propriet ... utilizzate per configurare l'adattatore di risorse WebSphere MQ sono equivalenti alle propriet ... delle classi WebSphere MQ per gli oggetti JMS e hanno la stessa funzione.

Installazione dell'adattatore di risorse WebSphere MQ

L'adattatore di risorse WebSphere MQ viene fornito come file RAR (resource archive). Installare il file RAR nel server delle applicazioni. Potrebbe essere necessario aggiungere directory al percorso di sistema.

L'adattatore di risorse WebSphere MQ viene fornito come un file RAR (Resource Archive) denominato wmq.jmsra.rar. Questo file è installato con WebSphere MQ classes for JMS nella directory illustrata nella Tabella 94 a pagina 735.

Tabella 94. La directory contenente wmq.jmsra.rar per ogni piattaforma

Piattaforma	**Cartella**
AIX, HP-UX, Linux e Solaris	*MQ_INSTALLATION_PATH* /java/lib/jca
IBM i	/QIBM/ProdData/mqm/java/lib/jca
Finestre	*MQ_INSTALLATION_PATH* \java\lib\jca
MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .	

Il file RAR contiene le classi WebSphere MQ per JMS e l'implementazione WebSphere MQ delle interfacce JCA.

È necessario installare il file RAR dell'adattatore risorse WebSphere MQ nel server delle applicazioni, ma il modo in cui si esegue questa operazione dipende dal server delle applicazioni. Consultare la documentazione per il proprio server delle applicazioni per informazioni su come installare un file RAR dell'adattatore risorse.

Per i collegamenti sui sistemi UNIX and Linux , verificare che la directory contenente le librerie JNI (Java Native Interface) si trovi nel percorso di sistema. Per l'ubicazione di questa directory, che contiene anche WebSphere MQ classes for JMS libraries, consultare Tabella 93 a pagina 726. In Windows, questa directory viene aggiunta automaticamente al percorso di sistema durante l'installazione di WebSphere MQ classes per JMS.

Le transazioni sono supportate sia in modalità client che bind.

L'adattatore di risorse WebSphere MQ e la versione delle classi WebSphere MQ per JMS utilizzate dall'adattatore di risorse devono essere allo stesso livello di rilascio.

WebSphere Application Server e l'adattatore di risorse WebSphere MQ
Non utilizzare l'adattatore risorse WebSphere MQ con WebSphere Application Server Versione 6. WebSphere Application Server, V7, include una versione dell'adattatore di risorsa WebSphere MQ V7 .

Non utilizzare l'adattatore di risorse WebSphere MQ all'interno di WebSphere Application Server, V6. Per accedere alle risorse di un gestore code WebSphere MQ dall'interno di WebSphere Application Server da un'applicazione JMS, utilizzare il provider di messaggistica WebSphere MQ . Il fornitore di messaggistica WebSphere MQ contiene una versione di WebSphere MQ classes per JMS.

WebSphere Application Server, V7, include una versione dell'adattatore di risorsa WebSphere MQ V7 .

Per ulteriori informazioni, consultare la technote Quale versione di WebSphere MQ Resource Adapter (RA) viene fornita con WebSphere Application Server ?.

WebSphere Application Server Liberty e l'adattatore di risorse IBM WebSphere MQ
L'adattatore di risorse IBM WebSphere MQ Version 7.5 può essere installato in WebSphere Application Server Liberty versione 8.5.5, Fix Pack 2 o versioni successive, utilizzando la funzione wmqJmsClient-1.1 . In alternativa, in base ad alcune limitazioni, è possibile installare l'adattatore di risorse utilizzando il supporto JCA (Java EE JCA) di Java Platform, Enterprise Edition generico.

Limitazioni generali quando si installa l'adattatore di risorse in Liberty

Le seguenti restrizioni si applicano all'adattatore di risorse Version 7.5 quando si utilizza la funzione wmqJmsClient-1.1 e anche quando si utilizza il supporto JCA generico:

- I IBM WebSphere MQ classes for Java non sono supportati in Liberty. Non devono essere utilizzati con la funzionalità di messaggistica IBM WebSphere MQ Liberty o con il supporto JCA generico. Per ulteriori informazioni, consultare Using WebSphere MQ Java Interfaces in J2EE/JEE Environments.
- L'adattatore di risorse IBM WebSphere MQ ha un tipo di trasporto BINDINGS_THEN_CLIENT. Questo tipo di trasporto non è supportato nella funzione di messaggistica IBM WebSphere MQ Liberty.
- La funzione IBM WebSphere MQ Advanced Message Security (IBM WebSphere MQ AMS) non è inclusa nella funzione di messaggistica IBM WebSphere MQ Liberty.

L'adattatore risorse IBM WebSphere MQ Version 7.5 non può essere utilizzato con la funzione wmqJmsClient-2.0 .

Configurazione dell'adattatore di risorse WebSphere MQ

Per configurare l'adattatore di risorse WebSphere MQ , definire diverse risorse JCA e proprietà di sistema.

Definire le risorse JCA nelle categorie seguenti:

- Le proprietà dell'oggetto ResourceAdapter , che rappresenta le proprietà globali dell'adattatore risorse, ad esempio il livello di traccia diagnostica. Tali proprietà sono descritte in “Configurazione dell'oggetto ResourceAdapter” a pagina 737.
- Le proprietà di un oggetto ActivationSpec , che determinano il modo in cui un MDB viene attivato per la comunicazione in entrata. Tali proprietà sono descritte in “Configurazione dell'adattatore di risorse per la comunicazione in entrata” a pagina 739.
- Le propriet ... di un oggetto ConnectionFactory , che il server delle applicazioni utilizza per creare un oggetto ConnectionFactory JMS per la comunicazione in uscita. Tali proprietà sono descritte in “Configurazione dell'adattatore di risorse per la comunicazione in uscita” a pagina 753.
- Le proprietà di un oggetto di destinazione gestito, che il server delle applicazioni utilizza per creare un oggetto coda JMS o un oggetto argomento JMS per la comunicazione in uscita. Queste proprietà sono descritte anche in “Configurazione dell'adattatore di risorse per la comunicazione in uscita” a pagina 753.

Il file RAR dell'adattatore risorse WebSphere MQ contiene un file denominato META-INF/ra.xml, che contiene un descrittore di distribuzione per l'adattatore risorse. Questo descrittore di distribuzione

è definito dallo schema XML in https://java.sun.com/xml/ns/j2ee/connector_1_5.xsd e contiene informazioni sull'adattatore di risorse e sui servizi che fornisce. Un server delle applicazioni potrebbe anche richiedere un piano di distribuzione per l'adattatore di risorse. Questo piano di distribuzione è specifico del server delle applicazioni. Ad esempio, WebSphere Application Server Community Edition richiede un piano di distribuzione denominato geronimo-ra.xml.

Se si utilizza SSL (Secure Sockets Layer), specificare le ubicazioni del file keystore e del file truststore come proprietà di sistema JVM, come nel seguente esempio:

```
java ... -Djavax.net.ssl.keyStore=
key_store_location
         -Djavax.net.ssl.trustStore=trust_store_location
         -Djavax.net.ssl.keyStorePassword=key_store_password
```

Queste proprietà non possono essere proprietà di un oggetto ActivationSpec o ConnectionFactory e non è possibile specificare più di un keystore per un server delle applicazioni. Le propriet ... si applicano all'intera JVM e, pertanto, potrebbero influire sul server delle applicazioni se altre applicazioni, in esecuzione sul server delle applicazioni, utilizzano connessioni SSL. Il server delle applicazioni potrebbe anche reimpostare queste proprietà su valori differenti. Per ulteriori informazioni sull'utilizzo di SSL con le classi WebSphere MQ per JMS, consultare “Utilizzo di SSL (Secure Sockets Layer) con le classi WebSphere MQ per JMS” a pagina 907.

Un programma IVT (Installation Verification Test) viene fornito con l'adattatore di risorse WebSphere MQ , ma è necessario configurare l'adattatore di risorse prima di eseguire il programma. Per informazioni su cosa è necessario configurare per eseguire il programma IVT, consultare “Il programma di verifica dell'installazione per l'adattatore di risorse WebSphere MQ” a pagina 783.

I log dell'adattatore di risorse, i messaggi di avvertenza e di errore utilizzano lo stesso meccanismo delle classi IBM WebSphere MQ per JMS, per i dettagli, consultare “Registrazione e IBM WebSphere MQ classes for JMS” a pagina 797. Per WebSphere Application Server, questi messaggi vengono automaticamente reindirizzati al log di output del server di applicazioni. Per altri server delle applicazioni, come WAS CE e JBoss, questi, per impostazione predefinita, andranno a un file denominato mqjms.log. Per configurare l'adattatore di risorse per registrare ulteriormente i messaggi di avvertenza nel log di output standard dei propri server delle applicazioni, impostare la seguente proprietà di sistema JVM per il proprio server delle applicazioni:

```
-Dcom.ibm.msg.client.commonservices.log.outputName=mqjms.log,stdout
```

Per i dettagli su come impostare una proprietà di sistema JVM, consultare la documentazione del server delle applicazioni.

Configurazione dell'oggetto ResourceAdapter
L'oggetto ResourceAdapter incapsula le proprietà globali dell'adattatore di risorse WebSphere MQ . Definire queste proprietà utilizzando le funzioni dell'adattatore di risorse.

L'oggetto ResourceAdapter ha due serie di proprietà:

- Proprietà associate alla traccia diagnostica
- Proprietà associate al pool di connessione gestito dall'adattatore di risorse

Il modo in cui si definiscono queste proprietà dipende dalle interfacce di gestione fornite dal server delle applicazioni.

Per ulteriori informazioni sulla definizione delle proprietà associate alla traccia diagnostica, consultare Traccia dell'adattatore di risorse IBM WebSphere MQ.

L'adattatore di risorse gestisce un pool di connessione interno di connessioni JMS utilizzate per consegnare i messaggi agli MDB Tabella 95 a pagina 738 elenca le proprietà dell'oggetto ResourceAdapter associate al pool di connessioni.

Tabella 95. Proprietà dell'oggetto ResourceAdapter associate al lotto connessioni			
Nome della proprietà	**Tipo**	**Valore predefinito**	**Descrizione**
maxConnections	Stringa	50	Il numero massimo di connessioni a un gestore code WebSphere MQ e il numero massimo di MDB distribuiti.
connectionConcurrency	Stringa	1	Il numero massimo di MDB per condividere una connessione JMS. La condivisione delle connessioni non è possibile e questa proprietà ha sempre il valore 1.
Conteggio reconnectionRetry	Stringa	5	Il numero massimo di tentativi effettuati dall'adattatore di risorse per riconnettersi a un gestore code WebSphere MQ in caso di errore di una connessione.
Intervallo reconnectionRetry	Stringa	300 000	Il tempo, in millisecondi, che l'adattatore di risorse attende prima di provare a riconnettersi a un gestore code WebSphere MQ .
Conteggio startupRetry	Stringa	0	Il numero predefinito di tentativi di connessione di un MDB all'avvio, se il gestore code non è in esecuzione quando viene avviato il server delle applicazioni.
Intervallo startupRetry	Stringa	30 000	Il tempo di sospensione predefinito tra i tentativi di avvio della connessione (in millisecondi).

Quando un MDB viene distribuito nel server delle applicazioni, viene creato un nuovo collegamento JMS e viene avviata una conversazione con il gestore code, purché non venga superato il numero massimo di connessioni specificato dalla proprietà maxConnection . Il numero massimo di MDB è quindi uguale al numero massimo di connessioni. Se il numero di MDB distribuiti raggiunge questo valore massimo, qualsiasi tentativo di distribuire un altro MDB ha esito negativo. Se un MDB viene arrestato, la sua connessione può essere utilizzata da un altro MDB.

In genere, se devono essere distribuiti molti MDB, è necessario aumentare il valore della proprietà maxConnections .

Le proprietà reconnectionRetryConteggio e reconnectionRetryIntervallo regolano il comportamento dell'adattatore di risorse quando le connessioni a un gestore code WebSphere MQ hanno esito negativo, ad esempio a causa di un errore di rete. Quando una connessione non riesce, l'adattatore di risorse sospende la consegna dei messaggi a tutti gli MDB forniti da tale connessione per un intervallo specificato dalla proprietà reconnectionRetryInterval. L'adattatore risorse tenta quindi di riconnettersi al gestore code. Se il tentativo ha esito negativo, l'adattatore di risorse effettua ulteriori tentativi di riconnessione a intervalli specificati dalla proprietà reconnectionRetryfino a quando non viene raggiunto il limite imposto dalla proprietà Conteggio reconnectionRetry. Se tutti i tentativi non riescono, la consegna viene arrestata in modo permanente fino a quando gli MDB non vengono riavviati manualmente.

In generale, l'oggetto ResourceAdapter non richiede alcuna gestione. Tuttavia, ad esempio, per abilitare la traccia diagnostica sui sistemi UNIX and Linux , è possibile impostare le seguenti proprietà:

```
traceEnabled:       true
traceLevel:         10
```

Queste proprietà non hanno effetto se l'adattatore di risorse non è stato avviato, ad esempio, quando le applicazioni che utilizzano le risorse WebSphere MQ sono in esecuzione solo nel contenitore client. In questa situazione, è possibile impostare le proprietà per la traccia diagnostica come proprietà di sistema

JVM (Java Virtual Machine). È possibile impostare le proprietà utilizzando l'indicatore -D sul comando **java** , come nel seguente esempio:

```
java ... -DtraceEnabled=true -DtraceLevel=6
```

Non è necessario definire tutte le proprietà dell'oggetto ResourceAdapter . Tutte le proprietà non specificate assumono i valori predefiniti. In un ambiente gestito, è preferibile non combinare i due modi di specificare le proprietà. Se si combinano, le proprietà di sistema JVM hanno la precedenza sulle proprietà dell'oggetto ResourceAdapter .

Configurazione dell'adattatore di risorse per la comunicazione in entrata
Per configurare la comunicazione in entrata, definire le proprietà di uno o più oggetti ActivationSpec .

Le proprietà di un oggetto ActivationSpec determinano il modo in cui un MDB (message drive bean) riceve i messaggi JMS da una coda WebSphere MQ . Il comportamento transazionale di MDB è definito nel relativo descrittore di distribuzione.

Un oggetto ActivationSpec ha due serie di proprietà:

- Proprietà utilizzate per creare una connessione JMS a un gestore code WebSphere MQ
- Proprietà utilizzate per creare un consumer di connessione JMS che distribuisce i messaggi in modo asincrono man mano che arrivano su una coda specificata

Il modo in cui si definiscono le propriet ... di un oggetto ActivationSpec dipende dalle interfacce di amministrazione fornite dal server delle applicazioni.

Tabella 96 a pagina 739 elenca le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS a un gestore code WebSphere MQ .

Tabella 96. Le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
applicationName	String a	• Il nome della classe di richiamo, se disponibile, è regolato in modo da non superare i 28 caratteri. Se non è disponibile, viene utilizzata la stringa `WebSphere MQ Client per Java`.	Il nome con cui un'applicazione è registrata con il gestore code. Questo nome applicazione viene visualizzato dal comando **DISPLAY CONN MQSC/PCF** (dove il campo è denominato **APPLTAG**) o nella visualizzazione IBM WebSphere MQ Esplora **Connessioni applicazione** (dove il campo è denominato **App name**).
brokerCCDurSubQueue[1]	String a	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Un nome coda	Il nome della coda da cui un utente della connessione riceve i messaggi di sottoscrizione durevoli
brokerCCSubcoda[1]	String a	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Un nome coda	Il nome della coda da cui un utente della connessione riceve messaggi di sottoscrizione non durevoli
brokerControlCoda[1]	String a	• **SYSTEM.BROKER.CONTROL.QUEUE** • Un nome coda	Il nome della coda di controllo del broker

Tabella 96. Le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
brokerQueueGestore[1]	Stringa	• **"" (stringa vuota)** • Un nome gestore code	Il nome del gestore code su cui è in esecuzione il broker
brokerSubQueue[1]	Stringa	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Un nome coda	Il nome della coda da cui un utente di messaggi non durevoli riceve i messaggi
brokerVersion[1]	Stringa	• **non specificato** - Dopo la migrazione del Broker da V6 a V7, impostare questa proprietà in modo che le intestazioni RFH2 non vengano più utilizzate. Dopo la migrazione, questa proprietà non è più rilevante. • **V1** - Per utilizzare un broker di pubblicazione / sottoscrizione WebSphere MQ . Oppure per utilizzare un broker di WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker in modalità di compatibilità. Questo valore è il valore predefinito se TRANSPORT è impostato su BIND o CLIENT. • **V2** - Per utilizzare un broker di WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker in modalità nativa. Questo valore è il valore predefinito se TRANSPORT è impostato su DIRECT o DIRECTHTTP.	La versione del broker utilizzato
ccdtURL	Stringa	• **vuoto** • Un URL (uniform resource locator)	Un URL che identifica il nome e l'ubicazione del file contenente la CCDT (client channel definition table) e specifica come è possibile accedere al file
CCSID	Stringa	• **819** • Un CCSID (coded character set identifier) supportato dalla JVM (Java virtual machine)	Il CCSID (coded character set identifier) per una connessione
canale	Stringa	• **SYSTEM.DEF.SVRCONN** • Il nome di un canale MQI	Il nome del canale MQi da utilizzare

Tabella 96. Le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
cleanupInterval[2]	int	• **3 600 000** • Un numero intero positivo	L'intervallo, in millisecondi, tra le esecuzioni in background del programma di utilità di ripulitura di pubblicazione / sottoscrizione
cleanupLevel[1]	String a	• **SICURA** • NESSUNA • forte • Forza • NONDUR	Il livello di ripulitura per un archivio di sottoscrizioni basato sul broker
clientID	String a	• **vuoto** • Un identificativo client	L'identificativo client per una connessione
cloneSupport	String a	• **DISABLED** - Solo un'istanza di un sottoscrittore di argomenti durevoli può essere eseguita alla volta. • ENABLED - Due o più istanze dello stesso sottoscrittore di argomenti durevoli possono essere eseguite simultaneamente, ma ciascuna istanza deve essere eseguita in una JVM (Java virtual machine) separata.	Se due o più istanze dello stesso sottoscrittore di argomenti durevoli possono essere eseguite contemporaneamente
ConnectionNameList	String a	• **host locale (1414)** • Una stringa composta da elementi separati da virgole in cui ogni elemento assume il formato: `HOSTNAME(PORT)` dove *HOSTNAME* è un nome DNS o un indirizzo IP.	Un elenco di nomi di connessione TCP/IP utilizzati per le comunicazioni in entrata. Quando specificato, **connectionNameList** sostituisce le proprietà **hostname** e **port** . Questa proprietà viene utilizzata per riconnettersi ai gestori code a più istanze. **connectionNameList** è simile nel formato a **localAddress**, ma non deve essere confuso con esso. **localAddress** specifica le caratteristiche delle comunicazioni locali, mentre **connectionNameList** specifica come raggiungere un gestore code remoto.

Tabella 96. Le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
FAILIFQUIESCE	boolean	• **true** • No	Indica se le chiamate a determinati metodi non riescono se il gestore code è in uno stato di sospensione
headerCompression	String	• **Nessuno** • Viene eseguita la compressione dell'intestazione del messaggio SYSTEM - RLE	Un elenco delle tecniche che può essere utilizzato per comprimere i dati di intestazione su una connessione
hostName	String	• **host locale** • Un nome host • Un indirizzo IP	Il nome host o l'indirizzo IP del sistema su cui si trova il gestore code. Le proprietà **hostname** e **port** vengono sostituite dalla proprietà **connectionNameList** quando viene specificata.
localAddress	String	• **vuoto** • Una stringa nel formato: `[host_name][(low_port[,high_port])]` dove *host_name* è un nome host o un indirizzo IP, *low_port* e *high_port* sono numeri di porta TCP e le parentesi indicano un componente facoltativo	Per una connessione a un gestore code, questa proprietà specifica uno o entrambi i seguenti elementi: • L'interfaccia di rete locale da utilizzare • La porta locale o l'intervallo di porte locali da utilizzare **localAddress** è simile nel formato a **connectionNameList**, ma non deve essere confuso con esso. **localAddress** specifica le caratteristiche delle comunicazioni locali, mentre **connectionNameList** specifica come raggiungere un gestore code remoto.
messageCompression	String	• **Nessuno** • Un elenco di uno o più dei seguenti valori separati da caratteri vuoti: RLE ZLIBFAST ZLIBHIGH	Un elenco delle tecniche che è possibile utilizzare per comprimere i dati del messaggio su una connessione

Tabella 96. Le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
messageRetention[1]	booleano	• **true** - I messaggi indesiderati rimangono nella coda di input • false - I messaggi indesiderati vengono gestiti in base alle opzioni di disposizione	Se il consumer della connessione conserva i messaggi indesiderati nella coda di input
messageSelection[1]	Stringa	• **client** • BROKER	Determina se la selezione del messaggio viene eseguita dalle classi WebSphere MQ per JMS o dal broker. La selezione dei messaggi da parte del broker non è supportata quando brokerVersion ha il valore 1.
password	Stringa	• **vuoto** • Una password	La password predefinita da utilizzare quando si crea una connessione al gestore code
pollingInterval[1]	int	• **5000** • Qualsiasi numero intero positivo	Se ciascun listener messaggi in una sessione non dispone di alcun messaggio adatto nella propria coda, questo valore è l'intervallo massimo, in millisecondi, che trascorre prima che ciascun listener messaggi provi di nuovo ad ottenere un messaggio dalla propria coda. Se si verifica spesso che non è disponibile alcun messaggio adatto per nessuno dei listener di messaggi in una sessione, aumentare il valore di questa proprietà. Questa proprietà è rilevante solo se TRANSPORT ha il valore BIND o CLIENT.
porta	int	• **1414** • Un numero di porta TCP	La porta su cui è in ascolto il gestore code. Le proprietà **hostname** e **port** vengono sostituite dalla proprietà **connectionNameList** quando viene specificata.

Tabella 96. Le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
providerVersion	string a	• **non specificato** • Una stringa in uno dei seguenti formati – V.R.M.F – V.R.M – V.R – V dove V, R, M e F sono valori interi maggiori o uguali a zero.	La versione, la release, il livello di modifica e il fix pack del gestore code a cui MDB intende connettersi.
queueManager	String a	• **"" (stringa vuota)** • Un nome gestore code	Il nome del gestore code a cui connettersi
receiveExit [3]	String a	• **vuoto** • Una stringa che comprende uno o più elementi separati da virgole, dove ogni elemento è il nome completo di una classe che implementa l'interfaccia WebSphere MQ classes per Java, MQReceiveExit	Identifica un programma di uscita di ricezione del canale o una sequenza di programmi di uscita di ricezione da eseguire in successione
Inizializzazione di receiveExit	String a	• **vuoto** • Una stringa che comprende uno o più elementi di dati utente separati da virgole	I dati utente passati ai programmi di uscita di ricezione del canale quando vengono richiamati

Tabella 96. Le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
rescanInterval[1]	int	• **5000** • Qualsiasi numero intero positivo	Quando un utente di messaggi nel dominio point - to - point utilizza un selettore di messaggi per selezionare i messaggi che desidera ricevere, WebSphere MQ classes per JMS ricerca nella coda WebSphere MQ i messaggi adatti nella sequenza determinata dall'attributo *MsgDeliverySequence* della coda. Quando WebSphere MQ classes per JMS trova un messaggio adatto e lo consegna all'utente, WebSphere MQ classes for JMS riprende la ricerca del successivo messaggio idoneo dalla posizione corrente nella coda. WebSphere MQ classes for JMS continua a ricercare la coda in questo modo fino a quando raggiunge la fine della coda o fino a quando l'intervallo di tempo in millisecondi, come determinato dal valore di questa proprietà, è scaduto. In ogni caso, WebSphere MQ classes per JMS ritorna all'inizio della coda per continuare la ricerca e inizia un nuovo intervallo di tempo.
securityExit [3]	Stringa	• **vuoto** • Il nome completo di una classe che implementa le classi WebSphere MQ per l'interfaccia Java, MQSecurityExit	Identifica un programma di uscita di sicurezza del canale
Inizializzazione di securityExit	Stringa	• **vuoto** • Una stringa di dati utente	I dati utente che vengono passati a un programma di uscita di sicurezza del canale quando viene richiamato

Tabella 96. Le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
sendExit [3]	String a	• **vuoto** • Una stringa che comprende uno o più elementi separati da virgole, dove ogni elemento è il nome completo di una classe che implementa le classi WebSphere MQ per l'interfaccia Java, MQSendExit	Identifica un programma di uscita di invio del canale o una sequenza di programmi di uscita di invio da eseguire in successione
SENDEXITINIT	String a	• **vuoto** • Una stringa che comprende uno o più elementi di dati utente separati da virgole	I dati utente passati ai programmi di uscita di invio del canale quando vengono richiamati
SHARECONVALLOWED	boolean o	• NO - Una connessione client non può condividere il proprio socket. • **YES** - Una connessione client può condividere il proprio socket.	Indica se una connessione client può condividere il proprio socket con altre connessioni JMS di livello superiore dallo stesso processo allo stesso gestore code, se le definizioni del canale corrispondono
sparseSubscriptions[1]	boolean o	• **false** - Le sottoscrizioni ricevono messaggi corrispondenti frequenti. • true - Le sottoscrizioni ricevono messaggi di corrispondenza non frequenti. Questo valore richiede che la coda di sottoscrizione possa essere aperta per la ricerca.	Controlla la politica di recupero messaggi di un oggetto TopicSubscriber
Archivi sslCert	String a	• **vuoto** • Una stringa di uno o più URL LDAP separati da spazi. Ogni URL LDAP ha il formato: `ldap://host_name[:port]` dove *host_name* è un nome host o un indirizzo IP, *port* è un numero di porta TCP e le parentesi indicano un componente facoltativo.	I server LDAP (Lightweight Directory Access Protocol) che contengono i CRL (Certificate Revocation List) da utilizzare su una connessione SSL
SSLCipherSuite	String a	• **vuoto** • Il nome di una CipherSuite	La CipherSuite da utilizzare per una connessione SSL
sslFipsRichiesto [2]	boolean o	• **No** • vero, true	Se una connessione SSL deve utilizzare una CipherSuite supportata dal provider IBM Java JSSE FIPS (IBMJSSEFIPS)

Tabella 96. Le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
SSLPeerName	String a	• **vuoto** • Un modello per i DN (distinguished name)	Per una connessione SSL, un modello utilizzato per controllare il DN (Distinguished Name) nel certificato digitale fornito dal gestore code
SSLResetCount	int	• **0** • Un numero intero compreso tra 0 e 999 999 999	Il numero totale di byte inviati e ricevuti da una connessione SSL prima che le chiavi segrete utilizzate da SSL vengano rinegoziati
Factory sslSocket	String a	Una stringa che rappresenta il nome classe completo di una classe che fornisce un'implementazione dell'interfaccia javax.net.ssl.SSLSocketFactory . Facoltativamente, includere un argomento da passare al metodo constructor, racchiuso tra parentesi.	Tutte le connessioni stabilite nell'ambito dell'oggetto gestito utilizzano i socket ottenuti da questa implementazione dell'interfaccia SSLSocketFactory .
statusRefreshInterval[1]	int	• **60000** • Qualsiasi numero intero positivo	L'intervallo, in millesimi di secondo, tra gli aggiornamenti della transazione di lunga durata che rileva quando un sottoscrittore perde la propria connessione al gestore code. Questa proprietà è rilevante solo se subscriptionStore ha il valore QUEUE.
subscriptionStore[1]	String a	• **BROKER** • MIGRAZIONE • CODA	Determina dove WebSphere MQ classes per JMS memorizza i dati persistenti relativi alle sottoscrizioni attive
transportType	String a	• **client** • BIND • BINDING_THEN_CLIENT	Indica se una connessione a un gestore code utilizza la modalità client o la modalità bind. Se viene specificato il valore BINDINGS_THEN_CLIENT, l'adattatore di risorse tenta prima di stabilire una connessione in modalità bind. Se questo tentativo di connessione non riesce, l'adattatore di risorse tenta di stabilire una connessione in modalità client.

Tabella 96. Le proprietà di un oggetto ActivationSpec utilizzate per creare una connessione JMS (Continua)			
Nome della proprietà	**Tipo**	**Valori validi (valore predefinito in grassetto)**	**Descrizione**
nome utente	String a	• **vuoto** • Un nome utente	Il nome dell'utente predefinito da utilizzare durante la creazione di una connessione a un gestore code
wildcardFormat	String a	• CHAR - Riconosce solo i caratteri jolly, come utilizzato nel broker versione 1 • **TOPIC** - Riconosce solo i caratteri jolly a livello di argomento, come utilizzato nel broker versione 2	La versione della sintassi dei caratteri jolly da utilizzare

Note:

1. Questa proprietà può essere utilizzata con la versione 7.0 di WebSphere MQ classes per JMS. Non influisce su un'applicazione connessa a un gestore code versione 7.0 a meno che la proprietà providerVersion non sia impostata su un numero di versione inferiore a 7.
2. Per informazioni importanti sull'utilizzo della proprietà sslFipsRequired, consultare "Limitazioni dell'adattatore di risorse IBM WebSphere MQ" a pagina 772.
3. Per informazioni su come configurare l'adattatore risorse in modo che possa individuare un'uscita, consultare "Configurazione di IBM WebSphere MQ classes for JMS per l'utilizzo delle uscite canale" a pagina 914.

Tabella 97 a pagina 748 elenca le proprietà di un oggetto ActivationSpec utilizzato per creare un consumer di connessione JMS.

Tabella 97. Proprietà di un oggetto ActivationSpec utilizzate per creare un consumer di connessione JMS			
Nome della proprietà	**Tipo**	**Valori validi (valore predefinito in grassetto)**	**Descrizione**
destinazione	String a	Un nome destinazione	La destinazione da cui ricevere i messaggi. La proprietà useJNDI determina come viene interpretato il valore di questa proprietà.
destinationType	String a	• javax.jms.Queue • javax.jms.Topic	Il tipo di destinazione, di coda o di argomento
maxMessages	int	• **1** • Un numero intero positivo	Il numero massimo di messaggi che è possibile assegnare ad una sessione server contemporaneamente. Se la specifica di attivazione sta consegnando messaggi a un MDB in una transazione XA, viene utilizzato il valore 1 indipendentemente dall'impostazione di questa proprietà.
Profondità maxPool	int	• **10** • Un numero intero positivo	Il numero massimo di sessioni server nel pool di sessioni server utilizzato dal consumer della connessione

Tabella 97. Proprietà di un oggetto ActivationSpec utilizzate per creare un consumer di connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
messageSelector	String a	• **vuoto** • Un'espressione del selettore messaggi SQL92	Un'espressione del selettore di messaggi che specifica quali messaggi devono essere consegnati
nonASFTimeout	int	• **0** • Un numero intero positivo	Un valore positivo indica che viene utilizzata la distribuzione non ASF. Il valore è il tempo, in millesimi di secondo, che una richiesta get attende per i messaggi che potrebbero non essere ancora arrivati (una chiamata get with wait). Il valore predefinito, 0, indica che viene utilizzata la consegna ASF. Questo parametro è valido solo quando l'applicazione è in esecuzione su WebSphere Application Server versione 7 o successiva.
nonASFRollbackabilitato	boolean o	• **false** - Il messaggio viene utilizzato anche se l'MDB ha esito negativo • true - L'errore nell'MDB causa il rollback del messaggio nella coda.	Se la consegna del messaggio si trova all'interno di un punto di sincronizzazione WebSphere MQ se l'MDB non è transattato. Ignorato se l'MDB viene transattato o se nonASFTimeout è impostato su 0.
poolTimeout	int	• **300000** • Un numero intero positivo	Il tempo, in millisecondi, in cui una sessione server inutilizzata viene tenuta aperta nel pool di sessioni server prima di essere chiusa a causa di inattività
READAHEADALLOWED	int	• **DESTINATION** - Determina se la lettura anticipata è consentita facendo riferimento alla definizione della coda o dell'argomento. • DISABLED - La lettura anticipata non è consentita. • ENABLED - La lettura anticipata è consentita. • QUEUE - Determinare se la lettura anticipata è consentita facendo riferimento alla definizione della coda. • TOPIC - Determinare se la lettura anticipata è consentita facendo riferimento alla definizione argomento.	Se a MDB è consentito utilizzare la lettura anticipata per ottenere messaggi non persistenti dalla destinazione in un buffer interno prima di riceverli

Tabella 97. Proprietà di un oggetto ActivationSpec utilizzate per creare un consumer di connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
readAheadClosePolicy	int	• **ALL** - Tutti i messaggi nel buffer di lettura anticipata interno vengono consegnati all'MDB prima dell'arresto. • CURRENT - Viene completato solo il richiamo MDB corrente, lasciando potenzialmente i messaggi nel buffer di lettura anticipata interno, che vengono quindi eliminati.	Cosa accade ai messaggi nel buffer di lettura anticipata interno quando MDB viene arrestato dall'amministratore.
receiveCCSID	int	• **0** - Utilizza JVM `Charset.defaultCharset` • 1208 - UTF-8 • Un CCSID (coded character set identifier) supportato	Proprietà di destinazione che imposta il CCSID di destinazione per la conversione del messaggio del gestore code. Il valore viene ignorato a meno che **`receiveConversion`** non sia impostato su `QMGR`
receiveConversion	String a	• **MSG CLI** • QMGR	Proprietà di destinazione che determina se la conversione dati verrà eseguita dal gestore code.
startTimeout	int	• **10 000** • Un numero intero positivo	Il tempo, in millisecondi, entro il quale deve iniziare la consegna di un messaggio a un MDB dopo che il lavoro per consegnare il messaggio è stato pianificato. Se questo tempo trascorre, viene eseguito il rollback del messaggio sulla coda.
subscriptionDurability	String a	• **NonDurable** - Una sottoscrizione non durevole viene utilizzata per consegnare i messaggi a un MDB che effettua la sottoscrizione all'argomento. • Durevole - Una sottoscrizione durevole viene utilizzata per consegnare i messaggi a un MDB che effettua la sottoscrizione all'argomento.	Se una sottoscrizione durevole o non durevole viene utilizzata per consegnare i messaggi a un MDB che effettua la sottoscrizione all'argomento
subscriptionName	String a	• **"" (stringa vuota)** • Un nome sottoscrizione	Il nome della sottoscrizione durevole

Tabella 97. Proprietà di un oggetto ActivationSpec utilizzate per creare un consumer di connessione JMS (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
useJNDI	boolean	• **false** - La proprietà denominata destinazione viene interpretata come il nome di una coda WebSphere MQ o di un argomento. • true - La proprietà denominata destinazione viene interpretata come il nome di un oggetto javax.jms.Queue o di un oggetto javax.jms.Topic nello spazio nomi JNDI del server delle applicazioni.	Determina il modo in cui viene interpretato il valore della proprietà denominata destinazione

Le proprietà ActivationSpec richiamate destination e destinationType devono essere definite esplicitamente. Tutte le altre proprietà sono facoltative.

Un oggetto ActivationSpec può avere proprietà in conflitto. Ad esempio, è possibile specificare le proprietà SSL per una connessione in modalità bind. In questo caso, il comportamento è determinato dal tipo di trasporto e dal dominio di messaggistica, che è point-to-point o pubblicazione / sottoscrizione come determinato dalla proprietà destinationType . Tutte le proprietà non applicabili al tipo di trasporto specificato o al dominio di messaggistica vengono ignorate.

Se si definisce una proprietà che richiede la definizione di altre proprietà, ma non si definiscono queste altre proprietà, l'oggetto ActivationSpec genera un'eccezione InvalidPropertyquando il relativo metodo validate () viene richiamato durante la distribuzione di un MDB. L'eccezione viene notificata all'amministratore del server delle applicazioni in un modo che dipende dal server delle applicazioni. Ad esempio, se si imposta la proprietà subscriptionDurability su Durable, che indica che si desidera utilizzare le sottoscrizioni durevoli, è necessario definire anche la proprietà subscriptionName .

Se le proprietà denominate ccdtURL e canale sono entrambe definite, viene generata un'eccezione InvalidProperty. Tuttavia, se si definisce solo la proprietà ccdtURL , lasciando la proprietà denominata channel con il valore predefinito SYSTEM.DEF.SVRCONN, non viene generata alcuna eccezione e la tabella di definizione del canale client identificata dalla proprietà ccdtURL viene utilizzata per avviare una connessione JMS.

La maggior parte delle proprietà di un oggetto ActivationSpec sono equivalenti alle proprietà di WebSphere MQ classes for JMS objects o parameters di WebSphere MQ classes for JMS methods. Tuttavia, tre proprietà di ottimizzazione e una proprietà di utilizzabilità non hanno equivalenti nelle classi WebSphere MQ per JMS:

startTimeout
Il tempo, in millesimi di secondo, durante il quale il gestore lavoro del server delle applicazioni attende che le risorse diventino disponibili dopo che l'adattatore risorse pianifica un oggetto Work per consegnare un messaggio a un MDB. Se questo tempo trascorre prima dell'avvio della consegna del messaggio, l'oggetto Work scade, viene eseguito il rollback del messaggio sulla coda e l'adattatore di risorse può quindi tentare nuovamente di consegnare il messaggio. Un'avvertenza viene scritta nella traccia diagnostica, se abilitata, ma non influisce sul processo di consegna dei messaggi. Si potrebbe prevedere che questa condizione si verifichi solo quando il server delle applicazioni sta riscontrando un carico molto elevato. Se la condizione si verifica regolarmente, aumentare il valore di questa proprietà per consentire al gestore lavoro di pianificare la consegna dei messaggi più a lungo.

Profondità maxPool
Il numero massimo di sessioni server nel pool di sessioni server utilizzato da un utente di connessione. Quando viene creata una sessione server, viene avviata una conversazione con un gestore code. Il consumer della connessione utilizza una sessione server per consegnare un messaggio a un MDB. Una maggiore profondità del pool consente la consegna simultanea di più messaggi in situazioni di volumi elevati, ma utilizza più risorse del server delle applicazioni. Se devono essere distribuiti molti MDB, considerare la possibilità di ridurre la profondità del pool per mantenere il carico sul server delle applicazioni ad un livello gestibile. Ogni utente di connessione utilizza il proprio lotto sessioni server, in modo che questa proprietà non definisca il numero totale di sessioni server disponibili per tutti gli utenti di connessione.

poolTimeout
Il tempo, in millisecondi, durante il quale una sessione server inutilizzata viene tenuta aperta nel pool di sessioni server prima di essere chiusa a causa di inattività. Un aumento transitorio del carico di lavoro del messaggio provoca la creazione di sessioni server aggiuntive per distribuire il carico ma, una volta che il carico di lavoro del messaggio è tornato normale, le sessioni server aggiuntive rimangono nel pool e non vengono utilizzate.

Ogni volta che viene utilizzata una sessione server, viene contrassegnata con una data / ora. Periodicamente un thread di scavenger verifica che ogni sessione server sia stata utilizzata entro il periodo specificato da questa proprietà. Se una sessione del server non è stata utilizzata, viene chiusa e rimossa dal pool di sessioni del server. Una sessione server potrebbe non essere chiusa immediatamente dopo la scadenza del periodo specificato, questa proprietà rappresenta il periodo minimo di inattività prima della rimozione.

useJNDI
Per una descrizione di questa proprietà, consultare Tabella 97 a pagina 748.

Per distribuire un MDB, definire prima le proprietà di un oggetto ActivationSpec , specificando le proprietà richieste da MDB. Il seguente esempio è una serie tipica di proprietà che è possibile definire esplicitamente:

```
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
```

Il server delle applicazioni utilizza le proprietà per creare un oggetto ActivationSpec , che viene quindi associato a MDB. Le proprietà dell'oggetto ActivationSpec determinano in che modo i messaggi vengono consegnati all'MDB. La distribuzione dell'MDB ha esito negativo se l'MDB richiede transazioni distribuite ma l'adattatore di risorse non supporta le transazioni distribuite. Per informazioni su come installare l'adattatore risorse in modo che le transazioni distribuite siano supportate, consultare “Installazione dell'adattatore di risorse WebSphere MQ” a pagina 735.

Se più di un MDB sta ricevendo messaggi dalla stessa destinazione, un messaggio inviato nel dominio point-to-point viene ricevuto da un solo MDB, anche se altri MDB sono idonei a ricevere il messaggio. In particolare, se due MDB stanno utilizzando diversi selettori di messaggi e un messaggio in entrata corrisponde a entrambi i selettori di messaggi, solo uno degli MDB riceve il messaggio. L'MDB scelto per ricevere un messaggio non è definito e non è possibile basarsi su un MDB specifico che riceve il messaggio. I messaggi inviati nel dominio di pubblicazione / sottoscrizione vengono ricevuti da tutti gli MDB idonei.

Gestione dei messaggi non elaborabili in ingresso nell'adattatore di risorse

In alcune circostanze, è possibile che venga eseguito il rollback di un messaggio consegnato a un MDB su una coda WebSphere MQ . Questo rollback può verificarsi, ad esempio, se un messaggio viene consegnato all'interno di un'unità di lavoro di cui viene eseguito il rollback. Un messaggio di cui è stato eseguito il rollback viene recapitato di nuovo, ma un messaggio formattato in modo non corretto potrebbe causare ripetutamente un errore MDB e quindi non può essere consegnato. Tale messaggio è chiamato un messaggio di veleno. È possibile configurare WebSphere MQ in modo che le classi WebSphere MQ per

JMS trasferisca automaticamente un messaggio non elaborabile in un'altra coda per un'ulteriore analisi o eliminino il messaggio.

Per i dettagli su come gestire i messaggi dannosi, consultare “Gestione dei messaggi non elaborabili in IBM WebSphere MQ classes for JMS” a pagina 891.

Attività correlate

Specifica che solo i CipherSpecs certificati FIPS vengono utilizzati al runtime sul client MQI

Riferimenti correlati

Federal Information Processing Standards (FIPS) per UNIX, Linux e Windows

Configurazione dell'adattatore di risorse per la comunicazione in uscita

Per configurare la comunicazione in uscita, definire le proprietà di un oggetto ConnectionFactory e di un oggetto di destinazione gestito.

Quando si utilizza la comunicazione in uscita, un'applicazione in esecuzione in un server delle applicazioni avvia una connessione a un gestore code, quindi invia i messaggi alle code e riceve i messaggi dalle code in modo sincrono. Ad esempio, il seguente metodo servlet, doGet(), utilizza la comunicazione in uscita:

```
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();
}
```

Quando il servlet riceve una richiesta HTTP GET, richiama un oggetto ConnectionFactory e un oggetto Queue dallo spazio nomi JNDI e utilizza gli oggetti per inviare un messaggio a una coda WebSphere MQ . Il servlet riceve quindi il messaggio che ha inviato.

Per configurare la comunicazione in uscita, definire le risorse JCA nelle seguenti categorie:

- Le proprietà di un oggetto ConnectionFactory che il server delle applicazioni utilizza per creare un oggetto ConnectionFactory JMS.
- Le proprietà di un oggetto di destinazione gestito, che il server delle applicazioni utilizza per creare un oggetto coda JMS o un oggetto argomento JMS.

Il modo in cui si definiscono queste proprietà dipende dalle interfacce di gestione fornite dal server delle applicazioni. Gli oggetti ConnectionFactory, Queue e Topic creati dal server delle applicazioni sono collegati in uno spazio dei nomi JNDI da cui possono essere recuperati da un'applicazione.

Generalmente, si definisce un oggetto ConnectionFactory per ogni gestore code a cui le applicazioni potrebbero aver bisogno di connettersi. Si definisce un oggetto Coda per ogni coda a cui le applicazioni potrebbero dover accedere nel dominio point - to - point. E si definisce un oggetto Argomento per ogni argomento che le applicazioni potrebbero voler pubblicare o sottoscrivere. Un oggetto ConnectionFactory può essere indipendente dal dominio. In alternativa, può essere specifico del dominio, un oggetto factory QueueConnectionper il dominio point-to-point o un oggetto factory TopicConnectionper il dominio di pubblicazione / sottoscrizione.

Tabella 98 a pagina 754 elenca le proprietà di un oggetto ConnectionFactory .

Tabella 98. Proprietà di un oggetto ConnectionFactory

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
applicationName	String a	• Il nome della classe di richiamo, se disponibile, è regolato in modo da non superare i 28 caratteri. Se non è disponibile, viene utilizzata la stringa `WebSphere MQ Client per Java`.	Il nome con cui un'applicazione è registrata con il gestore code. Questo nome applicazione viene visualizzato dal comando **`DISPLAY CONN MQSC/PCF`** (dove il campo è denominato **`APPLTAG`**) o nella visualizzazione IBM WebSphere MQ Esplora **Connessioni applicazione** (dove il campo è denominato **`App name`**).
brokerCCSubcoda[1]	String a	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Un nome coda	Il nome della coda da cui un utente della connessione riceve messaggi di sottoscrizione non durevoli.
brokerControlCoda[1]	String a	• **SYSTEM.BROKER.CONTROL.QUEUE** • Un nome coda	Il nome della coda di controllo broker.
brokerPubQueue[1]	String a	• **SYSTEM.BROKER.DEFAULT.STREAM** • Un nome coda	Il nome della coda in cui vengono inviati i messaggi pubblicati (la coda di flusso).
brokerQueueGestore[1]	String a	• **"" (stringa vuota)** • Un nome gestore code	Il nome del gestore code su cui è in esecuzione il broker.
brokerSubQueue[1]	String a	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Un nome coda	Il nome della coda da cui un utente di messaggi non durevoli riceve i messaggi. Per ulteriori informazioni, consultare la proprietà BROKERSUBQ .

Tabella 98. Proprietà di un oggetto ConnectionFactory (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
brokerVersion[1]	String a	• **non specificato** - Dopo che il broker è stato migrato da V6 a V7, impostare questa proprietà in modo che le intestazioni RFH2 non vengano più utilizzate. Dopo la migrazione, questa proprietà non è più rilevante. • **V1** - Per utilizzare un broker di pubblicazione / sottoscrizione IBM WebSphere MQ . oppure utilizzare un broker di IBM WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker in modalità di compatibilità. Questo valore è il valore predefinito se TRANSPORT è impostato su BIND o CLIENT. • **V2** - Per utilizzare un broker di IBM WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker in modalità nativa. Questo valore è il valore predefinito se TRANSPORT è impostato su DIRECT o DIRECTHTTP.	La versione del broker utilizzato.
ccdtURL	String a	• **vuoto** • Un URL (uniform resource locator)	Un URL che identifica il nome e l'ubicazione del file contenente la CCDT (client channel definition table) e specifica il modo in cui è possibile accedere al file.
CCSID	String a	• **819** • Un CCSID (coded character set identifier) supportato dalla JVM (Java virtual machine)	Il CCSID (coded character set identifier) per una connessione.
canale	String a	• **SYSTEM.DEF.SVRCONN** • Il nome di un canale MQI	Il nome del canale MQI da utilizzare.
cleanupInterval[2]	int	• **3 600 000** • Un numero intero positivo	L'intervallo, in millisecondi, tra le esecuzioni in background del programma di utilità di ripulitura di pubblicazione / sottoscrizione.
cleanupLevel[1]	String a	• **SICURA** • NESSUNA • forte • Forza • NONDUR	Il livello di ripulitura per un archivio sottoscrizioni basato su broker.

Tabella 98. Proprietà di un oggetto ConnectionFactory (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
clientID	Stringa	• **vuoto** • Un identificativo client	L'identificativo client per una connessione.
cloneSupport	Stringa	• **DISABLED** - Solo un'istanza di un sottoscrittore di argomenti durevoli può essere eseguita alla volta. • ENABLED - Due o più istanze dello stesso sottoscrittore di argomenti durevoli possono essere eseguite contemporaneamente, ma ciascuna istanza deve essere eseguita in una JVM (Java virtual machine) separata.	Se due o più istanze dello stesso sottoscrittore di argomenti durevoli possono essere eseguite contemporaneamente.
ConnectionNameList	Stringa	• **host locale (1414)** • Una stringa composta da elementi separati da virgole in cui ogni elemento assume il formato: `HOSTNAME(PORT)` dove *HOSTNAME* è un nome DNS o un indirizzo IP.	Un elenco di nomi di connessione TCP/IP utilizzati per le comunicazioni in uscita. **connectionNameList** sostituisce le proprietà **hostname** e **port** . Questa proprietà viene utilizzata per riconnettersi ai gestori code a più istanze. **connectionNameList** è simile nel formato a **localAddress**, ma non deve essere confuso con esso. **localAddress** specifica le caratteristiche delle comunicazioni locali, mentre **connectionNameList** specifica come raggiungere un gestore code remoto.
FAILIFQUIESCE	booleano	• **true** • No	Indica se le chiamate a determinati metodi hanno esito negativo se il gestore code è in uno stato di inattività.
headerCompression	Stringa	• **Nessuno** • Viene eseguita la compressione dell'intestazione del messaggio SYSTEM - RLE.	Un elenco delle tecniche che è possibile utilizzare per comprimere i dati di intestazione su una connessione.
hostName	Stringa	• **host locale** • Un nome host • Un indirizzo IP	Il nome host o l'indirizzo IP del sistema su cui si trova il gestore code. Le proprietà **hostname** e **port** vengono sostituite dalla proprietà **connectionNameList** quando viene specificata.

Tabella 98. Proprietà di un oggetto ConnectionFactory (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione

localAddress	String a	• **vuoto**
		• Una stringa nel formato:
		`[host_name]`
		`[(low_port[,high_port])]`
		dove *host_name* è un nome host o un indirizzo IP, *low_port* e *high_port* sono numeri di porta TCP e le parentesi indicano un componente facoltativo
			Per una connessione a un gestore code, questa proprietà specifica uno o entrambi i seguenti valori:
			• L'interfaccia di rete locale da utilizzare
			• La porta locale o l'intervallo di porte locali da utilizzare
			localAddress è simile nel formato a **connectionNameList**, ma non deve essere confuso con esso. **localAddress** specifica le caratteristiche delle comunicazioni locali, mentre **connectionNameList** specifica come raggiungere un gestore code remoto.

messageCompression	String a	• **Nessuno** • Un elenco di uno o più dei seguenti valori separati da caratteri vuoti: RLE ZLIBFAST ZLIBHIGH	Un elenco delle tecniche che è possibile utilizzare per comprimere i dati del messaggio su una connessione.
messageSelection[1]	String a	• **client** • BROKER	Determina se la selezione del messaggio viene eseguita dalle classi IBM WebSphere MQ per JMS o dal broker. La selezione dei messaggi da parte del broker non è supportata quando brokerVersion ha il valore 1.
password	String a	• **vuoto** • Una password	La password predefinita da utilizzare quando si crea una connessione al gestore code.

Tabella 98. Proprietà di un oggetto ConnectionFactory (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
pollingInterval[1]	int	• **5000** • Qualsiasi numero intero positivo	Se ciascun listener messaggi in una sessione non dispone di alcun messaggio adatto nella propria coda, questo valore è l'intervallo massimo, in millisecondi, che trascorre prima che ciascun listener messaggi provi di nuovo ad ottenere un messaggio dalla propria coda. Se si verifica spesso che non è disponibile alcun messaggio adatto per nessuno dei listener di messaggi in una sessione, aumentare il valore di questa proprietà. Questa proprietà è rilevante solo se TRANSPORT ha il valore BIND o CLIENT.
porta	int	• **1414** • Un numero di porta TCP	La porta su cui è in ascolto il gestore code. Le proprietà **hostname** e **port** vengono sostituite dalla proprietà **connectionNameList** quando viene specificata.
providerVersion	stringa	• **non specificato** • Una stringa in uno dei seguenti formati – V.R.M.F – V.R.M – V.R – V dove V, R, M e F sono valori interi maggiori o uguali a zero.	La versione, la release, il livello di modifica e il fix pack del gestore code al quale l'applicazione intende connettersi.
pubAckInterval[1]	int	• **25** • Un numero intero positivo	Il numero di messaggi pubblicati dal publisher prima che IBM WebSphere MQ classes for JMS richieda un riconoscimento dal broker.
queueManager	Stringa	• **"" (stringa vuota)** • Un nome gestore code	Il nome del gestore code a cui connettersi.
receiveExit [3]	Stringa	• **vuoto** • Una stringa che comprende uno o più elementi separati da virgole, dove ogni elemento è il nome completo di una classe che implementa l'interfaccia IBM WebSphere MQ classes for Java , MQReceiveExit	Identifica un programma di uscita di ricezione del canale o una sequenza di programmi di uscita di ricezione da eseguire in successione.

Tabella 98. Proprietà di un oggetto ConnectionFactory (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
Inizializzazione di receiveExit	String a	• **vuoto** • Una stringa che comprende uno o più elementi di dati utente separati da virgole	I dati utente passati ai programmi di uscita di ricezione del canale quando vengono richiamati.
rescanInterval[1]	int	• **5000** • Qualsiasi numero intero positivo	Quando un utente di messaggi nel dominio point - to - point utilizza un selettore di messaggi per selezionare quali messaggi desidera ricevere, WebSphere MQ classes per JMS ricerca nella coda IBM WebSphere MQ i messaggi idonei nella sequenza determinata dall'attributo *MsgDeliverySequence* della coda. Quando WebSphere MQ classes per JMS trova un messaggio adatto e lo consegna all'utente, WebSphere MQ classes for JMS riprende la ricerca del successivo messaggio idoneo dalla posizione corrente nella coda. WebSphere MQ classes for JMS continua a ricercare la coda in questo modo fino a quando raggiunge la fine della coda o fino a quando l'intervallo di tempo in millisecondi, come determinato dal valore di questa proprietà, è scaduto. In ogni caso, WebSphere MQ classes per JMS ritorna all'inizio della coda per continuare la ricerca e inizia un nuovo intervallo di tempo.
securityExit [3]	String a	• **vuoto** • Il nome completo di una classe che implementa le classi WebSphere MQ per l'interfaccia Java, MQSecurityExit	Identifica un programma di uscita di sicurezza del canale.
Inizializzazione di securityExit	String a	• **vuoto** • Una stringa di dati utente	I dati utente che vengono passati a un programma di uscita di sicurezza del canale quando viene richiamato.
SENDCHECKCOUNT	int	• **0** • Qualsiasi numero intero positivo	Il numero di chiamate di invio da consentire tra il controllo degli errori di inserimento asincrono, all'interno di una singola sessione JMS non transattiva.

Tabella 98. Proprietà di un oggetto ConnectionFactory (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
sendExit [3]	String a	• **vuoto** • Una stringa che comprende uno o più elementi separati da virgole, dove ogni elemento è il nome completo di una classe che implementa le classi WebSphere MQ per l'interfaccia Java, MQSendExit	Identifica un programma di uscita di invio del canale o una sequenza di programmi di uscita di invio da eseguire in successione.
SENDEXITINIT	String a	• **vuoto** • Una stringa che comprende uno o più elementi di dati utente separati da virgole	I dati utente trasmessi ai programmi di uscita di invio del canale quando vengono richiamati.
SHARECONVALLOWED	boolean o	• NO - Una connessione client non può condividere il proprio socket. • **YES** - Una connessione client può condividere il proprio socket.	Indica se una connessione client può condividere il proprio socket con altre connessioni JMS di livello superiore dallo stesso processo allo stesso gestore code, se le definizioni del canale corrispondono.
sparseSubscriptions[1]	boolean o	• **false** - Le sottoscrizioni ricevono messaggi corrispondenti frequenti. • true - Le sottoscrizioni ricevono messaggi di corrispondenza non frequenti. Questo valore richiede che la coda di sottoscrizione possa essere aperta per la ricerca.	Controlla la politica di richiamo messaggi di un oggetto TopicSubscriber .
Archivi sslCert	String a	• **vuoto** • Una stringa di uno o più URL LDAP separati da spazi. Ogni URL LDAP ha il formato: `ldap://host_name[:port]` dove *host_name* è un nome host o un indirizzo IP, *port* è un numero di porta TCP e le parentesi indicano un componente facoltativo.	I server LDAP (Lightweight Directory Access Protocol) che contengono CRL (Certificate Revocation List) da utilizzare su una connessione SSL.
SSLCipherSuite	String a	• **vuoto** • Il nome di una CipherSuite	La CipherSuite da utilizzare per una connessione SSL.
sslFipsRichiesto [2]	boolean o	• **No** • vero, true	Se una connessione SSL deve utilizzare una CipherSuite supportata dal provider IBM Java JSSE FIPS (IBMJSSEFIPS).

Tabella 98. Proprietà di un oggetto ConnectionFactory (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
SSLPeerName	String a	• **vuoto** • Un modello per i DN (distinguished name)	Per una connessione SSL, un modello utilizzato per controllare il DN (distinguished name) nel certificato digitale fornito dal gestore code.
SSLResetCount	int	• **0** • Un numero intero compreso tra 0 e 999 999 999	Il numero totale di byte inviati e ricevuti da una connessione SSL prima che le chiavi segrete utilizzate da SSL vengano rinegoziati.
Factory sslSocket	String a	Una stringa che rappresenta il nome classe completo di una classe che fornisce un'implementazione dell'interfaccia javax.net.ssl.SSLSocketFactory , includendo facoltativamente un argomento da passare al metodo constructor, racchiuso tra parentesi.	Tutte le connessioni stabilite nell'ambito dell'oggetto di destinazione gestito utilizzano i socket ottenuti da questa implementazione dell'interfaccia SSLSocketFactory .
statusRefreshInterval[1]	int	• **60000** • Qualsiasi numero intero positivo	L'intervallo, in millesimi di secondo, tra gli aggiornamenti della transazione di lunga durata che rileva quando un sottoscrittore perde la propria connessione al gestore code. Questa proprietà è rilevante solo se SUBSTORE ha il valore QUEUE.
subscriptionStore[1]	String a	• **BROKER** • MIGRAZIONE • CODA	Determina dove WebSphere MQ classes per JMS memorizza i dati persistenti relativi alle sottoscrizioni attive.
Corrispondenza targetClient	boolean o	• **true** • No	Se un messaggio di risposta, inviato alla coda identificata dal campo di intestazione JMSReplyTo di un messaggio in entrata, ha un'intestazione MQRFH2 solo se il messaggio in entrata ha un'intestazione MQRFH2 .

Tabella 98. Proprietà di un oggetto ConnectionFactory (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
temporaryModel	String a	• **SYSTEM.DEFAULT.MODEL.QUEUE** • SYSTEM.JMS.TEMPQ.MODEL • Qualsiasi stringa	Il nome della coda modello da cui vengono create le code temporanee JMS. Utilizzare SYSTEM.DEFAULT.MODEL.QUEUE se si verificano entrambe le seguenti condizioni: • L'applicazione utilizza una coda temporanea che accetterà messaggi non persistenti. • Solo un'applicazione creerà una coda temporanea sul gestore code a cui punta ConnectionFactory alla volta. Notare che SYSTEM.DEFAULT.MODEL.QUEUE può essere aperto solo da un'applicazione alla volta. Utilizzare SYSTEM.JMS.TEMPQ.MODEL. nelle seguenti situazioni: • Quando l'applicazione utilizza una coda temporanea che accetterà messaggi persistenti. • Se più applicazioni possono connettersi al gestore code a cui punta ConnectionFactory e tali applicazioni devono creare code temporanee contemporaneamente. Definire una nuova coda di modelli con l'attributo DEFPSIST impostato su YES e l'attributo DEFSOPT impostato su SHARED nella seguente situazione: • Quando l'applicazione utilizza una coda temporanea che accetta messaggi non persistenti e più applicazioni si connetteranno al gestore code a cui fa riferimento ConnectionFactory e tali applicazioni devono creare code temporanee contemporaneamente. Quando viene creata la nuova coda modello, impostare la proprietà temporaryModel sul nome della nuova coda modello.

Tabella 98. Proprietà di un oggetto ConnectionFactory (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
tempQPrefix	String a	• **"" (stringa vuota)** • Un prefisso che può essere utilizzato per formare il nome di una coda dinamica IBM WebSphere MQ . Le regole per formare il prefisso sono le stesse regole per formare il contenuto del campo *DynamicQName* in un descrittore di oggetti IBM WebSphere MQ , la struttura MQOD, ma l'ultimo carattere non vuoto deve essere un asterisco (*). Se il valore della proprietà è la stringa vuota, WebSphere MQ classes per JMS utilizza il valore AMQ.* quando si crea una coda dinamica.	Il prefisso utilizzato per formare il nome di una coda dinamica IBM WebSphere MQ .
TEMPTOPICPREFIX	String a	Qualsiasi stringa non null costituita solo da caratteri validi per una stringa di argomenti IBM WebSphere MQ	Durante la creazione di argomenti temporanei, JMS genera una stringa di argomenti nel formato "TEMP/*TEMPTOPICPREFIX*/*unique_id*" o, se questa proprietà viene lasciata con il valore predefinito, solo "TEMP/*unique_id*". La specifica di un TEMPTOPICPREFIX non vuoto consente la definizione di code modello specifiche per la creazione delle code gestite per i sottoscrittori di argomenti temporanei creati in questa connessione.
transportType	String a	• **client** • BIND • BINDING_THEN_CLIENT	Indica se una connessione a un gestore code utilizza la modalità client o la modalità bind. Se viene specificato il valore BINDINGS_THEN_CLIENT, l'adattatore di risorse tenta prima di stabilire una connessione in modalità bind. Se questo tentativo di connessione non riesce, l'adattatore risorse tenta di stabilire una connessione in modalità client.
nome utente	String a	• **vuoto** • Un nome utente	Il nome utente predefinito da utilizzare quando si crea una connessione a un gestore code.
wildcardFormat	int	• CHAR - Riconosce solo i caratteri jolly, come utilizzato nel broker versione 1 • TOPIC - Riconosce solo i caratteri jolly a livello di argomento, come utilizzato nel broker versione 2	La versione della sintassi dei caratteri jolly da utilizzare.

Tabella 98. Proprietà di un oggetto ConnectionFactory (Continua)			
Nome della proprietà	**Tipo**	**Valori validi (valore predefinito in grassetto)**	**Descrizione**
Note: 1. Questa proprietà può essere utilizzata con la versione 7.0 di IBM WebSphere MQ classes for JMS ma non influisce su un'applicazione connessa a un gestore code della versione 7.0 a meno che la proprietà providerVersion non sia impostata su un numero di versione inferiore a 7. 2. Per informazioni importanti sull'utilizzo della proprietà sslFipsRequired, consultare "Limitazioni dell'adattatore di risorse IBM WebSphere MQ" a pagina 772. 3. Per informazioni su come configurare l'adattatore risorse in modo che possa individuare un'uscita, consultare "Configurazione di IBM WebSphere MQ classes for JMS per l'utilizzo delle uscite canale" a pagina 914.			

Il seguente esempio mostra una serie tipica di proprietà di un oggetto ConnectionFactory :

```
channel:        SYSTEM.DEF.SVRCONN
hostName:       192.168.0.42
port:           1414
queueManager:   ExampleQM
transportType:  CLIENT
```

Tabella 99 a pagina 764 elenca le proprietà comuni a un oggetto Coda e a un oggetto Argomento.

Tabella 99. Proprietà comuni a un oggetto Coda e a un oggetto Argomento			
Nome della proprietà	**Tipo**	**Valori validi (valore predefinito in grassetto)**	**Descrizione**
CCSID	Stringa	• **1208** • Un CCSID (coded character set identifier) supportato dalla JVM (Java virtual machine)	Il CCSID (coded character set identifier) per la destinazione.
codifica	Stringa	• **Nativa** • Una stringa di tre caratteri: – Il primo carattere specifica la rappresentazione dei numeri interi binari: - *N* indica la codifica normale. - *R* indica la codifica inversa. – Il secondo carattere specifica la rappresentazione dei numeri interi decimali compressi: - *N* indica la codifica normale. - *R* indica la codifica inversa. – Il terzo carattere specifica la rappresentazione dei numeri a virgola mobile: - *N* indica la codifica IEEE standard. - *R* indica la codifica IEEE inversa. - *3* indica la codifica zSeries . NATIVE è equivalente alla stringa NNN.	La rappresentazione di numeri interi binari, interi decimali compressi e numeri a virgola mobile per la destinazione.

Tabella 99. Proprietà comuni a un oggetto Coda e a un oggetto Argomento (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
scadenza	Stringa	• **APP** - Il tempo di scadenza di un messaggio è determinato dal produttore del messaggio. • UNLIM - Un messaggio non scade mai. • 0 - Un messaggio non scade mai. • Un numero intero positivo che rappresenta la scadenza di un messaggio in millesimi di secondo.	L'ora di scadenza di un messaggio inviato alla destinazione.
FAILIFQUIESCE	Stringa	• **true** • No	Indica se un tentativo di accesso alla destinazione non riesce se il gestore code è in uno stato di inattività.
persistenza	Stringa	• **APP** - La persistenza di un messaggio è determinata dal produttore del messaggio. • QDEF - La persistenza di un messaggio è determinata dall'attributo *DefPersistence* della coda WebSphere MQ . • PERS - Un messaggio è persistente. • NON - Un messaggio è non persistente. • HIGH - La persistenza di un messaggio è determinata dall'attributo *NonPersistentMessageClass* della coda WebSphere MQ in base alla spiegazione in "Messaggi persistenti JMS" a pagina 906.	La persistenza di un messaggio inviato alla destinazione.
priorità	Stringa	• **APP** - La priorità di un messaggio è determinata dal produttore del messaggio. • QDEF - La priorità di un messaggio è determinata dall'attributi *DefPriority* della coda IBM WebSphere MQ . • Un numero intero compreso tra 0, la priorità più bassa, e 9, la priorità più alta.	La priorità di un messaggio inviato alla destinazione.

Tabella 99. Proprietà comuni a un oggetto Coda e a un oggetto Argomento (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
PUTASYNCALLOWED	Stringa	• QUEUE - Determinare se gli inserimenti asincroni sono consentiti facendo riferimento alla definizione della coda. • TOPIC - Determinare se gli inserimenti asincroni sono consentiti facendo riferimento alla definizione dell'argomento. • DESTINATION - Determinare se gli inserimenti asincroni sono consentiti facendo riferimento alla definizione della coda o dell'argomento. • DISABLED - Gli inserimenti asincroni non sono consentiti. • ENABLED - Le operazioni di inserimento asincrone sono consentite.	Indica se ai produttori di messaggi è consentito utilizzare gli inserimenti asincroni per inviare messaggi a questa destinazione.
READAHEADALLOWED	int	• **DESTINATION** - Determina se la lettura anticipata è consentita facendo riferimento alla definizione della coda o dell'argomento. • DISABLED - La lettura anticipata non è consentita. • ENABLED - La lettura anticipata è consentita. • QUEUE - Determinare se la lettura anticipata è consentita facendo riferimento alla definizione della coda. • TOPIC - Determinare se la lettura anticipata è consentita facendo riferimento alla definizione argomento.	Indica se gli utenti dei messaggi e i browser delle code possono utilizzare la lettura anticipata per ottenere i messaggi non persistenti dalla destinazione in un buffer interno prima di riceverli.
receiveCCSID	int	• **0** - Utilizza JVM `Charset.defaultCharset` • 1208 - UTF-8 • Un CCSID (coded character set identifier) supportato	Proprietà di destinazione che imposta il CCSID di destinazione per la conversione del messaggio del gestore code. Il valore viene ignorato a meno che **`receiveConversion`** non sia impostato su `QMGR`
receiveConversion	Stringa	• **MSG CLI** • QMGR	Proprietà di destinazione che determina se la conversione dati verrà eseguita dal gestore code.
targetClient	Stringa	• **JMS** - La destinazione di un messaggio è un'applicazione JMS. • MQ - La destinazione di un messaggio è un'applicazione IBM WebSphere MQ non JMS.	Indica se la destinazione di un messaggio inviato alla destinazione è un'applicazione JMS. Un messaggio con una destinazione che è un'applicazione JMS contiene un'intestazione MQRFH2 .

Tabella 100 a pagina 767 elenca le proprietà specifiche di un oggetto Coda.

Tabella 100. Proprietà specifiche di un oggetto Coda

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
baseQueueManagerName	String a	• **"" (stringa vuota)** • Un nome gestore code	Il nome del gestore code proprietario della coda IBM WebSphere MQ sottostante.
Nome baseQueue	String a	• **"" (stringa vuota)** • Un nome coda	Il nome della coda IBM WebSphere MQ sottostante.

Tabella 101 a pagina 767 elenca le proprietà specifiche per un oggetto Argomento.

Tabella 101. Proprietà specifiche di un oggetto argomento

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
Nome baseTopic	String a	• **"" (stringa vuota)** • Un nome argomento	Il nome dell'argomento sottostante.
brokerCCDurSubQueue[1]	String a	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Un nome coda	Il nome della coda da cui un utente della connessione riceve messaggi di sottoscrizione durevoli.
brokerDurSubQueue[1]	String a	• **SYSTEM.JMS.D.SUBSCRIBER.QUEUE** • Un nome coda	Il nome della coda da cui un sottoscrittore di argomenti durevoli riceve i messaggi. Per ulteriori informazioni, consultare la proprietà BROKEDURRSUBQ nella documentazione di WebSphere MQ Explorer.
brokerPubQueue[1]	String a	• **Non impostato** • Un nome coda	Il nome della coda in cui vengono inviati i messaggi pubblicati (la coda di flusso). Il valore di questa proprietà sovrascrive il valore della proprietà brokerPubQueue dell'oggetto ConnectionFactory . Tuttavia, se non si imposta il valore di questa proprietà, viene utilizzato il valore della proprietà brokerPubQueue dell'oggetto ConnectionFactory .
brokerPubQueueManager[1]	String a	• **"" (stringa vuota)** • Un nome gestore code	Il nome del gestore code proprietario della coda in cui vengono inviati i messaggi pubblicati sull'argomento.

Tabella 101. Proprietà specifiche di un oggetto argomento (Continua)

Nome della proprietà	Tipo	Valori validi (valore predefinito in grassetto)	Descrizione
brokerVersion[1]	String a	• **Non impostato** • 1 • 2	La versione del broker utilizzato. Il valore di questa proprietà sovrascrive il valore della proprietà brokerVersion dell'oggetto ConnectionFactory . Tuttavia, se non si imposta il valore di questa proprietà, viene utilizzato il valore della proprietà brokerVersion dell'oggetto ConnectionFactory .

Nota:

1. Questa proprietà può essere utilizzata con la versione 7.0 di IBM WebSphere MQ classes for JMS , ma non influisce su un'applicazione connessa a un gestore code della versione 7.0 a meno che la proprietà providerVersion dell'oggetto ConnectionFactory non sia impostata su un numero di versione inferiore a 7.

Il seguente esempio mostra una serie di proprietà di un oggetto Coda:

```
expiry:                 UNLIM
persistence:            QDEF
baseQueueManagerName:   ExampleQM
baseQueueName:          SYSTEM.JMS.TEMPQ.MODEL
```

Il seguente esempio mostra una serie di proprietà di un oggetto Argomento:

```
expiry:                 UNLIM
persistence:            NON
baseTopicName:          myTestTopic
```

Attività correlate

Specifica che solo i CipherSpecs certificati FIPS vengono utilizzati al runtime sul client MQI

Riferimenti correlati

Federal Information Processing Standards (FIPS) per UNIX, Linux e Windows

V 7.5.0.9 *Configurazione della proprietà di corrispondenza targetClientper una specifica di attivazione*

È possibile configurare la propriet. **targetClientMatching** per una specifica di attivazione in modo che un'intestazione MQRFH2 sia inclusa nei messaggi di risposta quando i messaggi di richiesta non contengono un'intestazione MQRFH2 . Ciò significa che tutte le proprietà del messaggio che un'applicazione definisce su un messaggio di risposta vengono incluse quando il messaggio viene inviato.

Informazioni su questa attività

Se un'applicazione MDB (Message - Driven Bean) utilizza messaggi che non contengono un'intestazione MQRFH2 , tramite una specifica di attivazione dell'adattatore di risorse IBM WebSphere MQ JCA, e successivamente invia messaggi di risposta alla destinazione JMS creata dal campo JMSReplyTo del messaggio di richiesta, i messaggi di risposta devono includere un'intestazione MQRFH2 , anche se i messaggi di richiesta non lo sono, altrimenti le proprietà del messaggio che l'applicazione ha definito in un messaggio di risposta vengono perse.

La proprietà **targetClientMatching** definisce se un messaggio di risposta, inviato alla coda identificata dal campo di intestazione JMSReplyTo di un messaggio in entrata, ha un'intestazione MQRFH2 solo se il messaggio in entrata ha un'intestazione MQRFH2 . Puoi configurare questa proprietà per una specifica di attivazione, sia in WebSphere Application Server tradizionale che in WebSphere Application Server Liberty.

Se si imposta il valore della proprietà **targetClientMatching** su `false`, è possibile includere un'intestazione MQRFH2 in un messaggio di risposta inviato a una destinazione JMS creata dall'intestazione JMSReplyTo di un messaggio di richiesta in entrata che non contenga un MQRFH2. Ciò è dovuto al fatto che la proprietà **targetClient** sulla destinazione JMS viene impostata sul valore `0`, che significa che i messaggi contengono un'intestazione MQRFH2 . La presenza dell'intestazione MQRFH2 nel messaggio in uscita consente la memoria delle proprietà del messaggio definite dall'utente sul messaggio quando viene inviato alla coda IBM WebSphere MQ .

Se la proprietà **targetClientMatching** è impostata su `true` e un messaggio di richiesta non include un'intestazione MQRFH2 , un'intestazione MQRFH2 non viene inclusa nel messaggio di risposta.

Procedura

- In WebSphere Application Server tradizionale, utilizzare la console di amministrazione per definire la proprietà **targetClientMatching** come proprietà personalizzata nella specifica di attivazione IBM WebSphere MQ :
  a) Nel pannello di navigazione, fare clic su **Risorse -> JMS -> Specifiche attivazione**.
  b) Selezionare il nome della specifica di attivazione che si desidera visualizzare o modificare.
  c) Fare clic su **Proprietà personalizzate -> Nuovo** e immettere i dettagli della nuova proprietà personalizzata.
     Impostare il nome della proprietà su `targetClientMatching`, il tipo su `java.lang.Boolean` e il valore su `false`.
- In WebSphere Application Server Liberty, specifica la proprietà **targetClientMatching** nella definizione di una specifica di attivazione all'interno di `server.xml`.
  Ad esempio:

```
<jmsActivationSpec id="SimpleMDBApplication/SimpleEchoMDB/SimpleEchoMDB">
<properties.wmqJms destinationRef="MDBRequestQ"
queueManager="MY_QMGR" transportType="BINDINGS" targetClientMatching="false"/>
<authData password="********" user="tom"/>
</jmsActivationSpec>
```

Concetti correlati

“Creazione di destinazioni in una applicazione JMS” a pagina 879
Invece di richiamare le destinazioni come oggetti gestiti da uno spazio nomi JNDI (Java Naming and Directory Interface), un'applicazione JMS può utilizzare una sessione per creare le destinazioni in modo dinamico durante il runtime. Un'applicazione può utilizzare un URI (uniform resource identifier) per identificare una coda WebSphere MQ o un argomento e, facoltativamente, per specificare una o più proprietà di un oggetto Coda o Argomento.

“Configurazione dell'adattatore di risorse per la comunicazione in uscita” a pagina 753
Per configurare la comunicazione in uscita, definire le proprietà di un oggetto ConnectionFactory e di un oggetto di destinazione gestito.

Modalità ASF e non ASF
La modalità ASF (Application Server Facilities) è il metodo predefinito con cui il servizio listener dei messaggi in WebSphere Application Server elabora i messaggi.

Il servizio listener dei messaggi ha due modalità operative, ASF (Application Server Facilities) e non - ASF (non - Application Server Facilities):

- La modalità ASF fornisce il supporto simultaneo e transazionale per le applicazioni. Per i bean di unità messaggi di pubblicazione / sottoscrizione, la modalità ASF fornisce una migliore velocità di trasmissione e simultaneità, poiché in modalità non ASF il listener è a thread singolo.
- La modalità non - ASF è principalmente utilizzata con provider di messaggistica di terze parti che non supportano JMS ASF, che è un'estensione facoltativa della specifica JMS. Anche la modalità non ASF è transazionale ma, poiché la lunghezza del percorso è più breve rispetto alla modalità ASF, di solito fornisce prestazioni migliori.

Per abilitare la modalità non ASF dell'operazione per tutti i listener bean basati sui messaggi sul server delle applicazioni, impostare questa proprietà su un valore diverso da zero.

Nota:

La modalità non ASF non può essere selezionata sui sistemi z/OS , pertanto in questo caso non è necessario impostare un valore diverso da zero per questa proprietà.

Elaborazione dei messaggi in modalità ASF

In modalità ASF, le sessioni e i thread del server vengono assegnati per il lavoro solo quando viene rilevato un messaggio adatto per l'MDB (message - driven bean). Il numero di thread che un MDB può elaborare simultaneamente è determinato dal valore della proprietà **Maximum Sessions** per la porta listener o la specifica di attivazione.

Elaborazione messaggi in modalità non - ASF

In modalità non ASF, i thread sono attivi dal momento in cui viene avviata la porta del listener o la specifica di attivazione. Il numero di thread attivi è determinato dal valore specificato per la proprietà **Maximum Sessions** . Il numero di thread specificato nella proprietà **Maximum Sessions** è attivo, indipendentemente dal numero di messaggi disponibili per l'elaborazione. Ogni thread attivo è una connessione di rete fisica individuale.

IBM WebSphere MQ Versione 7.0 o successiva consente di avere fino a dieci thread che condividono una singola connessione di rete fisica.

Concetti correlati

IBM WebSphere MQ classes per JMS Application Server Facilities
Questo argomento descrive il modo in cui WebSphere MQ classes per JMS implementa la classe ConnectionConsumer e la funzionalità avanzata nella classe Session. Riepiloga anche la funzione di un pool di sessioni server.

Attività correlate

Configurazione delle specifiche di attivazione per la modalità non - ASF
Le specifiche di attivazione sono il modo standardizzato per gestire e configurare la relazione tra un MDB (message driven bean) in esecuzione in WebSphere Application Server e una destinazione in IBM WebSphere MQ. In questa sezione viene illustrato come configurare WebSphere Application Server per utilizzare la modalità non ASF per elaborare i messaggi.

Informazioni correlate

Elaborazione dei messaggi in modalità ASF e non ASF

Configurazione delle specifiche di attivazione per la modalità non ASF
Le specifiche di attivazione sono il modo standardizzato per gestire e configurare la relazione tra un MDB (message driven bean) in esecuzione in WebSphere Application Server e una destinazione in IBM WebSphere MQ. In questa sezione viene illustrato come configurare WebSphere Application Server per utilizzare la modalità non ASF per elaborare i messaggi.

Prima di iniziare

Il modo in cui vengono definite le proprietà di una specifica di attivazione dipende dalle interfacce di gestione fornite dal server delle applicazioni. Questa attività presuppone che si stia utilizzando WebSphere Application Server versione 7 o successiva come server delle applicazioni e IBM WebSphere MQ come provider di messaggistica.

Nota:

Non è possibile selezionare la modalità non ASF sui sistemi z/OS .

Informazioni su questa attività

Le proprietà di una specifica di attivazione determinano il modo in cui un MDB (message drive bean) riceve i messaggi JMS da una coda IBM WebSphere MQ . Per configurare la modalità non ASF, definire le proprietà di una o più specifiche di attivazione.

Esistono diverse configurazioni IBM WebSphere MQ che è possibile utilizzare in modalità non - ASF. Con le seguenti configurazioni ogni thread utilizza una connessione di rete fisica separata:

- Un gestore code IBM WebSphere MQ Versione 7.x , utilizzando una factory di connessione con la proprietà della versione del provider impostata su 6.
- Un gestore code IBM WebSphere MQ Versione 7.x , che utilizza un factory di connessione con la proprietà della versione del provider impostata su 7 o non specificata, che si connette su un canale IBM WebSphere MQ che ha il parametro **SHARECNV** (conversazioni condivise) impostato su 0.

Per configurare non - ASF, impostare la proprietà ActivationSpec **NON.ASF.RECEIVE.TIMEOUT** su un numero intero positivo, che indica che viene utilizzata la consegna non ASF. Il valore è il tempo, in millesimi di secondo, che una richiesta get attende per i messaggi che potrebbero non essere ancora arrivati (una chiamata get with wait). Il valore predefinito, 0, indica che viene utilizzata la consegna ASF. Per ulteriori dettagli, consultare **Proprietà personalizzate del servizio listener dei messaggi**.

Questo parametro è valido solo quando l'applicazione è in esecuzione su WebSphere Application Server versione 7 o successiva.

Procedura

1. Avviare la console di gestione WebSphere Application Server .
2. Visualizzare la pagina delle impostazioni del servizio listener:
   a) Nel riquadro di navigazione, selezionare **Server> Tipo di server> WebSphere application servers**.
   b) Nel riquadro del contenuto, fare clic sul nome del server delle applicazioni.
   c) In **Communications**, fare clic su **Messaging> Message Listener Service**.
3. Impostare la proprietà personalizzata **NON.ASF.RECEIVE.TIMEOUT** come proprietà personalizzate del servizio listener dei messaggi.
   a) Fare clic su **Proprietà personalizzate**.
   b) Fare clic su **Nuovo**.
   c) Immettere il nome della proprietà **NON.ASF.RECEIVE.TIMEOUT**nel campo **Nome** .
   d) Immettere il valore richiesto nel campo **Valore** .
   e) Fare clic su **OK**.
4. Salvare le modifiche alla configurazione principale.
5. Per attivare la configurazione modificata, arrestare e riavviare il server delle applicazioni.

Risultati

Sono state configurate le proprietà del servizio listener dei messaggi per WebSphere Application Server per utilizzare la modalità non ASF.

Nota: Quando si utilizza la modalità non ASF, è necessario assicurarsi di consentire un periodo di tempo sufficiente per il completamento dell'elaborazione prima che venga raggiunto il timeout della durata totale della transazione, in modo da evitare timeout della transazione indesiderati. Per ulteriori dettagli, fare riferimento a **NON.ASF.RECEIVE.TIMEOUT** nella documentazione del prodotto WebSphere Application Server .

Concetti correlati

“Modalità ASF e non ASF” a pagina 769
La modalità ASF (Application Server Facilities) è il metodo predefinito con cui il servizio listener dei messaggi in WebSphere Application Server elabora i messaggi.

Configurazione dell'adattatore risorse per la comunicazione in entrata

Per configurare la comunicazione in entrata, definire le proprietà di uno o più oggetti ActivationSpec .

Informazioni correlate

Bean basati sui messaggi

Servizio listener messaggi

Elaborazione dei messaggi in modalità ASF e non ASF

Modalità di elaborazione dei messaggi in modalità non - ASF

Limitazioni dell'adattatore di risorse IBM WebSphere MQ

Quando si utilizza l'adattatore di risorse IBM WebSphere MQ , alcune funzioni di IBM WebSphere MQ non sono disponibili o sono limitate.

L'adattatore di risorse IBM WebSphere MQ ha i seguenti limiti:

- L'adattatore di risorse IBM WebSphere MQ è supportato su tutte le piatteforme IBM WebSphere MQ , ad eccezione di z/OS.
- L'adattatore di risorse IBM WebSphere MQ non supporta connessioni in tempo reale a un broker. Supporta solo connessioni a un gestore code IBM WebSphere MQ in modalità client o bind.
- L'adattatore risorse IBM WebSphere MQ non supporta i programmi di uscita del canale scritti in linguaggi diversi da Java.
- Mentre un server delle applicazioni è in esecuzione, il valore della proprietà sslFipsRequired deve essere true per tutte le risorse JCA o false per tutte le risorse JCA. Questo è un requisito anche se le risorse JCA non vengono utilizzate contemporaneamente. Se la proprietà sslFipsRequired ha valori differenti per diverse risorse JCA, IBM WebSphere MQ emette il codice motivo MQRC_UNSUPPORTED_CIPHER_SUITE, anche se non si sta utilizzando una connessione SSL.
- Non è possibile specificare più di un keystore per un server delle applicazioni. Se le connessioni vengono effettuate a più di un gestore code, tutte le connessioni devono utilizzare lo stesso keystore. Questa limitazione non si applica a WebSphere Application Server.
- Se si utilizza una tabella di definizione del canale client (CCDT) con più di una definizione di canale di connessione client adatta, in caso di errore l'adattatore di risorse potrebbe selezionare una definizione di canale differente e quindi un gestore code differente dalla CCDT, il che causerebbe problemi per il ripristino della transazione. L'adattatore di risorse non intraprende alcuna azione per impedire l'utilizzo di tale configurazione ed è responsabilità dell'utente evitare configurazioni che potrebbero causare problemi per il ripristino della transazione.
- La funzionalità di nuovi tentativi di connessione introdotta in IBM WebSphere MQ Version 7.0.1 non è supportata per connessioni in uscita quando in esecuzione in un contenitore JEE (EJB/Servlet). Il nuovo tentativo di connessione non è supportato per JMS in uscita quando l'adattatore viene utilizzato in un contesto del contenitore JEE, indipendentemente dalla configurazione della transazione o per l'utilizzo non transazionale.

Attività correlate

Specifica che solo i CipherSpecs certificati FIPS vengono utilizzati al runtime sul client MQI

Riferimenti correlati

Federal Information Processing Standards (FIPS) per UNIX, Linux e Windows

Impostazione post - installazione per le classi WebSphere MQ per le applicazioni JMS

Questo argomento indica le autorizzazioni WebSphere MQ per le applicazioni JMS necessarie per accedere alle risorse di un gestore code. Inoltre, introduce le modalità di connessione e descrive come configurare un gestore code in modo che le applicazioni possano connettersi in modalità client.

Verificare il file readme WebSphere MQ . Potrebbe contenere informazioni che sostituiscono le informazioni contenute in questo argomento.

Oggetti utilizzati da JMS che necessitano di autorizzazione per utenti non privilegiati

Gli utenti non privilegiati devono disporre dell'autorizzazione per accedere alle code utilizzate da JMS. Ogni applicazione JMS richiede l'autorizzazione per il gestore code con cui funziona.

Per i dettagli sul controllo accessi in IBM WebSphere MQ, consultare Impostazione della sicurezza su Windows, UNIX and Linux systems .

WebSphere Le classi di MQ per applicazioni JMS richiedono l'autorizzazione `connect` e `inq` per il gestore code. È possibile impostare le autorizzazioni appropriate utilizzando il comando di controllo **setmqaut** , ad esempio:

```
setmqaut -m QM1 -t qmgr -g jmsappsgroup +connect +inq
```

Per il dominio point - to - point, sono richieste le seguenti autorizzazioni:

- Le code utilizzate dagli oggetti MessageProducer richiedono l'autorità `put` .
- Le code utilizzate dagli oggetti MessageConsumer e QueueBrowser richiedono le autorizzazioni `get`, `inq`e `browse` .
- Il metodo QueueSession.createTemporaryQueue () deve accedere alla coda modello specificata dalla proprietà TEMPMODEL dell'oggetto factory QueueConnection. Per impostazione predefinita, questa coda modello è SYSTEM.TEMP.MODEL.QUEUE.

Se una di queste code è una coda alias, le relative code di destinazione richiedono l'autorizzazione di interrogazione. Se la coda di destinazione è una coda cluster, è necessaria anche l'autorizzazione di ricerca.

Per il dominio di pubblicazione / sottoscrizione, le seguenti code vengono utilizzate se le classi WebSphere MQ per JMS si collegano a un gestore code IBM WebSphere MQ in modalità di migrazione del provider di messaggistica 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

Per ulteriori informazioni sulla modalità di migrazione del provider di messaggistica IBM WebSphere MQ , consultare Quando utilizzare **PROVIDERVERSION**

Inoltre, se le classi WebSphere MQ per JMS si collegano a un gestore code in questa modalità, qualsiasi applicazione che pubblica i messaggi deve accedere alla coda di flusso specificata dal factory TopicConnectiono dall'oggetto argomento. Per impostazione predefinita, questa coda è SYSTEM.BROKER.DEFAULT.STREAM.

Se si utilizza il provider di messaggistica ConnectionConsumer, IBM WebSphere MQ Resource Adapter o WebSphere Application Server IBM WebSphere MQ , potrebbe essere necessaria un'autorizzazione aggiuntiva.

Le code che devono essere lette da ConnectionConsumer devono avere le autorizzazioni `get` , `inq`e `browse` . La coda di messaggi non recapitabili del sistema e qualsiasi coda di backout-requeue o coda di report utilizzata da ConnectionConsumer devono disporre delle autorizzazioni `put` e `passall` .

Quando un'applicazione utilizza la modalità normale del provider di messaggistica WebSphere MQ per eseguire la messaggistica di pubblicazione / sottoscrizione, l'applicazione utilizza la funzionalità di pubblicazione / sottoscrizione integrata fornita dal gestore code. Per informazioni sulla protezione degli argomenti e delle code utilizzati, consultare Sicurezza di pubblicazione / sottoscrizione .

Modalità di connessione per le classi WebSphere MQ per JMS

Un'applicazione WebSphere MQ classes può connettersi a un gestore code in modalità client o bind. In modalità client, WebSphere MQ classes per JMS si connette al gestore code su TCP/IP. In modalità bind, WebSphere MQ classes for JMS si collega direttamente al gestore code mediante JNI (Java Native Interface).

Un'applicazione in esecuzione in WebSphere Application Server su z/OS può connettersi a un gestore code in modalità di bind o client, ma un'applicazione in esecuzione in qualsiasi altro ambiente su z/OS può connettersi a un gestore code solo in modalità di bind. Un'applicazione in esecuzione su qualsiasi altra piattaforma può connettersi a un gestore code in modalità bind o client.

È possibile utilizzare la versione corrente o precedente supportata di WebSphere MQ classes per JMS con un gestore code corrente ed è possibile utilizzare una versione corrente o precedente supportata del gestore code con la versione corrente di WebSphere MQ classes per JMS. Se si mischiano versioni differenti, la funzione è limitata al livello della versione precedente.

Le seguenti sezioni descrivono ogni modalità di connessione in modo più dettagliato.

Modalità client

Per connettersi a un gestore code in modalità client, un'applicazione WebSphere MQ classes per JMS può essere eseguita sullo stesso sistema su cui è in esecuzione il gestore code o su un sistema diverso. In ogni caso, WebSphere MQ classes for JMS si connette al gestore code su TCP/IP.

Modalità bind

Per connettersi a un gestore code in modalità bind, un'applicazione WebSphere MQ classes per JMS deve essere eseguita sullo stesso sistema su cui è in esecuzione il gestore code.

Le classi WebSphere MQ per JMS si collegano direttamente al gestore code utilizzando JNI (Java Native Interface). Per utilizzare il trasporto dei collegamenti, le classi WebSphere MQ per JMS devono essere eseguite in un ambiente che ha accesso alle librerie WebSphere MQ Java Native Interface; consultare “Configurazione delle librerie JNI (Java Native Interface)” a pagina 726 per ulteriori informazioni.

Le classi WebSphere MQ per JMS supportano i seguenti valori per *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

Per cambiare le opzioni di connessione utilizzate dalle classi WebSphere MQ per JMS, modificare la proprietà del factory di connessione CONNOPT.

Per ulteriori informazioni sulle opzioni di connessione, consultare “Connessione a un gestore code utilizzando la chiamata MQCONNX” a pagina 209

Per utilizzare il trasporto dei collegamenti, il Java Runtime Environment utilizzato deve supportare il CCSID (Coded Character Set Identifier) del gestore code a cui si collegano le classi WebSphere MQ per JMS .

I dettagli su come determinare quali CCSID sono supportati da un JRE (Java Runtime Environment) possono essere trovati in WebSphere MQ FDC con ID probe 21 generato quando si utilizzano le classi WebSphere MQ V7 per Java o WebSphere MQ V7 per JMS .

Bind, quindi modalità client

Questa è l'opzione predefinita. Quando ci si connette a un gestore code in questa modalità, un'applicazione WebSphere MQ classes per JMS tenterà di connettersi in modalità bind, il che richiede che il gestore code risieda sulla stessa macchina dell'applicazione. Se la connessione non ha esito positivo, l'applicazione tenterà di connettersi in modalità client, consentendo al gestore code di risiedere localmente sulla stessa macchina dell'applicazione o in remoto.

Configurazione del gestore code in modo che WebSphere MQ classes per applicazioni JMS possa connettersi in modalità client

Per configurare il gestore code in modo che le classi WebSphere MQ per le applicazioni JMS possano connettersi in modalità client, è necessario creare una definizione di canale di connessione server e avviare un listener.

Su z/OS, è necessario installare la funzione Client Attachment.

Creazione di una definizione di canale di connessione server

Su tutte le piatteforme, è possibile utilizzare il comando MQSC DEFINE CHANNEL per creare una definizione di canale di connessione server. Vedi il seguente esempio:

```
DEFINE CHANNEL(JAVA.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)
```

In IBM i, è possibile utilizzare il comando CL CRTMQMCHL, come nel seguente esempio:

```
CRTMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN)
                                TRPTYPE(*TCP)
                                MQMNAME(QMGRNAME)
```

In questo comando, *QMGRNAME* è il nome del gestore code.

È anche possibile creare una definizione di canale di connessione server utilizzando IBM WebSphere MQ Explorer, che viene eseguito su Linux e Windows, o le operazioni e i pannelli di controllo su z/OS.

Il nome del canale (JAVA.CHANNEL negli esempi precedenti) deve corrispondere al nome canale specificato dalla proprietà CHANNEL della factory di connessione utilizzata dall'applicazione per connettersi al gestore code. Il valore predefinito della proprietà CHANNEL è SYSTEM.DEF.SVRCONN.

Avvio di un listener

È necessario avviare un listener per il gestore code se non ne è già stato avviato uno.

Su tutte le piattaforme, è possibile utilizzare il comando MQSC START LISTENER per avviare un listener ma, ad eccezione di z/OS, è necessario creare prima un oggetto listener utilizzando il comando MQSC DEFINE LISTENER. Vedi il seguente esempio:

```
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) PORT(1414)
START LISTENER(LISTENER.TCP)
```

Su sistemi UNIX, Linuxe Windows , è anche possibile utilizzare il comando di controllo **runmqlsr** per avviare un listener, come nel seguente esempio:

```
runmqlsr -t tcp -p 1414 -m QMgrName
```

In questo comando, *QMgrName* è il nome del gestore code.

È anche possibile avviare un listener utilizzando WebSphere MQ Explorer, che viene eseguito su Linux e su Windows, o le operazioni e i pannelli di controllo su z/OS.

Il numero della porta su cui il listener è in ascolto deve corrispondere al numero di porta specificato dalla proprietà PORT della factory di connessione utilizzata dall'applicazione per connettersi al gestore code. Il valore predefinito della proprietà PORT è 1414.

Il test di verifica dell'installazione point-to-point per WebSphere MQ classes per JMS

Un programma IVT (point - to - point installation verification test) viene fornito con WebSphere MQ classes per JMS. Il programma si connette a un gestore code in modalità bind o client, invia un messaggio alla coda denominata SYSTEM.DEFAULT.LOCAL.QUEUEe riceve il messaggio dalla coda. Il programma può creare e configurare tutti gli oggetti richiesti in modo dinamico al runtime oppure può utilizzare JNDI per richiamare gli oggetti gestiti da un servizio di directory.

Eseguire il test di verifica dell'installazione senza utilizzare prima JNDI perché il test è autonomo e non richiede l'uso di un servizio directory. Per una descrizione degli oggetti gestiti, consultare “Tipi di oggetto JMS” a pagina 941.

Il test di verifica dell'installazione point-to-point senza utilizzare JNDI

In questo test, il programma IVT crea e configura tutti gli oggetti che richiede dinamicamente al runtime e non utilizza JNDI.

Viene fornito uno script per eseguire il programma IVT. Lo script è denominato IVTRun sui sistemi UNIX and Linux e IVTRun.bat su Windowse si trova nella sottodirectory `bin` della directory di installazione di WebSphere MQ classes per JMS.

Per eseguire il test in modalità bind, immettere il seguente comando:

```
IVTRun -nojndi [-m qmgr] [-v providerVersion] [-t]
```

Per eseguire la verifica in modalità client, configurare prima il gestore code come descritto in “Preparazione ed esecuzione dei programmi di esempio” a pagina 110. Tenere presente che il canale da utilizzare assume il valore predefinito **SYSTEM.DEF.SVRCONN** e la coda da utilizzare è **SYSTEM.DEFAULT.LOCAL.QUEUE**, quindi immettere il seguente comando:

```
IVTRun -nojndi -client -m qmgr -host hostname [-port port] [-channel channel]
       [-v providerVersion] [-ccsid ccsid] [-t]
```

Non viene fornito alcuno script equivalente su sistemi z/OS , ma è possibile eseguire l'IVT in modalità bind richiamando direttamente la classe Java, utilizzando il seguente comando:

```
java com.ibm.mq.jms.MQJMSIVT -nojndi [-m qmgr] [-v providerVersion] [-t]
```

Il percorso classi deve contenere com.ibm.mqjms.jar.

I parametri sui comandi hanno i seguenti significati:

-m *gestore code*
Il nome del gestore code a cui si connette il programma IVT. Se si esegue la verifica in modalità bind e si omette questo parametro, il programma IVT si connette al gestore code predefinito.

-host *nome host*
Il nome host o l'indirizzo IP del sistema su cui è in esecuzione il gestore code.

-port *porta*
Il numero della porta su cui il listener del gestore code è in ascolto. Il valore predefinito è 1414.

-channel *canale*
Il nome del canale MQI che il programma IVT utilizza per connettersi al gestore code. Il valore predefinito è SYSTEM.DEF.SVRCONN.

-v *providerVersion*
Il livello di rilascio del gestore code a cui il programma IVT prevede di connettersi.

Questo parametro viene utilizzato per impostare la proprietà PROVIDERVERSION di un oggetto factory MQQueueConnectione ha gli stessi valori validi della proprietà PROVIDERVERSION. Per ulteriori informazioni su questo parametro, inclusi i relativi valori validi, consultare la descrizione della proprietà PROVIDERVERSION in Proprietà degli oggetti IBM WebSphere MQ classes for JMS .

Il valore predefinito è `unspecified`.

-ccsid *idccs*
L'identificativo (CCSID) della serie di caratteri codificati, o codepage, che deve essere utilizzata dal collegamento. Il valore predefinito è 819.

-t
La traccia è attivata. Per impostazione predefinita, la traccia è disattivata.

Un test riuscito produce un output simile al seguente output di esempio:

```
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
```

Il test di verifica dell'installazione point-to-point utilizzando JNDI

In questa verifica, il programma IVT utilizza JNDI per richiamare gli oggetti gestiti da un servizio directory.

Prima di poter eseguire il test, è necessario configurare un servizio di directory basato su un server LDAP (Lightweight Directory Access Protocol) o sul file system locale. È inoltre necessario configurare lo strumento di gestione WebSphere MQ JMS in modo che possa utilizzare il servizio directory per memorizzare gli oggetti gestiti. Per ulteriori informazioni su questi prerequisiti, consultare “Prerequisiti per le classi WebSphere MQ per JMS” a pagina 718. Per informazioni su come configurare lo strumento di gestione JMS WebSphere MQ , consultare “Configurazione dello strumento di amministrazione JMS” a pagina 937.

Il programma IVT deve essere in grado di utilizzare JNDI per richiamare un oggetto factory MQQueueConnectione un oggetto MQQueue dal servizio directory. Viene fornito uno script per creare questi oggetti gestiti. Lo script è denominato IVTSetup sui sistemi UNIX and Linux e IVTSetup.bat su Windowse si trova nella sottodirectory `bin` della directory di installazione di WebSphere MQ classes per JMS. Per eseguire lo script, immettere il comando seguente:

```
IVTSetup
```

Lo script richiama lo strumento di amministrazione JMS di WebSphere MQ per creare gli oggetti amministrati.

L'oggetto factory MQQueueConnectionè collegato con il nome ivtQCF e viene creato con i valori predefiniti per tutte le proprietà, il che significa che il programma IVT viene eseguito in modalità bind e si connette al gestore code predefinito. Se si desidera che il programma IVT venga eseguito in modalità client o che si connetta a un gestore code diverso da quello predefinito, è necessario utilizzare lo strumento di amministrazione JMS WebSphere MQ o WebSphere MQ Explorer per modificare le proprietà appropriate dell'oggetto factory MQQueueConnection. Per informazioni su come utilizzare lo strumento di gestione JMS WebSphere MQ , consultare “Utilizzo dello strumento di amministrazione JMS WebSphere MQ” a pagina 936. Per informazioni su come utilizzare WebSphere MQ Explorer, consultare la guida fornita con WebSphere MQ Explorer.

L'oggetto MQQueue è collegato al nome ivtQ e viene creato con i valori predefiniti per tutte le proprie proprietà, tranne per la proprietà QUEUE, che ha il valore SYSTEM.DEFAULT.LOCAL.QUEUE.

Una volta creati gli oggetti gestiti, è possibile eseguire il programma IVT. Per eseguire il test utilizzando JNDI, immettere il seguente comando:

```
IVTRun -url "providerURL" [-icf initCtxFact] [-t]
```

I parametri sul comando hanno i seguenti significati:

-url "*providerURL*"
L'URL (uniform resource locator) del servizio directory. L'URL può avere uno dei formati seguenti:

- `ldap://hostname/contextName`, per un servizio di directory basato su un server LDAP
- `file:/directoryPath`, per un servizio di directory basato sul file system locale

Notare che è necessario racchiudere l'URL tra virgolette (").

-icf *initCtxFact*
Il nome classe del factory di contesto iniziale, che deve essere uno dei seguenti valori:

- `com.sun.jndi.ldap.LdapCtxFactory`, per un servizio di directory basato sul server LDAP. Questo è il valore predefinito.
- `com.sun.jndi.fscontext.RefFSContextFactory`, per un servizio directory basato sul file system locale.

-t
La traccia è attivata. Per impostazione predefinita, la traccia è disattivata.

Un test riuscito produce un output simile a quello per un test riuscito senza utilizzare JNDI. La differenza principale è che l'output indica che la verifica sta utilizzando JNDI per richiamare un oggetto factory MQQueueConnectione un oggetto MQQueue.

Sebbene non sia strettamente necessario, si consiglia di riordinare dopo il test eliminando gli oggetti gestiti creati dallo script IVTSetup. Per questo scopo viene fornito uno script. Lo script è denominato IVTTidy sui sistemi UNIX and Linux e IVTTidy.bat su Windows, e si trova nella sottodirectory `bin` della directory di installazione di WebSphere MQ classes per JMS.

Determinazione dei problemi per il test di verifica dell'installazione point - to - point

Il test di verifica dell'installazione potrebbe avere esito negativo per i seguenti motivi:

- Se il programma IVT scrive un messaggio che indica che non è possibile trovare una classe, verificare che il percorso di classe sia impostato correttamente, come descritto in "Variabili di ambiente utilizzate dalle classi IBM WebSphere MQ per JMS" a pagina 724.
- Il test potrebbe avere esito negativo con il seguente messaggio:

```
Failed to connect to queue manager 'qmgr' with connection mode 'connMode'
and host name 'hostname'
```

  e un codice di errore associato 2059. Le variabili nel messaggio hanno i seguenti significati:

  qmgr
  Il nome del gestore code a cui il programma IVT sta tentando di connettersi. Questo inserimento del messaggio è vuoto se il programma IVT sta tentando di connettersi al gestore code predefinito in modalità bind.

  connMode
  La modalità di connessione, che è `Bindings` o `Client`.

  nomehost
  Il nome host o l'indirizzo IP del sistema su cui è in esecuzione il gestore code.

  Questo messaggio indica che il gestore code a cui il programma IVT sta tentando di connettersi non è disponibile. Verificare che il gestore code sia in esecuzione e, se il programma IVT sta tentando di connettersi al gestore code predefinito, accertarsi che il gestore code sia definito come gestore code predefinito per il sistema.
- Il test potrebbe avere esito negativo con il seguente messaggio:

```
Failed to open MQ queue 'SYSTEM.DEFAULT.LOCAL.QUEUE'
```

  Questo messaggio indica che la coda SYSTEM.DEFAULT.LOCAL.QUEUE non esiste sul gestore code a cui è connesso il programma IVT. In alternativa, se la coda esiste, il programma IVT non può aprire la coda poiché non è abilitata per l'inserimento e il richiamo dei messaggi. Verificare che la coda esista e che sia abilitata per l'inserimento e il richiamo dei messaggi.
- Il test potrebbe avere esito negativo con il seguente messaggio:

```
Unable to bind to object
```

  Questo messaggio indica che esiste una connessione con il server LDAP, ma che il server non è configurato correttamente. Il server LDAP non è configurato per memorizzare gli oggetti Java oppure le autorizzazioni sugli oggetti o il suffisso non sono corrette. Per ulteriore assistenza in questa situazione, consultare la documentazione per il proprio server LDAP.
- Il test potrebbe avere esito negativo con il seguente messaggio:

```
The security authentication was not valid that was supplied for
QueueManager 'qmgr' with connection mode 'Client' and host name 'hostname'
```

  Questo messaggio indica che il gestore code non è configurato correttamente per accettare una connessione client dal sistema. Vedi "Preparazione ed esecuzione dei programmi di esempio" a pagina 110 per i dettagli.

Il test di verifica dell'installazione di pubblicazione / sottoscrizione per WebSphere MQ classes per JMS

Un programma IVT (publish/subscribe installation verification test) viene fornito con WebSphere MQ classes per JMS. Il programma si connette a un gestore code in modalità bind o client, effettua la sottoscrizione a un argomento, pubblica un messaggio sull'argomento e riceve il messaggio appena pubblicato. Il programma può creare e configurare tutti gli oggetti richiesti in modo dinamico al runtime oppure può utilizzare JNDI per richiamare gli oggetti gestiti da un servizio di directory.

Eseguire il test di verifica dell'installazione senza utilizzare prima JNDI perché il test è autonomo e non richiede l'uso di un servizio directory. Per una descrizione degli oggetti gestiti, consultare "Tipi di oggetto JMS" a pagina 941.

Il test di verifica dell'installazione di pubblicazione / sottoscrizione senza utilizzare JNDI

In questo test, il programma IVT crea e configura tutti gli oggetti che richiede dinamicamente al runtime e non utilizza JNDI.

Viene fornito uno script per eseguire il programma IVT. Lo script è denominato PSIVTRun sui sistemi UNIX and Linux e PSIVTRun.bat su Windowse si trova nella sottodirectory `bin` della directory di installazione di WebSphere MQ classes per JMS.

Per eseguire il test in modalità bind, immettere il seguente comando:

```
PSIVTRun -nojndi [-m qmgr] [-bqm brokerQmgr] [-v providerVersion] [-t]
```

Per eseguire la verifica in modalit ... client, configurare prima il gestore code come descritto in “Preparazione ed esecuzione dei programmi di esempio” a pagina 110 , osservando che il canale da utilizzare assume il valore predefinito SYSTEM.DEF.SVRCONN, quindi immettere il seguente comando:

```
PSIVTRun -nojndi -client -m qmgr -host hostname [-port port] [-channel channel]
         [-bqm brokerQmgr] [-v providerVersion] [-ccsid ccsid] [-t]
```

I parametri sui comandi hanno i seguenti significati:

-m *gestore code*
Il nome del gestore code a cui si connette il programma IVT. Se si esegue la verifica in modalità bind e si omette questo parametro, il programma IVT si connette al gestore code predefinito.

-host *nome host*
Il nome host o l'indirizzo IP del sistema su cui è in esecuzione il gestore code.

-port *porta*
Il numero della porta su cui il listener del gestore code è in ascolto. Il valore predefinito è 1414.

-channel *canale*
Il nome del canale MQI che il programma IVT utilizza per connettersi al gestore code. Il valore predefinito è `SYSTEM.DEF.SVRCONN`.

-bqm *brokerQmgr*
Il nome del gestore code su cui è in esecuzione il broker. Il valore predefinito è il nome del gestore code a cui si connette il programma IVT.

Questo parametro è rilevante solo se il parametro -v specifica un numero di versione del gestore code inferiore a 7 e si sta utilizzando WebSphere Event Broker o WebSphere Message Broker come broker di pubblicazione / sottoscrizione.

-v *providerVersion*
Il livello di rilascio del gestore code a cui il programma IVT prevede di connettersi.

Questo parametro viene utilizzato per impostare la proprietà PROVIDERVERSION di un oggetto factory MQTopicConnectione ha gli stessi valori validi della proprietà PROVIDERVERSION. Per ulteriori informazioni su questo parametro, inclusi i relativi valori validi, consultare la descrizione della proprietà PROVIDERVERSION in Proprietà degli oggetti IBM WebSphere MQ classes for JMS .

Il valore predefinito è `unspecified`.

-ccsid *idccs*
L'identificativo (CCSID) della serie di caratteri codificati, o codepage, che deve essere utilizzata dal collegamento. Il valore predefinito è 819.

-t
La traccia è attivata. Per impostazione predefinita, la traccia è disattivata.

Un test riuscito produce un output simile al seguente output di esempio:

```
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
```

Il test di verifica dell'installazione di pubblicazione / sottoscrizione utilizzando JNDI

In questa verifica, il programma IVT utilizza JNDI per richiamare gli oggetti gestiti da un servizio directory.

Prima di poter eseguire il test, è necessario configurare un servizio di directory basato su un server LDAP (Lightweight Directory Access Protocol) o sul file system locale. È inoltre necessario configurare lo strumento di gestione WebSphere MQ JMS in modo che possa utilizzare il servizio directory per memorizzare gli oggetti gestiti. Per ulteriori informazioni su questi prerequisiti, consultare “Prerequisiti per le classi WebSphere MQ per JMS” a pagina 718. Per informazioni su come configurare lo strumento di gestione JMS WebSphere MQ , consultare “Configurazione dello strumento di amministrazione JMS” a pagina 937.

Il programma IVT deve essere in grado di utilizzare JNDI per richiamare un oggetto factory MQTopicConnectione un oggetto MQTopic dal servizio directory. Viene fornito uno script per creare questi oggetti gestiti. Lo script è denominato IVTSetup sui sistemi UNIX and Linux e IVTSetup.bat su Windowse si trova nella sottodirectory `bin` della directory di installazione di WebSphere MQ classes per JMS. Per eseguire lo script, immettere il comando seguente:

```
IVTSetup
```

Lo script richiama lo strumento di amministrazione JMS di WebSphere MQ per creare gli oggetti amministrati.

L'oggetto factory MQTopicConnectionè collegato con nome ivtTCF e viene creato con i valori predefiniti per tutte le relative proprietà, il che significa che il programma IVT viene eseguito in modalità bind, si connette al gestore code predefinito e utilizza la funzione di pubblicazione / sottoscrizione integrata. Se si desidera che il programma IVT venga eseguito in modalità client, connettersi a un gestore code diverso dal gestore code predefinito oppure utilizzare WebSphere Event Broker o WebSphere Message Broker invece della funzione di pubblicazione / sottoscrizione integrata, è necessario utilizzare lo strumento di amministrazione JMS di WebSphere MQ o WebSphere MQ Explorer per modificare le proprietà appropriate dell'oggetto factory MQTopicConnection. Per informazioni su come utilizzare lo strumento di gestione JMS WebSphere MQ , consultare “Utilizzo dello strumento di amministrazione JMS WebSphere MQ” a pagina 936. Per informazioni su come utilizzare WebSphere MQ Explorer, consultare la guida fornita con WebSphere MQ Explorer.

L'oggetto MQTopic è collegato con il nome ivtT ed è creato con i valori predefiniti per tutte le relative proprietà, ad eccezione della proprietà TOPIC, che ha il valore MQJMS/PSIVT/Information.

Una volta creati gli oggetti gestiti, è possibile eseguire il programma IVT. Per eseguire il test utilizzando JNDI, immettere il seguente comando:

```
PSIVTRun -url "providerURL" [-icf initCtxFact] [-t]
```

I parametri sul comando hanno i seguenti significati:

-url "*providerURL*"
L'URL (uniform resource locator) del servizio directory. L'URL può avere uno dei formati seguenti:

- `ldap://hostname/contextName`, per un servizio di directory basato su un server LDAP
- `file:/directoryPath`, per un servizio di directory basato sul file system locale

Notare che è necessario racchiudere l'URL tra virgolette (").

-icf *initCtxFact*
Il nome classe del factory di contesto iniziale, che deve essere uno dei seguenti valori:

- `com.sun.jndi.ldap.LdapCtxFactory`, per un servizio di directory basato sul server LDAP. Questo è il valore predefinito.
- `com.sun.jndi.fscontext.RefFSContextFactory`, per un servizio directory basato sul file system locale.

-t
La traccia è attivata. Per impostazione predefinita, la traccia è disattivata.

Un test riuscito produce un output simile a quello per un test riuscito senza utilizzare JNDI. La differenza principale è che l'output indica che la verifica sta utilizzando JNDI per richiamare un oggetto factory MQTopicConnectione un oggetto MQTopic.

Sebbene non sia strettamente necessario, si consiglia di riordinare dopo il test eliminando gli oggetti gestiti creati dallo script IVTSetup. Per questo scopo viene fornito uno script. Lo script è denominato IVTTidy sui sistemi UNIX and Linux e IVTTidy.bat su Windows, e si trova nella sottodirectory `bin` della directory di installazione di WebSphere MQ classes per JMS.

Determinazione dei problemi per il test di verifica dell'installazione di pubblicazione / sottoscrizione

Il test di verifica dell'installazione potrebbe avere esito negativo per i seguenti motivi:

- Se il programma IVT scrive un messaggio che indica che non è possibile trovare una classe, verificare che il percorso di classe sia impostato correttamente, come descritto in “Variabili di ambiente utilizzate dalle classi IBM WebSphere MQ per JMS” a pagina 724.
- Il test potrebbe avere esito negativo con il seguente messaggio:

```
Failed to connect to queue manager 'qmgr' with
connection mode 'connMode' and host name 'hostname'
```

e un codice di errore associato 2059. Le variabili nel messaggio hanno i seguenti significati:

qmgr
Il nome del gestore code a cui il programma IVT sta tentando di connettersi. Questo inserimento del messaggio è vuoto se il programma IVT sta tentando di connettersi al gestore code predefinito in modalità bind.

connMode
La modalità di connessione, che è `Bindings` o `Client`.

nomehost
Il nome host o l'indirizzo IP del sistema su cui è in esecuzione il gestore code.

Questo messaggio indica che il gestore code a cui il programma IVT sta tentando di connettersi non è disponibile. Verificare che il gestore code sia in esecuzione e, se il programma IVT sta tentando di connettersi al gestore code predefinito, accertarsi che il gestore code sia definito come gestore code predefinito per il sistema.

- Il test potrebbe avere esito negativo con il seguente messaggio:

```
Unable to bind to object
```

Questo messaggio indica che esiste una connessione con il server LDAP, ma che il server non è configurato correttamente. Il server LDAP non è configurato per memorizzare gli oggetti Java oppure le autorizzazioni sugli oggetti o il suffisso non sono corrette. Per ulteriore assistenza in questa situazione, consultare la documentazione per il proprio server LDAP.

- Il test potrebbe avere esito negativo con il seguente messaggio:

```
The security authentication was not valid that was supplied for
QueueManager 'qmgr' with connection mode 'Client' and host name 'hostname'
```

Questo messaggio indica che il gestore code non è configurato correttamente per accettare una connessione client dal sistema. Per ulteriori informazioni, consultare “Preparazione ed esecuzione dei programmi di esempio” a pagina 110.

Il programma di verifica dell'installazione per l'adattatore di risorse WebSphere MQ

Il programma IVT viene fornito come file EAR. Per utilizzare il programma, è necessario distribuirlo e definire alcuni oggetti come risorse JCA.

Il programma IVT (installation verification test) viene fornito come file EAR (enterprise archive) denominato wmq.jmsra.ivt.ear. Questo file viene installato con le classi WebSphere MQ per JMS nella stessa directory del file RAR dell'adattatore di risorse WebSphere MQ , wmq.jmsra.rar. Per informazioni su dove sono installati questi file, consultare “Installazione dell'adattatore di risorse WebSphere MQ” a pagina 735.

È necessario distribuire il programma IVT sul server delle applicazioni. Il programma IVT include un servlet e un MDB che verifica che un messaggio possa essere inviato e ricevuto da una coda WebSphere MQ . Facoltativamente, è possibile utilizzare il programma IVT per verificare che l'adattatore di risorse WebSphere MQ sia stato configurato correttamente per supportare le transazioni distribuite.

Prima di poter eseguire il programma IVT, è necessario definire un oggetto ConnectionFactory , un oggetto Queue e possibilmente un oggetto Activation Specification come risorse JCA e verificare che il server delle applicazioni crei gli oggetti JMS da queste definizioni e li collega in uno spazio dei nomi JNDI. È possibile scegliere le proprietà degli oggetti, ma la seguente serie di proprietà è un esempio semplice:

oggetto ConnectionFactory

```
channel:              SYSTEM.DEF.SVRCONN
hostName:             localhost
port:                 1414
queueManager:         ExampleQM
transportType:        CLIENT
```

Oggetto coda

```
baseQueueManagerName:  ExampleQM
baseQueueName:         TEST.QUEUE
```

Per impostazione predefinita, il programma IVT si aspetta che un oggetto ConnectionFactory sia collegato nello spazio nomi JNDI con il nome jms / ivt/IVTCF e un oggetto Queue con il nome jms / ivt/IVTQueue. È possibile utilizzare nomi diversi, ma in tal caso, è necessario immettere i nomi degli oggetti nella pagina iniziale del programma IVT e modificare il file EAR in modo appropriato.

Dopo aver distribuito il programma IVT e dopo che il server delle applicazioni ha creato gli oggetti JMS e li ha collegati allo spazio dei nomi JNDI, è possibile avviare il programma IVT immettendo un URL nel seguente formato nel browser Web:

```
http://app_server_host:port/WMQ_IVT/
```

dove *app_server_host* è l'indirizzo IP o il nome del sistema su cui è in esecuzione il server delle applicazioni e *port* è il numero della porta TCP su cui è in ascolto il server delle applicazioni. Di seguito è riportato un esempio:

```
http://localhost:9080/WMQ_IVT/
```

Figura 122 a pagina 784 mostra la pagina iniziale del programma IVT.

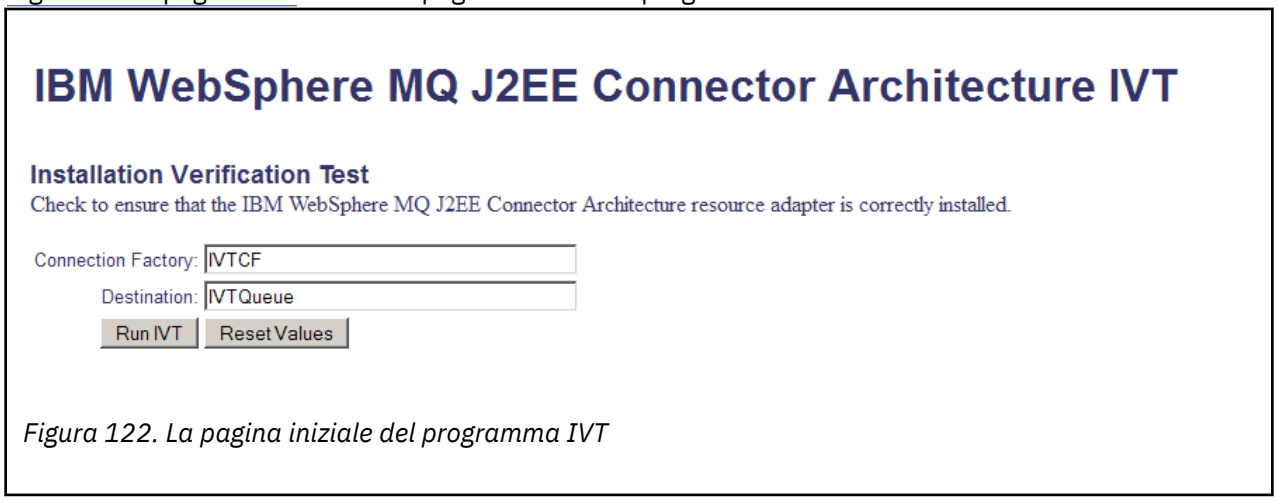


Figura 122. La pagina iniziale del programma IVT

Per eseguire il test, fare clic su **Esegui IVT**. Figura 123 a pagina 785 mostra la pagina visualizzata se l'IVT ha esito positivo.

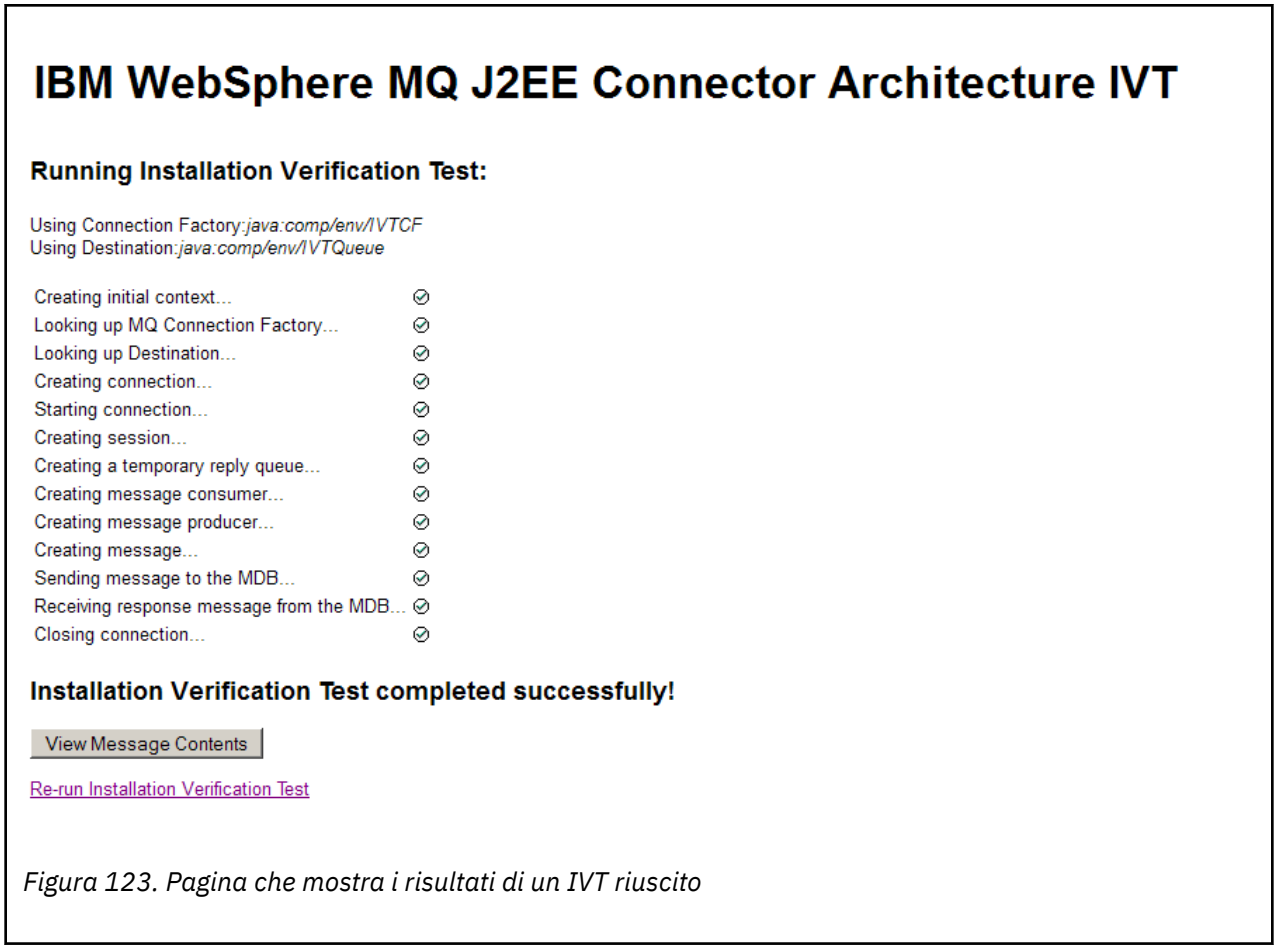


Figura 123. Pagina che mostra i risultati di un IVT riuscito

Se l'IVT non riesce, viene visualizzata una pagina simile a quella mostrata in Figura 124 a pagina 786 .
Per ulteriori informazioni sulla causa dell'errore, fare clic su **Visualizza traccia di stack**.

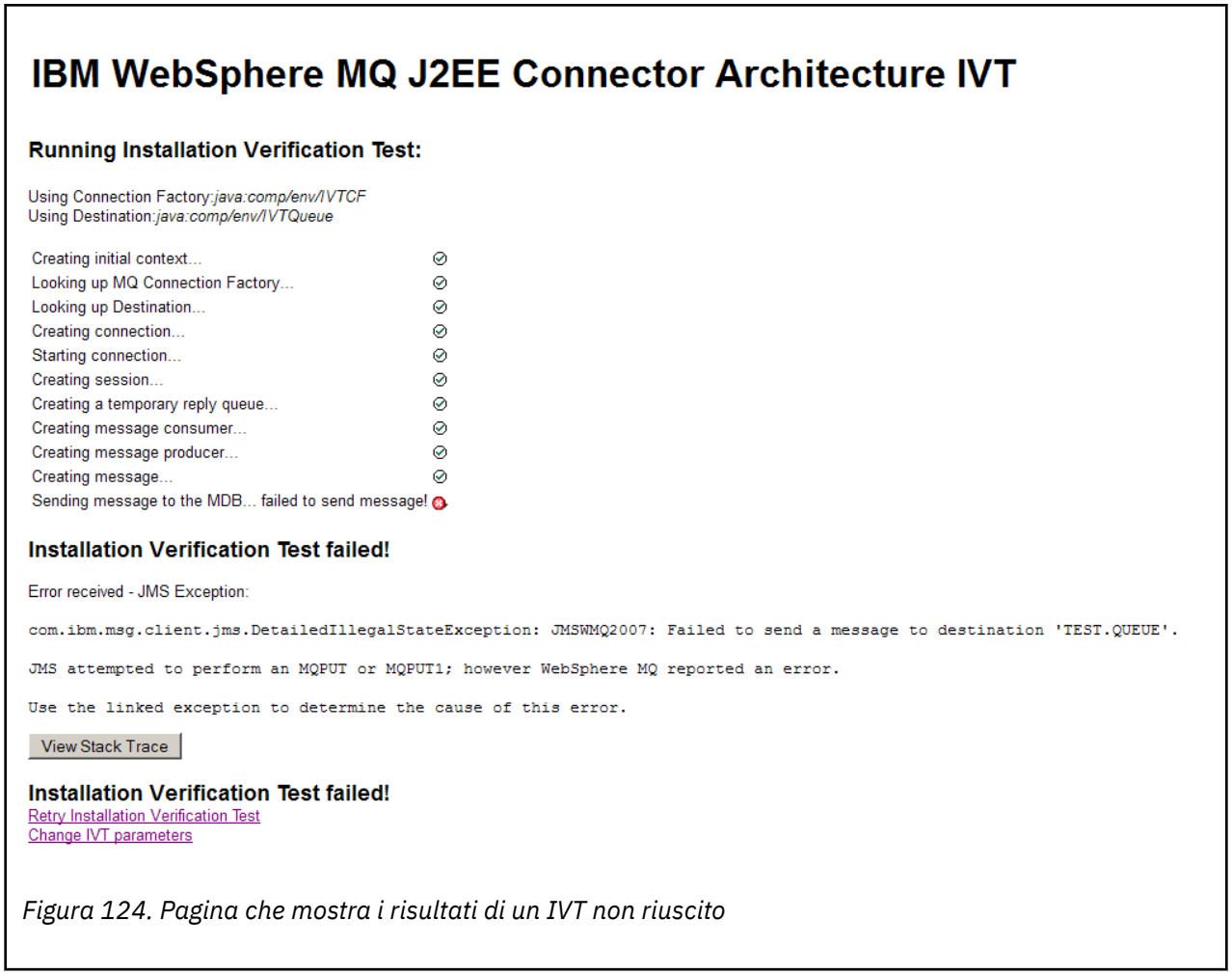


Figura 124. Pagina che mostra i risultati di un IVT non riuscito

Per istruzioni dettagliate e informazioni sugli script del programma di utilità forniti per distribuire l'applicazione IVT sui server delle applicazioni JBoss e WAS CE, consultare:

Attività correlate

“Installazione e verifica dell'adattatore di risorse MQ in WAS CE” a pagina 786
Installazione dell'adattatore di risorse IBM WebSphere MQ e esecuzione dell'applicazione IVT (Installation Verification Test) in WebSphere Application Server CE.

“Installazione e verifica dell'adattatore di risorse in JBoss AS 5.1 e 6” a pagina 789
Dopo aver installato l'adattatore di risorse IBM WebSphere MQ su JBoss AS 5.1 o 6, è possibile testare l'installazione dell'adattatore di risorse installando ed eseguendo l'applicazione IVT (Installation Verification Test).

Installazione e verifica dell'adattatore di risorse MQ in WAS CE

Installazione dell'adattatore di risorse IBM WebSphere MQ e esecuzione dell'applicazione IVT (Installation Verification Test) in WebSphere Application Server CE.

Prima di iniziare

Questa attività presuppone che si disponga di un server WebSphere Application Server CE in esecuzione e che si abbia familiarità con le attività di gestione standard per esso. Questa attività presuppone inoltre che si disponga di un'installazione IBM WebSphere MQ sul sistema locale e che si abbia familiarità con le attività di gestione standard.

Se si sta utilizzando l'adattatore di risorse per connettersi a un client IBM WebSphere MQ e si desidera eseguire transazioni XA distribuite, è necessario seguire le operazioni aggiuntive contrassegnate con **Solo XA client**.

1. Creare un gestore code denominato ExampleQMe configurarlo come descritto in "Preparazione ed esecuzione dei programmi di esempio" a pagina 110 notando che il listener deve essere avviato sulla porta 1414, il canale da utilizzare è denominato SYSTEM.DEF.SVRCONN e la coda utilizzata dall'applicazione IVT è denominata TEST.QUEUE. Anche alla coda modello SYSTEM.DEFAULT.MODEL.QUEUE dovrà essere concessa l'autorizzazione DSP e PUT in modo che questa applicazione possa creare una coda di risposta temporanea. Se si desidera utilizzare un gestore code differente, dettagli di connessione differenti o una coda diversa, consultare "Distribuzione dell'applicazione IVT su WAS CE con un ambiente MQ personalizzato" a pagina 788.
2. Ottenere il file dell'adattatore di risorse (wmq.jmsra.rar), l'applicazione IVT (wmq.jmsra.ivt.ear) e WAS_CE_jmsra_deployment_plan.xml e WAS_CE_jmsra_ivt_deployment_plan.xml deployment plan files. Per i dettagli sull'ubicazione di questi file, consultare "Installazione dell'adattatore di risorse WebSphere MQ" a pagina 735.

Per una descrizione dei collegamenti e delle connessioni in modalità client, consultare "Modalità di connessione per le classi WebSphere MQ per JMS" a pagina 774.

Se si desidera utilizzare una coda, un gestore code, una porta, un host, un canale o utilizzare la modalità di bind invece della modalità client, consultare "Distribuzione dell'applicazione IVT su WAS CE con un ambiente MQ personalizzato" a pagina 788.

Procedura

1. **Solo client XA**: modificare la copia del file WAS_CE_jmsra_deployment_plan.xml .
   a) Individuare la definizione della connessione jms / ivt/IVTCF e modificarla in modo che il factory di connessione sia abilitato XA - transaction.
      i) Commentare la sezione NonXA :

      ```
      <conn:xa-transaction>
      ```

      ii) Annullare il commento della sezione di configurazione XA:

      ```
      <conn:xa-transaction>
          <conn:transaction-caching/>
      </conn:xa-transaction>
      ```

   b) Salva le tue modifiche.
2. Opzionale: **Solo client XA**: modificare il descrittore di assemblaggio di MDB per richiedere le transazioni. Ciò forza l'MDB nell'IVT a partecipare a una transazione XA, anche se l'applicazione IVT funziona ancora senza questa modifica.
   a) Aprire il file wmq.jmsra.ivt.ear.
   b) Aprire il file WMQ_IVT_MDB.jar al suo interno.
   c) Modificare META-INF/ejb-jar.xml.
      i) Commentare o eliminare la riga all'interno del descrittore dell'assembly:

      ```
      <trans-attribute>NotSupported</trans-attribute>
      ```

      ii) Eliminare il commento dalla riga all'interno del descrittore dell'assembly:

      ```
      <trans-attribute>Required</trans-attribute>
      ```

      iii) Salvare le modifiche e aggiornare il file nel file WMQ_IVT_MDB.jar .
      iv) Aggiornare il file wmq.jmsra.ivt.ear con il file WMQ_IVT_MDB.jar modificato.
3. Distribuire l'adattatore risorse al server utilizzando il file del piano di distribuzione modificato.

a) Per fare ciò sulla riga comandi, immettere il seguente comando WAS CE:

```
deploy -u system -p manager deploy wmq.jmsra.rar WAS_CE_jmsra_deployment_plan.xml
```

b) Utilizzando l'interfaccia di amministrazione Web, andare a **Applicazioni> Deployer**

i) Impostare l'archivio in modo che sia il file wmq.jmsra.rar .

ii) Impostare il piano sul file WAS_CE_jmsra_deployment_plan.xml .

iii) Verificare che sia selezionato 'Avvia applicazione dopo l'installazione'.

iv) Fai clic su **Installa**.

4. Distribuire l'applicazione IVT al server utilizzando il piano di distribuzione fornito.

a) Sulla riga comandi, questa operazione può essere effettuata utilizzando il seguente comando WAS CE:

```
deploy -u system -p manager deploy wmq.jmsra.ivt.ear WAS_CE_jmsra_ivt_deployment_plan.xml
```

b) Utilizzando l'interfaccia di amministrazione Web, andare a **Applicazioni> Deployer**

i) Impostare l' **Archivio** come file `wmq.jmsra.ivt.ear` .

ii) Impostare il **Piano** come file `WAS_CE_jmsra_ivt_deployment_plan.xml` .

iii) Verificare che sia selezionato **Avvia applicazione dopo l'installazione** .

iv) Fai clic su **Installa**.

5. Eseguire l'applicazione IVT. Per ulteriori dettagli, vedere “Il programma di verifica dell'installazione per l'adattatore di risorse WebSphere MQ” a pagina 783. Per WAS CE l'URL predefinito è `http://localhost:8080/WMQ_IVT/`.

Distribuzione dell'applicazione IVT su WAS CE con un ambiente MQ personalizzato

Se si desidera utilizzare una coda, un gestore code, una porta, un host, un canale o utilizzare la modalità di bind anziché la modalità client, è necessario modificare l'applicazione IVT e gli script associati in WebSphere Application Server CE prima di distribuire l'adattatore risorse o l'applicazione IVT.

Informazioni su questa attività

Se si desidera eseguire la distribuzione in una configurazione diversa da quella specificata in “Installazione e verifica dell'adattatore di risorse MQ in WAS CE” a pagina 786, ovvero, se si desidera utilizzare una coda, un gestore code, una porta, un host, un canale o utilizzare la modalità di bind invece della modalità client, effettuare le operazioni riportate di seguito prima di distribuire l'adattatore di risorse o l'applicazione IVT.

Procedura

1. Se si desidera specificare un gestore code e una coda differenti da utilizzare per l'applicazione IVT, impostare i valori per il gestore code e la coda in `WAS_CE_jmsra_deployment_plan.xml`, per i dettagli, fare riferimento a “Impostazione dei valori per il gestore code e la coda” a pagina 789.
2. Se si desidera specificare un gestore code e una coda differenti nella configurazione per l'MDB (message - driven bean), impostare i valori per il gestore code e la coda utilizzati in `WAS_CE_jmsra_ivt_deployment_plan.xml`, per i dettagli, consultare “Impostazione dei valori per la configurazione MDB” a pagina 789.
3. Se si sta configurando l'adattatore di risorse per la connessione a IBM WebSphere MQ in modalità bind, assicurarsi che le librerie JNI si trovino nel percorso di sistema o nel percorso per WAS CE. Per ulteriori informazioni, consultare “Installazione e verifica dell'adattatore di risorse MQ in WAS CE” a pagina 786.
4. Se l'adattatore di risorse è già stato distribuito, è possibile ridistribuirlo con il piano di distribuzione modificato per modificare le impostazioni utilizzando il seguente comando:

```
deploy --user system --password manager redeploy wmq.jmsra.rar
WAS_CE_jmsra_deployment_plan.xml
```

Operazioni successive

Continuare la distribuzione dell'adattatore di risorse come descritto in “Installazione e verifica dell'adattatore di risorse MQ in WAS CE” a pagina 786.

Impostazione dei valori per il gestore code e la coda

Spiega come impostare i valori per il gestore code e la coda utilizzati in `WAS_CE_jmsra_deployment_plan.xml`.

Procedura

In `WAS_CE_jmsra_deployment_plan.xml`, impostare i valori per il gestore code e la coda che si sta utilizzando per l'applicazione IVT.

Per la definizione della connessione jms / ivt/IVTCF:

1. Impostare il valore dell'elemento queueManager in modo che sia il nome del gestore code.
2. Se si utilizza una connessione client, impostare il valore dei vari elementi di connessione client in modo che sia appropriato per una connessione al gestore code.
3. Se si sta utilizzando una connessione di bind:
   a. Impostare il valore dell'elemento transportType su BINDINGS.
   b. Commentare o eliminare i vari elementi di connessione client.
4. Per la destinazione del messaggio jms / ivt/IVTQueue, impostare il valore dell'elemento Nome baseQueuein modo che sia il nome della coda creata per l'applicazione IVT
5. Salva le tue modifiche.

Impostazione dei valori per la configurazione MDB

Spiega come impostare i valori per la configurazione MDB in `WAS_CE_jmsra_deployment_plan.xml`.

Procedura

In `WAS_CE_jmsra_ivt_deployment_plan.xml` , impostare i valori per il gestore code e la coda utilizzati nella configurazione per MDB.

Per il bean basato sui messaggi WMQ_IVT_MDB:

1. Impostare il valore dell'elemento queueManager in modo che sia il nome del gestore code.
2. Se si utilizza una connessione client, impostare il valore dei vari elementi di connessione client in modo che sia appropriato per una connessione al gestore code.
3. Se si sta utilizzando una connessione di bind:
   a. Impostare il valore dell'elemento transportType su BINDINGS.
   b. Commentare o eliminare i vari elementi di connessione client.
4. Salva le tue modifiche.

Installazione e verifica dell'adattatore di risorse in JBoss AS 5.1 e 6

Dopo aver installato l'adattatore di risorse IBM WebSphere MQ su JBoss AS 5.1 o 6, è possibile testare l'installazione dell'adattatore di risorse installando ed eseguendo l'applicazione IVT (Installation Verification Test).

Prima di iniziare

Importante: Queste istruzioni sono per JBoss AS 5.1 e 6, non sono valide per JBoss AS 7.

Per informazioni sull'installazione dell'adattatore di risorse in JBoss EAP 6.3, consultare “Installazione e verifica dell'adattatore di risorse in JBoss EAP 6.3” a pagina 792.

Questa attività presuppone che l'utente disponga di un server JBoss in esecuzione e che abbia familiarità con le attività di gestione standard. Questa attività presuppone inoltre che si disponga di un'installazione IBM WebSphere MQ sul sistema locale e che si abbia familiarità con le attività di gestione standard.

Se si sta utilizzando l'adattatore di risorse per connettersi a un client IBM WebSphere MQ e si deve eseguire transazioni XA distribuite, è necessario seguire i passaggi aggiuntivi contrassegnati come **Solo XA client**. Per una descrizione dei collegamenti e delle connessioni in modalità client, consultare “Modalità di connessione per le classi WebSphere MQ per JMS” a pagina 774.

Procedura

1. Creare un gestore code denominato ExampleQMe configurarlo come descritto in “Preparazione ed esecuzione dei programmi di esempio” a pagina 110.

   Quando si configura il gestore code, tenere presente quanto segue:

   - Il listener deve essere avviato sulla porta 1414.
   - Il canale da utilizzare è denominato SYSTEM.DEF.SVRCONN.
   - La coda utilizzata dall'applicazione IVT è denominata TEST.QUEUE.

   La coda modello SYSTEM.DEFAULT.MODEL.QUEUE deve essere concessa l'autorizzazione DSP e PUT in modo che questa applicazione possa creare una coda di risposta temporanea.

   Se si desidera utilizzare un gestore code differente, dettagli di connessione differenti o una coda diversa, consultare “Distribuzione dell'applicazione IVT su WAS CE con un ambiente MQ personalizzato” a pagina 788.
2. Ottenere il file dell'adattatore risorse (`wmq.jmsra.rar`), l'applicazione IVT (`wmq.jmsra.ivt.ear`) e il file `jboss-jmsra-ds.xml` .

   Per l'ubicazione di questi file, consultare “Installazione dell'adattatore di risorse WebSphere MQ” a pagina 735.
3. **Solo client XA**: modificare il file `jboss-jmsra-ds.xml` per abilitare le transazioni XA sul factory di connessione.
   a) Commentare o eliminare la riga all'interno della definizione del factory di connessione `<local-transaction/>`.
   b) Eliminare il commento dalla riga all'interno della definizione di factory di connessione `<xa-transaction/>`.
   c) Salva le tue modifiche.
4. **Solo client XA**: (facoltativo) modificare il descrittore di assemblaggio di MDB per richiedere le transazioni. Ciò forza l'MDB nell'IVT a partecipare a una transazione XA, anche se l'applicazione IVT funziona ancora senza questa modifica.
   a) Aprire il file `wmq.jmsra.ivt.ear`.
   b) Aprire il file `WMQ_IVT_MDB.jar` al suo interno.
   c) Modificare `META-INF/ejb-jar.xml`:
      i) Commentare o eliminare la riga all'interno del descrittore dell'assembly:

      ```
      <trans-attribute>NotSupported</trans-attribute>
      ```

      ii) Eliminare il commento dalla riga all'interno del descrittore dell'assembly:

      ```
      <trans-attribute>Required</trans-attribute>
      ```

      iii) Salvare le modifiche e aggiornare il file nel file `WMQ_IVT_MDB.jar` .
      iv) Aggiornare il file `wmq.jmsra.ivt.ear` con il `WMQ_IVT_MDB.jar`modificato.

5. Distribuire l'adattatore risorse al server copiando il file `wmq.jmsra.rar` nella directory `jboss/server/default/deploy`.
6. Creare le risorse JMS richieste per l'applicazione IVT copiando il file `jboss-jmsra-ds.xml` nella directory `jboss/server/default/deploy`.
7. Distribuire l'applicazione IVT copiando il file `wmq.jmsra.ivt.ear` nella directory `jboss/server/default/deploy`.
8. Eseguire l'applicazione IVT. Per ulteriori dettagli, vedere “Il programma di verifica dell'installazione per l'adattatore di risorse WebSphere MQ” a pagina 783. Per JBoss, l'URL predefinito è `http://localhost:8080/WMQ_IVT/`.

Distribuzione dell'applicazione IVT in JBoss con un ambiente IBM WebSphere MQ personalizzato

Quando si installa l'adattatore di risorse IBM WebSphere MQ in JBoss, se si desidera utilizzare una coda, un gestore code, una porta, un host, un canale o utilizzare la modalità di bind invece della modalità client, è necessario prima modificare l'applicazione IVT e gli script associati in JBoss prima di distribuire l'adattatore di risorse o l'applicazione IVT.

Informazioni su questa attività

Importante: Queste istruzioni sono applicabili solo per Java EE Versioni 6 e 5, non per Java EE Versione 7. l'utilizzo di queste istruzioni per JBoss Versione 8 (WildFly) non è pertanto supportato.

Se si desidera distribuire a una configurazione diversa da quella specificata in “Installazione e verifica dell'adattatore di risorse in JBoss AS 5.1 e 6” a pagina 789, ovvero, se si desidera utilizzare un gestore code, una coda, una porta, un host, un canale differenti o utilizzare la modalità bind invece della modalità client, completare la seguente procedura prima di distribuire l'adattatore risorse o l'applicazione IVT.

Procedura

1. Se si desidera specificare un gestore code e una coda diversi da utilizzare per l'applicazione IVT, impostare i valori per il gestore code e la coda.
   a) Per la definizione della connessione jms / ivt/IVTCF:
      i) Impostare il valore di queueManager config - property in modo che sia il nome del gestore code.
      ii) Se si utilizza una connessione client, impostare il valore dei vari elementi di connessione client in modo che sia appropriato per una connessione al gestore code.
      iii) Se si utilizza una connessione di bind, impostare il valore dell'elemento transportType su BINDINGS, quindi impostare come commento o eliminare i vari elementi di connessione client.
   b) Per l'mbean jms / ivt/IVTQ, impostare il valore dell'elemento Nome baseQueuein modo che sia il nome della coda creata per l'applicazione IVT.
   c) Salva le tue modifiche.
2. Se si desidera specificare un gestore code e una coda differenti nella configurazione per l'MDB (message - driven bean), modificare la configurazione dell'MDB per connettersi al gestore code e alla coda.
   a) Aprire il file `wmq.jmsra.ivt.ear`.
   b) Aprire il `WMQ_IVT_MDB.jar` al suo interno.
   c) Modificare `META-INF/ejb-jar.xml`:
      i) Impostare il valore di queueManager activation - config - property in modo che sia il nome del gestore code.
      ii) Se si utilizza una connessione client, impostare il valore delle varie proprietà di configurazione di attivazione della connessione client in modo che sia appropriato per una connessione al proprio gestore code.

iii) Se si utilizza una connessione di bind, impostare il valore di transportType activation - config - property su BINDINGS, quindi impostare come commento o eliminare i vari elementi della connessione client.

d) Salvare le modifiche e aggiornare il file all'interno del file `WMQ_IVT_MDB.jar` .

e) Aggiornare il file `wmq.jmsra.ivt.ear` con il `WMQ_IVT_MDB.jar`modificato.

3. Se si sta configurando l'adattatore di risorse per connettersi a IBM WebSphere MQ in modalità bind, assicurarsi che le librerie JNI si trovino nel percorso di sistema o nel percorso per JBoss. Per dettagli, consultare "Configurazione delle librerie JNI (Java Native Interface)" a pagina 726.

Operazioni successive

Continuare la distribuzione dell'adattatore di risorse come descritto in "Installazione e verifica dell'adattatore di risorse in JBoss AS 5.1 e 6" a pagina 789.

.

Installazione e verifica dell'adattatore di risorse in JBoss EAP 6.3

Dopo aver installato l'adattatore di risorse IBM WebSphere MQ in JBoss Enterprise Application Platform (EAP) 6.3, su un server autonomo o su un server in esecuzione in un dominio gestito, è possibile verificare l'installazione dell'adattatore di risorse installando ed eseguendo l'applicazione IVT (Installation Verification Test).

Informazioni su questa attività

Importante: Queste istruzioni sono solo per JBoss EAP 6.3 . Per informazioni sull'installazione dell'adattatore di risorse in JBoss AS 5.1 e 6, consultare "Installazione e verifica dell'adattatore di risorse in JBoss AS 5.1 e 6" a pagina 789.

Questa attività presuppone che l'utente disponga di un server JBoss in esecuzione e che abbia familiarità con le attività di gestione standard. Questa attività presuppone inoltre che si abbia un'installazione di IBM WebSphere MQ sul proprio sistema locale e che si abbia familiarità con la gestione standard.

Procedura

1. Creare un gestore code denominato ExampleQMe configurarlo come descritto in "Preparazione ed esecuzione dei programmi di esempio" a pagina 110.

   Quando si configura il gestore code, tenere presente quanto segue:

   - Il listener deve essere avviato sulla porta 1414.
   - Il canale da utilizzare è denominato SYSTEM.DEF.SVRCONN.
   - La coda utilizzata dall'applicazione IVT è denominata TEST.QUEUE.

   La coda modello SYSTEM.DEFAULT.MODEL.QUEUE deve essere concessa l'autorizzazione DSP e PUT in modo che questa applicazione possa creare una coda di risposta temporanea.

   Se si desidera utilizzare un gestore code differente, dettagli di connessione differenti o una coda diversa, consultare "Distribuzione dell'applicazione IVT su WAS CE con un ambiente MQ personalizzato" a pagina 788.
2. Ottenere il file dell'adattatore risorse (`wmq.jmsra.rar`) e l'applicazione IVT (`wmq.jmsra.ivt.ear`).

   Per l'ubicazione di questi file, consultare "Installazione dell'adattatore di risorse WebSphere MQ" a pagina 735.
3. Installare l'adattatore di risorse e verificare l'installazione eseguendo l'applicazione IVT (Installation Verification Test):

   - Se si sta installando l'adattatore di risorse su un server autonomo, consultare "Installazione e test su un server autonomo" a pagina 793.

- Se si sta installando l'adattatore di risorse su un server in esecuzione in un dominio gestito, consultare “Installazione e test su un server in esecuzione in un dominio gestito” a pagina 794.

Installazione e test su un server autonomo

Dopo aver installato l'adattatore di risorse IBM WebSphere MQ su JBoss EAP 6.3 su un server autonomo, è possibile verificare l'installazione dell'adattatore di risorse installando ed eseguendo l'applicazione IVT (Installation Verification Test).

Informazioni su questa attività

Le informazioni in questa attività sono per l'installazione e la verifica dell'adattatore di risorse su un server autonomo. Se si sta installando l'adattatore di risorse su un server in esecuzione in un dominio gestito, consultare “Installazione e test su un server in esecuzione in un dominio gestito” a pagina 794.

Importante: Queste istruzioni sono solo per JBoss EAP 6.3 . Per informazioni sull'installazione dell'adattatore di risorse in JBoss AS 5.1 e 6, consultare “Installazione e verifica dell'adattatore di risorse in JBoss AS 5.1 e 6” a pagina 789.

Procedura

1. Distribuire l'adattatore risorse al server copiando il file `wmq.jmsra.rar` nella directory <*EAP_HOME*>/`standalone/deployments`.
2. Creare le risorse JMS richieste per l'applicazione IVT aggiungendo le seguenti voci alla sezione `<resource-adapters>` del file <*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. Aggiungere le seguenti informazioni ai parametri di avvio del server delle applicazioni:

```
-Dcom.ibm.mq.connector.IVTMDBCFJNDIName=java:jboss/jms/ivt/IVTCF
```

4. Distribuire l'applicazione IVT copiando il file `wmq.jmsra.ivt.ear` nella directory *<EAP_HOME>*/`standalone/deployments`.
5. Avviare il server delle applicazioni.
6. Eseguire l'applicazione IVT.

   Per ulteriori informazioni, consultare “Il programma di verifica dell'installazione per l'adattatore di risorse WebSphere MQ” a pagina 783. Per JBoss, l'URL predefinito è `http://localhost:8080/WMQ_IVT/`.

   Nota: I nomi JNDI utilizzati per le risorse JMS richieste per l'esecuzione dell'applicazione IVT sono i seguenti:

```
Connection Factory : IVTCF
JNDI name          : java:jboss/jms/ivt/IVTCF

Destination        : IVTQueue
JNDI name          : java:jboss/jms/ivt/IVTQueue
```

   Quando si avvia l'applicazione IVT utilizzando l'URL specificato in precedenza, immettere i nomi JNDI di queste risorse nei rispettivi campi ed eseguire l'applicazione facendo clic su **Esegui IVT**.

Installazione e test su un server in esecuzione in un dominio gestito

Dopo aver installato l'adattatore di risorse IBM WebSphere MQ su JBoss EAP 6.3 su un server in esecuzione in un dominio gestito, è possibile verificare l'installazione dell'adattatore di risorse installando ed eseguendo l'applicazione IVT (Installation Verification Test).

Informazioni su questa attività

Le informazioni in questa attività sono per l'installazione e la verifica dell'adattatore risorse su un server in esecuzione in un dominio gestito. Se si sta installando l'adattatore di risorse su un server autonomo, consultare “Installazione e test su un server autonomo” a pagina 793.

Importante: Queste istruzioni sono solo per JBoss EAP 6.3 . Per informazioni sull'installazione dell'adattatore di risorse in JBoss AS 5.1 e 6, consultare “Installazione e verifica dell'adattatore di risorse in JBoss AS 5.1 e 6” a pagina 789.

Procedura

1. Distribuire l'adattatore risorse al server utilizzando la console di gestione JBoss o la CLI di gestione.
2. Creare le risorse JMS richieste per l'applicazione IVT aggiungendo le seguenti voci alla sezione `<resource-adapters>` di `<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. Aggiungere le seguenti informazioni ai parametri di avvio del server delle applicazioni:

```
-Dcom.ibm.mq.connector.IVTMDBCFJNDIName=java:jboss/jms/ivt/IVTCF
```

4. Arrestare e riavviare il server delle applicazioni.
5. Distribuire l'applicazione IVT utilizzando la console di gestione JBoss o la CLI di gestione.
6. Eseguire l'applicazione IVT.

   Per ulteriori informazioni, consultare “Il programma di verifica dell'installazione per l'adattatore di risorse WebSphere MQ” a pagina 783. Per JBoss, l'URL predefinito è `http://localhost:8080/WMQ_IVT/`.

   Nota: I nomi JNDI utilizzati per le risorse JMS richieste per l'esecuzione dell'applicazione IVT sono i seguenti:

```
Connection Factory : IVTCF
JNDI name          : java:jboss/jms/ivt/IVTCF

Destination        : IVTQueue
JNDI name          : java:jboss/jms/ivt/IVTQueue
```

   Immettere i nomi JNDI di queste risorse nei rispettivi campi quando si avvia l'applicazione IVT utilizzando l'URL specificato in precedenza, quindi eseguire l'applicazione facendo clic su **Esegui IVT**.

Script forniti con le classi WebSphere MQ per JMS

Viene fornito un certo numero di script per assistere le attività comuni che devono essere eseguite quando si utilizzano le classi WebSphere MQ per JMS.

Tabella 102 a pagina 795 elenca tutti gli script e i loro utilizzi. Gli script si trovano nella sottodirectory `bin` della directory di installazione di WebSphere MQ classes per JMS.

Tabella 102. Script forniti con le classi WebSphere MQ per JMS

Utilità	**Utilizza**
Ripulitura[1]	Questo script viene mantenuto per la compatibilit ... con le release precedenti ma non esegue alcuna funzione. La ripulitura manuale delle informazioni di sottoscrizione non è più necessaria
DefaultConfiguration	Esegue l'applicazione di configurazione predefinita su piattaforme diverse da Windows.
formatLog[1]	Questo script viene mantenuto per la compatibilit ... con le release precedenti ma non esegue alcuna funzione. L'output del log viene ora prodotto in testo leggibile.
IVTRUN[1] Configurazione IVT[1] IVTTidy[1]	Utilizzato nel test di verifica dell'installazione point-to-point, come descritto in “Il test di verifica dell'installazione point-to-point per WebSphere MQ classes per JMS” a pagina 776.

Tabella 102. Script forniti con le classi WebSphere MQ per JMS (Continua)	
Utilità	**Utilizza**
Admin JMS[1]	Esegue lo strumento di gestione JMS WebSphere MQ , come descritto in “Richiamo dello strumento di amministrazione IBM WebSphere MQ classes for JMS” a pagina 936.
JMSAdmin.config	Il file di configurazione per lo strumento di amministrazione JMS di WebSphere MQ , come descritto in “Configurazione dello strumento di amministrazione JMS” a pagina 937.
PSIVTRun[1]	Esegue il programma di verifica dell'installazione di pubblicazione / sottoscrizione, come descritto in “Il test di verifica dell'installazione di pubblicazione / sottoscrizione per WebSphere MQ classes per JMS” a pagina 779.
PSReportDump.class	Questa classe viene mantenuta per la compatibilità con le release precedenti, ma non esegue alcuna funzione.
setjmsenv	Imposta le variabili di ambiente per l'esecuzione di un'applicazione WebSphere MQ classes per JMS in una JVM (Java virtual machine) a 32 bit su sistemi UNIX and Linux , come descritto in “Variabili di ambiente utilizzate dalle classi IBM WebSphere MQ per JMS” a pagina 724.
setjmsenv64	Imposta le variabili di ambiente per l'esecuzione di un'applicazione WebSphere MQ classes per JMS in una JVM a 64 bit su sistemi UNIX and Linux , come descritto in “Variabili di ambiente utilizzate dalle classi IBM WebSphere MQ per JMS” a pagina 724.
Nota: 1. Su Windows, il nome file ha l'estensione .bat.	

Supporto per OSGi

OSGi fornisce un framework che supporta la distribuzione delle applicazioni come bundle. Nove bundle OSGi vengono forniti come parte di IBM WebSphere MQ classes for JMS .

OSGi fornisce un frameworkJava generico, sicuro e gestito, che supporta la distribuzione di applicazioni fornite sotto forma di bundle. I dispositivi conformi a OSGi possono scaricare e installare i bundle e rimuoverli quando non sono più necessari. Il framework gestisce l'installazione e l'aggiornamento dei bundle in modo dinamico e scalabile.

Il IBM WebSphere MQ classes for JMS. include i seguenti bundle OSGi.

com.ibm.msg.client.osgi.jms< numero versione> .jar
Il livello comune di codice in IBM WebSphere MQ classes for JMS. Per informazioni sull'architettura a livelli delle classi WebSphere MQ per JMS, vedere “Un'architettura a livelli” a pagina 800.

com.ibm.msg.client.osgi.jms.prereq_< numero versione> .jar
I file JAR (Java archive) prerequisiti per il livello comune.

com.ibm.msg.client.osgi.commonservices.j2se_<version numero> .jar
Servizi comuni per applicazioni Java Platform, Standard Edition (Java SE).

com.ibm.msg.client.osgi.nls_< numero versione> .jar
Messaggi per il livello comune.

com.ibm.msg.client.osgi.wmq_< numero versione> .jar
Il provider di messaggistica IBM WebSphere MQ in IBM WebSphere MQ classes for JMS.
Per informazioni sull'architettura a livelli di IBM WebSphere MQ classes for JMS , consultare “Un'architettura a livelli” a pagina 800.

com.ibm.msg.client.osgi.wmq.prereq_< numero versione> .jar
I file JAR prerequisiti per il provider di messaggistica IBM WebSphere MQ .

com.ibm.msg.client.osgi.wmq.nls_< numero versione> .jar
Messaggi per il provider di messaggistica IBM WebSphere MQ .

com.ibm.mq.osgi.directip_< numero versione> .jar
I file JAR per consentire al fornitore di messaggistica IBM WebSphere MQ di creare una connessione in tempo reale a un broker.

dove < numero di versione> è il numero di versione di WebSphere MQ che è stato installato.

I bundle vengono installati nella sottodirectory `java/lib/OSGi` dell'installazione di WebSphere MQ o nella cartella `java\lib\OSGi` in Windows.

Il bundle com.ibm.mq.osgi.java < numero versione> .jar, che è anche installato nella sottodirectory `java/lib/OSGi` dell'installazione di WebSphere MQ o nella cartella `java\lib\OSGi` su Windows, fa parte delle classi WebSphere MQ per Java. Questo bundle non deve essere caricato in un ambiente di runtime OSGi con le classi WebSphere MQ per JMS caricate.

I bundle OSGi per le classi WebSphere MQ per JMS sono stati scritti nella specifica OSGi Release 4. Non funzionano in un ambiente OSGi Release 3.

È necessario impostare correttamente il percorso di sistema o il percorso della libreria in modo che l'ambiente di runtime OSGi possa trovare i file DLL o le librerie condivise richiesti.

Se si utilizzano i bundle OSGi per IBM WebSphere MQ classes for JMS, gli argomenti temporanei non funzionano. Inoltre, le classi di uscita del canale scritte in Java non sono supportate a causa di un problema intrinseco nel caricamento delle classi in un ambiente con più programmi di caricamento classi come OSGi. Un bundle utente può essere a conoscenza delle classi IBM WebSphere MQ per i bundle JMS, ma i bundle IBM WebSphere MQ classes for JMS non sono a conoscenza di alcun bundle utente. Di conseguenza, il programma di caricamento classi utilizzato in un bundle IBM WebSphere MQ classes for JMS non può caricare una classe di uscita canale che si trova in un bundle utente.

Per ulteriori informazioni su OSGi, consultare il sito Web OSGi Alliance .

Risoluzione dei problemi con le classi IBM WebSphere MQ per JMS

È possibile esaminare i problemi eseguendo i programmi di verifica dell'installazione e utilizzando le funzionalità di traccia e di log.

Se un programma non viene completato correttamente, eseguire uno dei programmi di verifica dell'installazione, come descritto in “Il test di verifica dell'installazione point-to-point per WebSphere MQ classes per JMS” a pagina 776 e “Il test di verifica dell'installazione di pubblicazione / sottoscrizione per WebSphere MQ classes per JMS” a pagina 779, e seguire i consigli forniti nei messaggi di diagnostica.

Registrazione e IBM WebSphere MQ classes for JMS

Per default, l'output del log viene inviato al file `mqjms.log` . È possibile reindirizzarlo a un file o a una directory specifici.

La funzione di registrazione IBM WebSphere MQ classes for JMS viene fornita per segnalare problemi gravi, in particolare problemi che potrebbero indicare errori di configurazione piuttosto che errori di programmazione. Per impostazione predefinita, l'output del log viene inviato al file `mqjms.log` nella directory di lavoro JVM.

È possibile reindirizzare l'output del log a un altro file impostando la proprietà com.ibm.msg.client.commonservices.log.outputName. Il valore per questa proprietà può essere:

- Un nome di percorso singolo.
- Un elenco separato da virgole di nomi percorso (tutti i dati vengono registrati in tutti i file).

Ciascun nome percorso può essere:

- Assoluto o relativo.
- `stderr` o `System.err` per rappresentare il flusso di errori standard.

- `sttdout` o `System.out` per rappresentare il flusso di output standard.

Se il valore della proprietà identifica una directory, l'output del log viene scritto in `mqjms.log` in tale directory. Se il valore della proprietà identifica un file specifico, l'output del log viene scritto in tale file.

È possibile impostare questa proprietà nel file di configurazione IBM WebSphere MQ classes for JMS o come proprietà di sistema nel comando **java** . Nel seguente esempio, la proprietà è impostata come una proprietà di sistema e identifica un file specifico:

```
java -Djava.library.path=library_path
     -Dcom.ibm.msg.client.commonservices.log.outputName=/mydir/mylog.txt
      MyAppClass
```

Nel comando, *library_path* è il percorso della directory contenente le librerie IBM WebSphere MQ classes for JMS (consultare "Configurazione delle librerie JNI (Java Native Interface)" a pagina 726).

È possibile disabilitare l'output del log impostando la proprietà com.ibm.msg.client.commonservices.log.status su OFF. Il valore predefinito di questa proprietà è ON.

I valori `System.err` e `System.out` possono essere impostati per inviare output di log ai flussi `System.err` e `System.out` .

Introduzione alle classi WebSphere MQ per JMS, per i programmatori

WebSphere MQ classes for JMS è il fornitore JMS fornito con WebSphere MQ. WebSphere MQ classes for JMS implementa le interfacce definite nel pacchetto javax.jms e fornisce inoltre due serie di estensioni all'API JMS. Le applicazioni Java Platform, Standard Edition (Java SE) e Java Platform, Enterprise Edition (Java EE) possono utilizzare le classi WebSphere MQ per JMS.

La specificazione JMS definisce una serie di interfacce che le applicazioni possono utilizzare per eseguire operazioni di messaggistica. L'ultima versione della specifica è la versione 1.1. Il pacchetto javax.jms specifica i dettagli delle interfacce JMS e un provider JMS implementa tali interfacce per un prodotto di messaggistica specifico. WebSphere MQ classes per JMS è un provider JMS che implementa le interfaccia JMS per WebSphere MQ.

Il flusso di logica all'interno di un'applicazione JMS inizia con gli oggetti ConnectionFactory e Destination. L'applicazione utilizza un oggetto ConnectionFactory per creare un oggetto Connection, che rappresenta la connessione attiva dall'applicazione a un server di messaggistica. L'applicazione utilizza l'oggetto Connection per creare un oggetto Session, un contesto a thread singolo per la produzione e l'utilizzo di messaggi. L'applicazione può quindi utilizzare l'oggetto Session e un oggetto Destination per creare un oggetto MessageProducer , che l'applicazione utilizza per inviare messaggi alla destinazione specificata. La destinazione è una coda o un argomento nel sistema di messaggistica ed è incapsulata dall'oggetto Destinazione. L'applicazione può anche utilizzare l'oggetto Session e un oggetto Destination per creare un oggetto MessageConsumer , che l'applicazione utilizza per ricevere i messaggi inviati alla destinazione specificata.

La specifica JMS prevede che gli oggetti ConnectionFactory e Destination siano oggetti gestiti. Un amministratore crea e gestisce oggetti gestiti in un repository centrale e un'applicazione JMS richiama tali oggetti utilizzando JNDI (Java Naming and Directory Interface). Il repository di oggetti gestiti può variare da un file semplice a una directory LDAP (Lightweight Directory Access Protocol).

WebSphere MQ classes per JMS supporta l'utilizzo di oggetti amministrati. Un'applicazione può utilizzare tutte le funzioni di WebSphere MQ esposte tramite le classi WebSphere MQ per JMS senza avere alcuna informazione specifica di WebSphere MQcodificata all'interno dell'applicazione stessa. Questa disposizione fornisce all'applicazione un grado di indipendenza dalla configurazione WebSphere MQ sottostante. Per ottenere questa indipendenza, l'applicazione può utilizzare JNDI per richiamare factory di connessione e destinazioni memorizzate come oggetti amministrati e utilizzare solo le interfacce definite nel pacchetto javax.jms per eseguire operazioni di messaggistica. Un amministratore può utilizzare lo strumento di amministrazione WebSphere MQ JMS o IBM WebSphere MQ Explorer per creare e gestire gli oggetti gestiti in un repository centrale. Un server delle applicazioni, tuttavia, generalmente fornisce il proprio repository per gli oggetti gestiti e i propri strumenti per la creazione e la gestione degli oggetti. Un'applicazione Java EE può quindi utilizzare JNDI per richiamare gli oggetti gestiti dal repository del server delle applicazioni o da un repository centrale.

WebSphere MQ classes for JMS fornisce anche le estensioni all'API JMS. Le release precedenti delle classi WebSphere MQ per JMS contengono estensioni implementate negli oggetti MQConnectionFactory, MQQueue e MQTopic. Questi oggetti hanno proprietà e metodi specifici per WebSphere MQ. Gli oggetti possono essere gestiti oppure un'applicazione può creare gli oggetti in modo dinamico al runtime. Questa release di WebSphere MQ classes for JMS conserva queste estensioni ed è possibile continuare ad utilizzare, senza modifiche, tutte le applicazioni che utilizzano queste estensioni. Queste estensioni sono note come *estensioni JMS WebSphere MQ*. Si noti che, in questa serie di documentazione, gli oggetti creati dinamicamente da una applicazione al runtime *non* vengono considerati oggetti gestiti.

Oltre alle estensioni JMS di WebSphere MQ , questa release delle classi WebSphere MQ per JMS fornisce una serie più generica di estensioni all'API JMS. Queste estensioni sono note come *IBM JMS*e hanno i seguenti obiettivi generali:

- Per fornire un livello maggiore di coerenza tra i provider JMS IBM
- Per semplificare la scrittura di un'applicazione bridge tra due sistemi di messaggistica IBM
- Per semplificare la porta di un'applicazione da un fornitore JMS IBM a un altro

Il focus principale di queste estensioni riguarda la creazione e la configurazione di factory di connessione e destinazioni in modo dinamico al runtime, ma le estensioni forniscono anche funzioni che non sono direttamente correlate alla messaggistica, come la funzione per la determinazione dei problemi.

Un factory di connessione, una coda o un oggetto argomento creato utilizzando l'interfaccia javax.jms o una serie di estensioni JMS può essere indirizzato utilizzando una qualsiasi di queste API; ovvero, può essere assegnato a una qualsiasi delle interfacce. Per mantenere la portabilità dell'applicazione al livello più alto, utilizzare l'API più generica adatta ai propri requisiti.

Sia le applicazioni Java SE che Java EE possono utilizzare WebSphere MQ classes per JMS. Sulla piattaforma Java EE , WebSphere MQ classes for JMS supporta due tipi di comunicazione tra un componente di una applicazione e un gestore code WebSphere MQ :

Comunicazione in uscita
Utilizzando l'API di JMS direttamente, un componente dell'applicazione crea una connessione a un gestore code, quindi invia e riceve messaggi.

Ad esempio, il componente dell'applicazione pu essere un client dell'applicazione, un servlet, un JSP (JavaServer Page), un EJB (enterprise Java bean) o un MDB (message driven bean). In questo tipo di comunicazione, il contenitore del server delle applicazioni fornisce solo funzioni di basso livello a supporto delle operazioni di messaggistica, come il pool di connessioni e la gestione thread.

Comunicazione in ingresso
Un messaggio che arriva a una destinazione viene consegnato a un MDB, che lo elabora.

Le applicazioni Java EE utilizzano MDB per elaborare i messaggi in modo asincrono. Un MDB agisce come un listener di messaggi JMS ed è implementato da un metodo onMessage(), che definisce il modo in cui un messaggio viene elaborato. Un MDB viene distribuito nel contenitore EJB di un application server. Il modo preciso in cui un MDB è configurato dipende dal server delle applicazioni che si sta utilizzando, ma le informazioni di configurazione devono specificare a quali gestori code connettersi, come connettersi al gestore code, quale destinazione monitorare per i messaggi e il comportamento transazionale di MDB. Queste informazioni vengono quindi utilizzate dal contenitore EJB. Quando un messaggio che soddisfa i criteri di selezione dell'MDB arriva alla destinazione specificata, il contenitore EJB utilizza le classi WebSphere MQ per JMS per recuperare il messaggio dal gestore code e quindi consegna il messaggio all'MDB richiamandone il metodo onMessage().

Classi IBM WebSphere MQ per l'architettura JMS

IBM WebSphere MQ classes per JMS, come fornito in IBM WebSphere MQ Versione 7.0e release successivi, contiene una serie di miglioramenti rispetto alle release precedenti. Alcuni di questi miglioramenti sono il risultato di modifiche all'implementazione delle classi IBM WebSphere MQ per JMS, mentre altri sono il risultato delle classi IBM WebSphere MQ per JMS che sfruttano le modifiche alla funzione IBM WebSphere MQ sottostante.

Le seguenti sezioni riepilogano i miglioramenti chiave.

Un'architettura a livelli

Nelle release precedenti di WebSphere MQ, l'implementazione di WebSphere MQ classes per JMS è stata interamente specifica di WebSphere MQ. Anche altri prodotti IBM che forniscono sistemi di messaggistica hanno incluso provider JMS, ma questi provider JMS hanno molto poco o nulla in comune con l'implementazione delle classi WebSphere MQ per JMS.

Da WebSphere MQ V7.0, WebSphere MQ classes per JMS ha un'architettura a livelli. Il livello superiore del codice è un livello comune che può essere utilizzato da qualsiasi fornitore JMS IBM . Quando un'applicazione richiama un metodo JMS, qualsiasi elaborazione della chiamata che non è specifica di un sistema di messaggistica viene eseguita dal livello comune, che fornisce anche una risposta congruente alla chiamata. Qualsiasi elaborazione della chiamata specifica di un sistema di messaggistica viene delegata ad un livello inferiore. Figura 125 a pagina 800 mostra l'architettura a livelli.

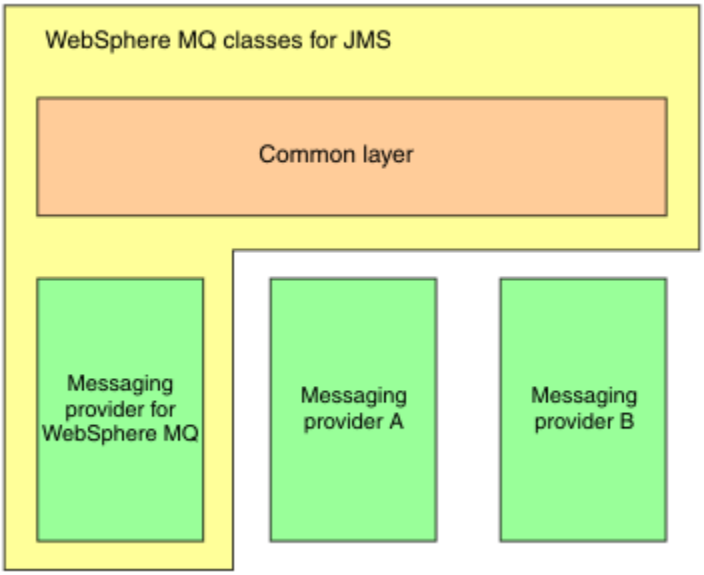


Figura 125. L'architettura a livelli per i provider JMS IBM

Il passaggio a un'architettura a livelli ha i seguenti obiettivi:

- Per migliorare la congruenza del comportamento dei vari provider JMS IBM
- Per semplificare la scrittura di un'applicazione bridge tra due sistemi di messaggistica IBM
- Per semplificare la porta di un'applicazione da un fornitore JMS IBM a un altro

Questa implementazione delle classi WebSphere MQ per JMS introduce anche una nuova serie di estensioni all'API JMS. Queste estensioni sono note come *estensioni JMS IBM*. Il focus principale di queste estensioni riguarda la creazione e la configurazione di factory di connessione e destinazioni in modo dinamico al runtime.

Un'applicazione che utilizza le estensioni JMS IBM inizia creando un JmsFactoryFactory, specificando come parametro una costante che identifica il sistema di messaggistica scelto. L'applicazione utilizza l'oggetto factory JmsFactoryper creare factory di connessione e destinazioni che dispongono delle classi specializzate corrette per il sistema di messaggistica scelto.

L'applicazione può quindi configurare le factory di connessione e le destinazioni impostando le loro proprietà. Le estensioni JMS di IBM forniscono una serie di metodi per impostare le proprietà. Questi metodi sono indipendenti da qualsiasi sistema di messaggistica. Ogni tipo di dati ha il suo metodo set e ogni proprietà è identificata da un nome, definito come membro finale statico della classe WMQConstants. Quando un'applicazione richiama un metodo, uno dei parametri sulla chiamata è il nome della proprietà e l'altro parametro è il valore della proprietà.

Ad esempio, se WebSphere MQ è il sistema di messaggistica, una delle proprietà di una factory di connessione è il nome del gestore code a cui connettersi. Utilizzando le estensioni JMS IBM , un'applicazione imposta il nome del gestore code su JUPITER richiamando il seguente metodo:

```
JmsConnectionFactory myCF;
...
myCF.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "JUPITER");
```

Al contrario, un'applicazione può eseguire la stessa funzione richiamando il seguente metodo:

```
MQConnectionFactory myCF;
...
myCF.setQueueManager("JUPITER");
```

Questo metodo è un'estensione WebSphere MQ JMS ed è specifico per WebSphere MQ come sistema di messaggistica. L'utilizzo di questo metodo rende quindi l'applicazione potenzialmente meno facile da trasferire ad un altro fornitore JMS IBM .

La relazione tra le classi WebSphere MQ per JMS e le classi WebSphere MQ per Java

Nelle release di WebSphere MQ, precedenti alla versione 7.0, WebSphere MQ classes per JMS è stato implementato quasi interamente come un livello di codice sulle classi WebSphere MQ per Java. Questa disposizione ha causato una certa confusione tra gli sviluppatori dell'applicazione poiché l'impostazione dei campi o dei metodi di chiamata nella classe MQEnvironment può causare effetti indesiderati e imprevisti sul comportamento di runtime del codice scritto utilizzando WebSphere MQ classes per JMS. Inoltre, l'implementazione delle classi WebSphere MQ per JMS aveva alcuni vincoli nelle aree in cui l'API JMS non è un adattamento naturale rispetto alle classi WebSphere MQ per Java, e questi vincoli hanno portato ad alcuni problemi relativi alle prestazioni di runtime.

Da WebSphere MQ V7.0, l'implementazione delle classi WebSphere MQ per JMS non dipende più dalle classi WebSphere MQ per Java. Le classi WebSphere MQ per Java e WebSphere MQ per JMS sono ora peer che utilizzano un'interfaccia Java comune per MQI. Questa disposizione consente un maggiore spazio per l'ottimizzazione delle prestazioni e significa che l'impostazione dei campi o dei metodi di chiamata nella classe MQEnvironment non ha alcun effetto sul comportamento di runtime del codice scritto utilizzando le classi WebSphere MQ per JMS. Figura 126 a pagina 801 mostra la relazione tra le classi WebSphere MQ per JMS e le classi WebSphere MQ per Java nelle release precedenti di WebSphere MQ e in WebSphere MQ V7.0 e release successive.

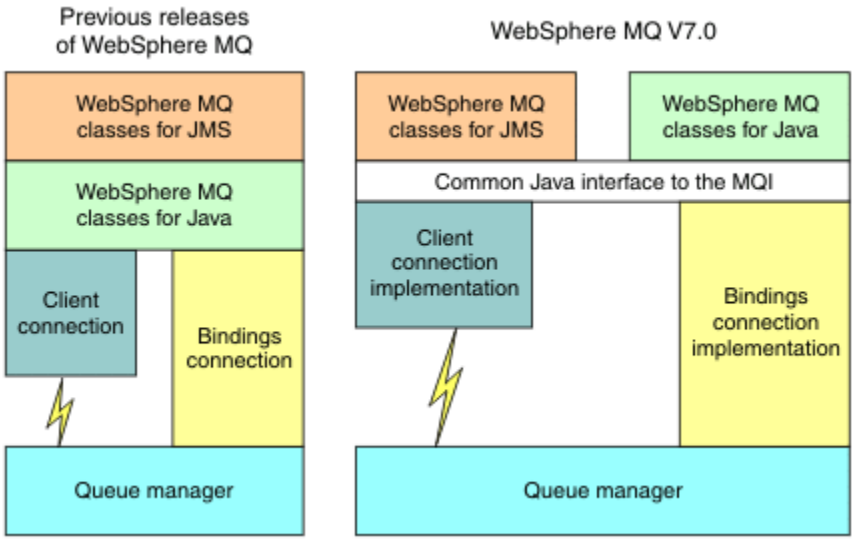


Figura 126. La relazione tra le classi WebSphere MQ per JMS e le classi WebSphere MQ per Java

Per mantenere la compatibilità con le release precedenti, le classi di uscita del canale scritte in Java possono ancora utilizzare le classi WebSphere MQ per le interfacce Java, anche se le classi di uscita del canale vengono richiamate dalle classi WebSphere MQ per JMS. Tuttavia, l'utilizzo delle classi WebSphere

MQ per le interfacce Java significa che le applicazioni dipendono ancora dalle classi WebSphere MQ per il file JAR Java, com.ibm.mq.jar. Se non si desidera com.ibm.mq.jar nel percorso classe, è possibile utilizzare la nuova serie di interfacce nel pacchetto com.ibm.mq.exits .

È ora possibile creare e configurare oggetti amministrati JMS con WebSphere MQ Explorer.

Pubblicazione/sottoscrizione della messaggistica

WebSphere MQ V7.0e release successive, contengono la funzione di pubblicazione / sottoscrizione integrata. Questa funzione sostituisce WebSphere MQ Publish / Subscribe, che è stato fornito con WebSphere MQ V6.0.

Le classi WebSphere MQ per le applicazioni JMS possono utilizzare la funzione di pubblicazione / sottoscrizione integrata e possono utilizzarla invece di utilizzare WebSphere Event Broker o WebSphere Message Broker per la messaggistica di pubblicazione / sottoscrizione con WebSphere MQ come trasporto. La configurazione delle classi WebSphere MQ per JMS per utilizzare la nuova funzione è più semplice rispetto alla configurazione delle classi WebSphere MQ per JMS per l'utilizzo di WebSphere MQ Publish / Subscribe, WebSphere Event Broker o WebSphere Message Broker. Gli amministratori e sviluppatori di applicazioni non devono più gestire code di pubblicazione, code di sottoscrittori, archivi di sottoscrizioni e ripulitura dei sottoscrittori. Inoltre, gli oggetti ConnectionFactory e Topic hanno un numero minore di proprietà.

La funzione di pubblicazione / sottoscrizione integrata fornisce anche alcune funzioni aggiuntive come le pubblicazioni conservate e una scelta di due schemi di caratteri jolly per la specifica di una serie di argomenti a cui un'applicazione desidera sottoscrivere.

Un'applicazione può ancora utilizzare una connessione in tempo reale a un broker di WebSphere Event Broker o WebSphere Message Broker per la messaggistica di pubblicazione / sottoscrizione. Questo riferimento non è modificato.

Le applicazioni che utilizzano WebSphere MQ Publish / Subscribe possono utilizzare la funzione di pubblicazione / sottoscrizione integrata senza modifiche quando il gestore code a cui sono connessi viene aggiornato. Le proprietà impostate da un'applicazione, ma non richieste dalla funzione di pubblicazione / sottoscrizione integrata, vengono ignorate.

Provider di messaggistica WebSphere MQ

Il provider di messaggistica WebSphere MQ dispone di due modalità operative:

- *WebSphere MQ modalità normale del provider di messaggistica*
- *WebSphere MQ*

La modalità normale del provider di messaggistica WebSphere MQ utilizza tutte le funzioni di WebSphere MQ Versione 7.0e i successivi gestori code delle release per implementare JMS. Questa modalità viene utilizzata solo per connettersi a un gestore code WebSphere MQ e può connettersi a WebSphere MQ Versione 7.0e ai gestori code delle release successive in modalità client o bind. Questa modalit ... Š ottimizzata per utilizzare la nuova WebSphere MQ Versione 7.0 e la successiva funzione di rilascio.

La modalità di migrazione del provider di messaggistica WebSphere MQ è basata su WebSphere MQ Versione 6.0 e utilizza solo funzioni disponibili in WebSphere MQ Versione 6.0 per implementare JMS. È possibile connettersi a WebSphere MQ Versione 7.0 e successivi gestori code della release utilizzando la modalità di migrazione del provider di messaggistica WebSphere MQ ma non è possibile utilizzare alcuna delle ottimizzazioni della Versione 7.0 . Questa modalità consente le connessioni a una delle seguenti versioni del gestore code:

1. WebSphere MQ Versione 7.0, e successive, gestore code in modalità bind o client, ma questa modalità utilizza solo le funzioni disponibili per un gestore code WebSphere MQ Versione 6.0
2. WebSphere MQ Versione 6.0 o precedente gestore code in modalità client

Se si desidera connettersi a WebSphere Event Broker o WebSphere Message Broker utilizzando WebSphere MQ Enterprise Transport, utilizzare la modalità di migrazione del provider di messaggistica WebSphere MQ . Se si utilizza WebSphere MQ Real-Time Transport, la modalità di migrazione del provider

di messaggistica WebSphere MQ viene selezionata automaticamente, poiché sono state esplicitamente selezionate le proprietà nell'oggetto factory di connessione. La connessione a WebSphere Event Broker o WebSphere Message Broker mediante WebSphere MQ Enterprise Transport segue le regole generali per la selezione della modalità descritte in Regole per la selezione della modalità del provider di messaggistica WebSphere MQ .

Utilizzo asincrono dei messaggi

WebSphere MQ V7.0 e qualsiasi release successivo supporta l'utilizzo asincrono dei messaggi. Un'applicazione può registrare una funzione di callback per una destinazione. Quando un messaggio appropriato viene inviato alla destinazione, WebSphere MQ richiama la funzione e trasmette il messaggio come parametro. La funzione elabora quindi il messaggio in modo asincrono. Nelle release precedenti di WebSphere MQ, questa funzione era disponibile solo quando si utilizzano le classi WebSphere MQ per JMS.

WebSphere MQ classes per JMS è stato modificato per sfruttare questa nuova funzione in WebSphere MQ V7.0 e in qualsiasi release successivo. L'implementazione dei listener di messaggi JMS è ora più semplice rispetto a WebSphere MQe WebSphere MQ classes per JMS non deve più eseguire il polling di una destinazione per controllare se un messaggio adatto è stato inviato alla destinazione. Le prestazioni dei listener di messaggi JMS vengono migliorate di conseguenza, in particolare quando un'applicazione utilizza più listener di messaggi in una sessione per monitorare più destinazioni. La velocità di trasmissione dei messaggi viene incrementata e il tempo impiegato per consegnare un messaggio ad un listener di messaggi dopo che è arrivato ad una destinazione viene ridotto.

Gli MDB (Message Driven Beans) hanno miglioramenti delle prestazioni simili. Inoltre, a causa di un altro miglioramento della funzione WebSphere MQ , più MDB che utilizzano messaggi dalla stessa destinazione ora riscontrano un conflitto ridotto sui messaggi.

Selezioni messaggi

Ad eccezione della selezione dei messaggi in base all'identificativo del messaggio o all'identificativo di correlazione, tutta la selezione dei messaggi nelle release di WebSphere MQ precedenti alla versione 7.0 è stata effettuata dalle classi WebSphere MQ per JMS. In WebSphere MQ V7.0, e in qualsiasi release successivo, tutta la scelta dei messaggi viene effettuata dal gestore code.

Di conseguenza, la velocità di trasmissione dei messaggi viene incrementata per le applicazioni che utilizzano i messaggi utilizzando la selezione dei messaggi. Il miglioramento delle prestazioni è maggiore per un'applicazione che si connette in modalità client in quanto solo i messaggi che soddisfano i criteri di selezione vengono trasportati sulla rete e WebSphere MQ classes for JMS gestisce solo i messaggi che distribuisce all'applicazione.

Condivisione di una connessione di comunicazione

Nei rilasci precedenti di WebSphere MQ, se un'applicazione client WebSphere MQ si connettesse a un gestore code più di una volta utilizzando lo stesso canale MQI, ogni istanza del canale MQI richiedeva una connessione TCP separata. In WebSphere MQ V7.0 e in qualsiasi release successivo, ogni connessione a un gestore code che utilizza lo stesso canale MQI può condividere una singola connessione TCP. Questa disposizione significa che sono richieste meno risorse di rete e che il tempo totale impiegato per creare più connessioni al gestore code è ridotto, in particolare quando si utilizza SSL perché l'handshake SSL si verifica solo una volta all'inizio della connessione TCP.

WebSphere MQ classes per JMS sfrutta questo miglioramento. Per un'applicazione che si connette a un gestore code in modalità client, WebSphere MQ classes per JMS potrebbe creare più di una connessione a un gestore code utilizzando il canale MQI con il nome specificato come proprietà dell'oggetto ConnectionFactory . Ognuna di queste connessioni al gestore code può ora condividere una singola connessione TCP.

Lettura anticipata sulle connessioni client

Se un'applicazione utilizza una connessione client per utilizzare messaggi non persistenti da una destinazione, la destinazione può essere configurata in modo che WebSphere MQ classes per JMS utilizzi un buffer per memorizzare i messaggi di interesse prima di consegnarli all'applicazione. Questa ottimizzazione è denominata *lettura anticipata* e può essere utilizzata dalle applicazioni che utilizzano i messaggi in modo sincrono richiamando il metodo receive () e dai listener di messaggi e dagli MDB, che utilizzano i messaggi in modo asincrono. La lettura anticipata è particolarmente efficace per le destinazioni con un numero elevato di messaggi che devono essere utilizzati rapidamente.

La lettura anticipata non è valida per i messaggi persistenti perché, se i messaggi persistenti fossero letti in un buffer, il gestore code non sarebbe più in grado di ripristinare i messaggi in seguito a un errore. Tuttavia, un'applicazione che utilizza i messaggi da una destinazione con una combinazione di messaggi persistenti e non persistenti può ancora utilizzare la lettura anticipata. L'ordine dei messaggi viene conservato, ma i vantaggi di runtime della lettura anticipata si applicano solo ai messaggi non persistenti.

Quando si decide se utilizzare la lettura anticipata, considerare i seguenti punti:

- Se un'applicazione utilizza i messaggi da una destinazione configurata per la lettura anticipata e l'applicazione termina per qualsiasi motivo, tutti i messaggi non persistenti attualmente memorizzati nel buffer vengono eliminati.
- Se tutte le seguenti condizioni sono vere, i messaggi inviati a una coda in una sessione potrebbero non essere ricevuti nell'ordine in cui sono stati inviati:
  - Un'applicazione utilizza due utenti di messaggi nella stessa sessione per utilizzare i messaggi dalla coda.
  - Ciascun utente del messaggio utilizza un oggetto Destinazione differente per la coda.
  - Uno o entrambi gli oggetti di destinazione sono configurati per la lettura anticipata.

invio di messaggi

Quando un'applicazione invia messaggi a una destinazione, la destinazione può essere configurata in modo che, quando l'applicazione richiama send (), WebSphere MQ classes for JMS inoltra il messaggio al gestore code e restituisce il controllo all'applicazione senza determinare se il gestore code ha ricevuto il messaggio in modo sicuro. Le classi WebSphere MQ per JMS possono funzionare in questo modo solo per i messaggi non persistenti e per i messaggi persistenti inviati in una sessione transata.

Per i messaggi persistenti inviati in una sessione sottoposta a transazione, l'applicazione determina se il gestore code ha ricevuto i messaggi in modo sicuro quando richiama commit (). Per i messaggi inviati in una sessione non sottoposta a transazione, la proprietà SENDCHECKCOUNT dell'oggetto ConnectionFactory specifica il numero di messaggi da inviare prima che WebSphere MQ classes for JMS verifichi che il gestore code abbia ricevuto i messaggi in modo sicuro.

Questa ottimizzazione è di maggior vantaggio per un'applicazione che si connette a un gestore code in modalità client e deve inviare una sequenza di messaggi in rapida successione, ma non richiede un feedback immediato dal gestore code per ogni messaggio inviato.

Uscite canale

Quando vengono richiamate da WebSphere MQ classes for JMS, i programmi di uscita del canale scritti in C o C++ ora si comportano come quando vengono richiamati da un client WebSphere MQ MQI. Le prestazioni delle classi di uscita del canale scritte in Java sono migliorate ed è ora possibile scrivere le classi di uscita del canale utilizzando una nuova serie di interfacce nel pacchetto com.ibm.mq.exits invece di utilizzare le interfacce nelle classi WebSphere MQ per Java.

Proprietà dei messaggi

Un messaggio JMS è costituito da una serie di campi di intestazione, una serie di proprietà e un corpo che contiene i dati dell'applicazione. Come minimo, un messaggio WebSphere MQ è costituito da un descrittore di messaggi e dai dati dell'applicazione.

Quando un'applicazione WebSphere MQ classes for JMS invia un messaggio JMS, WebSphere MQ classes for JMS associa il messaggio JMS in un messaggio WebSphere MQ . Alcuni dei campi e delle proprietà dell'intestazione JMS vengono mappati in campi nel descrittore del messaggio e altri in campi in un'intestazione WebSphere MQ aggiuntiva denominata intestazione MQRFH2 . Quando un'applicazione WebSphere MQ classes for JMS riceve un messaggio JMS, WebSphere MQ classes for JMS esegue la mappatura inversa.

Un'applicazione che utilizza MQI per ricevere i messaggi da un'applicazione WebSphere MQ per JMS deve quindi essere in grado di gestire un'intestazione MQRFH2 . Se l'applicazione non riesce a gestire un'intestazione MQRFH2 , la proprietà TARGCLIENT dell'oggetto di destinazione può essere impostata per indicare a WebSphere MQ classes for JMS di non includere un'intestazione MQRFH2 nei messaggi WebSphere MQ . Tuttavia, escludendo l'intestazione MQRFH2 , le informazioni contenute in alcuni campi e proprietà dell'intestazione JMS vengono perse.

Allo stesso modo, un'applicazione che utilizza MQI per inviare messaggi a un'applicazione WebSphere MQ per JMS deve includere un'intestazione MQRFH2 in ogni messaggio. Se un'intestazione MQRFH2 non è inclusa, WebSphere MQ classes per JMS può impostare solo quei campi di intestazione JMS e le proprietà che possono essere derivate dai campi in un descrittore di messaggi.

WebSphere MQ V7.0 fornisce un ulteriore supporto per le applicazioni che utilizzano MQI per ricevere e inviare messaggi dalle classi WebSphere MQ per le applicazioni JMS.

Quando un'applicazione richiama MQGET per ricevere un messaggio da un'applicazione WebSphere MQ per JMS, l'applicazione può decidere di riceverlo in uno dei modi seguenti:

1. Il messaggio viene consegnato con un descrittore del messaggio, un'intestazione MQRFH2 che contiene i dati derivati dai campi e dalle proprietà dell'intestazione JMS e i dati dell'applicazione.
2. Il messaggio viene consegnato con un descrizione del messaggio, i dati dell'applicazione e una serie di proprietà del messaggio.

Nell'opzione 2, ogni proprietà del messaggio rappresenta un campo di intestazione JMS o una proprietà che è stata originariamente mappata dalle classi WebSphere MQ per JMS in un campo in un'intestazione MQRFH2 . Dopo la chiamata MQGET, l'applicazione può utilizzare la chiamata MQINQMP per ottenere i valori delle proprietà del messaggio. L'utilizzo dell'opzione 2 invece dell'opzione 1 per ricevere un messaggio semplifica la logica dell'applicazione nei modi seguenti:

- L'applicazione non deve analizzare la parte variabile dell'intestazione MQRFH2 , che contiene il campo di intestazione JMS e i dati della proprietà codificati in formato simile a XML.
- L'applicazione non deve convertire i dati carattere nella porzione variabile dell'intestazione MQRFH2 .

Di conseguenza, prima che un'applicazione chiami MQPUT per l'invio di un messaggio a un'applicazione WebSphere MQ per JMS, l'applicazione può utilizzare la chiamata MQSETMP per impostare i valori delle proprietà del messaggio invece di creare un'intestazione MQRFH2 .

Funzionalità

WebSphere MQ classes per JMS contiene una serie di miglioramenti relativi alla praticità:

- Traccia.

  WebSphere MQ classes per JMS contiene una classe che un'applicazione può utilizzare per controllare la traccia. Un'applicazione può avviare e arrestare la traccia, specificare il livello di dettaglio richiesto in una traccia e personalizzare l'emissione della traccia in vari modi.
- Registrazione.

  WebSphere MQ classes for JMS conserva un file di log che contiene i messaggi relativi agli errori che è necessario correggere. I messaggi sono scritti in testo semplice. WebSphere MQ classes per JMS

contiene una classe che un'applicazione può utilizzare per specificare l'ubicazione del file di log e la sua dimensione massima.

- FFST (First Failure Support Technology) (FFST).

  Se si verifica un errore grave, WebSphere MQ classes for JMS genera un report FFST in un file FDC. Il report FFST contiene informazioni che IBM Service può utilizzare per diagnosticare il problema più rapidamente.

- Informazioni sulla versione.

  WebSphere MQ classes for JMS contiene una classe che un'applicazione può utilizzare per interrogare la versione delle classi WebSphere MQ per JMS.

- Messaggi di eccezione.

  I messaggi di eccezione sono stati migliorati per fornire ulteriori informazioni sulle cause degli errori e sulle azioni richieste per correggere gli errori.

- Server delle applicazioni.

  L'integrazione delle funzioni di funzionalità di WebSphere MQ classes for JMS con quelle di WebSphere Application Server è stata migliorata.

MQC viene sostituito da MQConstants

Un nuovo package, com.ibm.mq.constants, viene fornito con IBM WebSphere MQ Versione 7.0. Questo pacchetto contiene la classe MQConstants, che implementa un numero di interfacce. MQConstants contiene le definizioni di tutte le costanti presenti nell'interfaccia MQC e un certo numero di nuove costanti. Le interfacce in questo pacchetto seguono da vicino i nomi dei file di intestazione delle costanti utilizzati in IBM WebSphere MQ.

Ad esempio, l'interfaccia CMQC contiene una costante MQOO_INPUT_SHARED; questa interfaccia e costante corrispondono al file di intestazione `cmqc.h` e alla costante MQOO_INPUT_SHARED.

com.ibm.mq.constants può essere utilizzato con entrambe le classi IBM WebSphere MQ per Java e IBM WebSphere MQ per JMS.

MQC è ancora presente e ha le costanti che aveva precedentemente. Tuttavia, per tutte le nuove applicazioni, è necessario utilizzare il package com.ibm.mq.constants .

Scrittura delle classi WebSphere MQ per le applicazioni JMS

Dopo una breve introduzione al modello JMS, questo argomento fornisce una guida dettagliata su come scrivere le classi WebSphere MQ per applicazioni JMS.

Modello JMS

Il modello JMS definisce una serie di interfacce che le applicazioni Java possono utilizzare per eseguire operazioni di messaggistica. Le classi WebSphere MQ per JMS, come provider JMS, definisce il modo in cui gli oggetti JMS sono correlati ai concetti di WebSphere MQ . La specifica JMS prevede che alcuni oggetti JMS siano oggetti gestiti.

La specifica JMS e il pacchetto javax.jms definiscono una serie di interfacce che possono essere utilizzate dalle applicazioni Java per eseguire operazioni di messaggistica. Il seguente elenco riepiloga le interfacce JMS principali:

Destination
Una destinazione è il punto in cui un'applicazione invia i messaggi o è un'origine da cui un'applicazione riceve i messaggi o entrambi.

ConnectionFactory
Un oggetto ConnectionFactory contiene una serie di proprietà di configurazione per una connessione. Un'applicazione utilizza una produzione connessioni per creare una connessione.

Connessione
Un oggetto Connection incapsula una connessione attiva dell'applicazione a un server di messaggistica. Un'applicazione utilizza una connessione per creare sessioni.

Sessione
Una sessione è un contesto a thread singolo per inviare e ricevere messaggi. Un'applicazione utilizza una sessione per creare messaggi, produttori di messaggi e utenti di messaggi. Una sessione è sottoposta a transazione o non è stata sottoposta a transazione.

Messaggio
Un oggetto Message incapsula un messaggio che un'applicazione invia o riceve.

MessageProducer
Un'applicazione utilizza un produttore di messaggi per inviare messaggi a una destinazione.

MessageConsumer
Un'applicazione utilizza un utente di messaggi per ricevere i messaggi inviati a una destinazione.

Figura 127 a pagina 807 mostra questi oggetti e le relazioni.

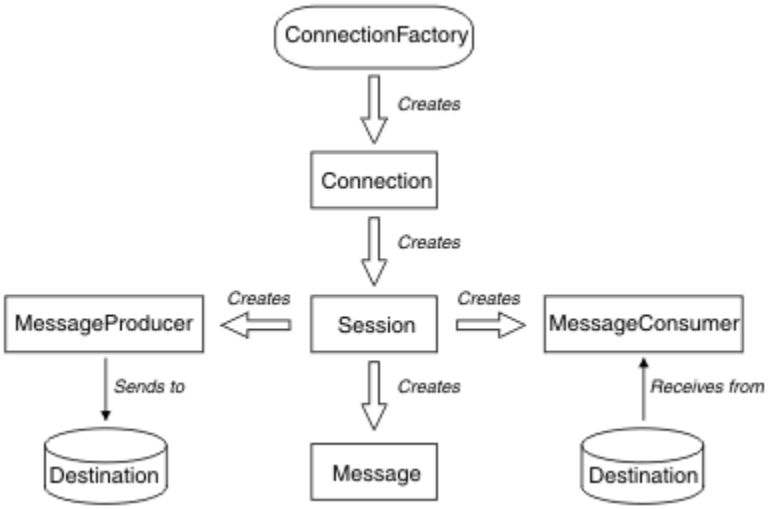


Figura 127. Oggetti JMS e relative relazioni

Un oggetto di destinazione, ConnectionFactoryo Connection può essere utilizzato contemporaneamente da thread differenti di un'applicazione a più thread, ma un oggetto Session, MessageProducero MessageConsumer non può essere utilizzato contemporaneamente da thread differenti. Il modo più semplice per garantire che un oggetto Session, MessageProducero MessageConsumer non venga utilizzato contemporaneamente consiste nel creare un oggetto Session separato per ciascun thread.

JMS supporta due stili di messaggistica:

- Messaggistica point-to-point
- Pubblicazione/sottoscrizione della messaggistica

Questi stili di messaggistica vengono anche indicati come *domini di messaggistica*ed è possibile combinare entrambi gli stili di messaggistica in un'applicazione. Nel dominio point-to-point, una destinazione è una coda e, nel dominio di pubblicazione / sottoscrizione, una destinazione è un argomento.

Con versioni di JMS precedenti a JMS 1.1, la programmazione per il dominio point - to - point utilizza una serie di interfacce e metodi e la programmazione per il dominio di pubblicazione / sottoscrizione utilizza un'altra serie. I due set sono simili, ma separati. Con JMS 1.1, è possibile utilizzare una serie di interfacce e metodi comuni che supportano entrambi i domini di messaggistica. Le interfacce comuni forniscono una vista indipendente dal dominio di ciascun dominio di messaggistica. Tabella 103 a pagina 808 elenca le interfacce indipendenti del dominio JMS e le relative interfacce specifiche del dominio corrispondenti.

Tabella 103. Le interfacce specifiche e indipendenti del dominio JMS		
Interfacce indipendenti dal dominio	**Interfacce specifiche del dominio per il dominio point - to - point**	**Interfacce specifiche del dominio per il dominio di pubblicazione / sottoscrizione**
ConnectionFactory	Factory QueueConnection	Factory TopicConnection
Connessione	QueueConnection	TopicConnection
Destination	Coda	Argomento
Sessione	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

JMS 1.1 conserva tutte le interfacce specifiche del dominio, pertanto le applicazioni esistenti possono ancora utilizzare tali interfacce. Per le nuove applicazioni, tuttavia, considerare l'utilizzo delle interfacce indipendenti dal dominio.

Nelle classi WebSphere MQ per JMS, gli oggetti JMS sono correlati ai concetti di WebSphere MQ nei seguenti modi:

- Un oggetto Connection ha proprietà derivate dalle proprietà del factory di connessione utilizzato per creare la connessione. Queste proprietà controllano il modo in cui un'applicazione si connette a un gestore code. Esempi di queste proprietà sono il nome del gestore code e, per un'applicazione che si connette al gestore code in modalità client, il nome host o l'indirizzo IP del sistema su cui è in esecuzione il gestore code.
- Un oggetto Session incapsula un handle di connessione WebSphere MQ , che quindi definisce l'ambito transazionale della sessione.
- Un oggetto MessageProducer e un oggetto MessageConsumer incapsulano un handle di oggetto WebSphere MQ .

Quando si utilizzano le classi WebSphere MQ per JMS, si applicano tutte le normali regole di WebSphere MQ . Notare, in particolare, che un'applicazione può inviare un messaggio a una coda remota, ma può ricevere un messaggio solo da una coda di proprietà del gestore code a cui è connessa l'applicazione.

La specifica JMS prevede che gli oggetti ConnectionFactory e Destination siano oggetti gestiti. Un amministratore crea e gestisce oggetti gestiti in un repository centrale e un'applicazione JMS richiama tali oggetti utilizzando JNDI (Java Naming and Directory Interface).

In WebSphere MQ classes per JMS, l'implementazione dell'interfaccia di destinazione è una superclasse astratta di Coda e Argomento, e quindi un'istanza di Destinazione è un oggetto Coda o un oggetto Argomento. Le interfacce indipendenti dal dominio trattano una coda o un argomento come una destinazione. Il dominio di messaggistica per un oggetto MessageProducer o MessageConsumer è determinato se la destinazione è una coda o un argomento.

In WebSphere MQ classes per JMS, quindi, gli oggetti dei seguenti tipi possono essere oggetti gestiti:

- ConnectionFactory
- Factory QueueConnection
- Factory TopicConnection
- Coda
- Argomento
- XAConnectionFactory
- Factory XAQueueConnection
- Factory XATopicConnection

Messaggi JMS

I messaggi JMS sono composti da un'intestazione, proprietà e un corpo. JMS definisce cinque tipi di corpo del messaggio.

I messaggi JMS sono composti dalle seguenti parti:

Intestazione
Tutti i messaggi supportano la stessa serie di campi di intestazione. I campi di intestazione contengono valori utilizzati dai client e dai provider per identificare e instradare i messaggi.

Proprietà
Ogni messaggio contiene una funzionalità integrata per supportare i valori delle proprietà definite dall'applicazione. Le proprietà forniscono un meccanismo efficiente per filtrare i messaggi definiti dall'applicazione.

Corpo
JMS definisce diversi tipi di corpo del messaggio che coprono la maggior parte degli stili di messaggistica attualmente in uso.

JMS definisce cinque tipi di corpo del messaggio:

Flusso
Un flusso di valori primitivi Java. Viene riempito e letto in modo sequenziale.

Associa
Una serie di coppie nome - valore, dove i nomi sono stringhe e i valori sono tipi primitivi Java. È possibile accedere alle voci in modo sequenziale o casuale per nome. L'ordine delle voci non è definito.

Testo
Un messaggio contenente un java.lang.String.

Oggetto
Un messaggio contenente un oggetto Java serializzabile

Byte
Un flusso di byte non interpretati. Questo tipo di messaggio è per codificare letteralmente un corpo in modo che corrisponda a un formato di messaggio esistente.

Il campo di intestazione JMSCorrelationID viene utilizzato per collegare un messaggio con un altro. In genere collega un messaggio di risposta con il messaggio di richiesta. JMSCorrelationID può contenere un ID messaggio specifico del provider, una stringa specifica dell'applicazione o un valore byte nativo del provider [].

Selettori di messaggi in JMS

I messaggi possono contenere valori di proprietà definiti dall'applicazione. Un'applicazione può utilizzare i selettori di messaggi per disporre di messaggi di filtro del provider JMS.

Un messaggio contiene una funzionalità integrata per supportare i valori delle proprietà definite dall'applicazione. In effetti, ciò fornisce un meccanismo per aggiungere campi di intestazione specifici dell'applicazione a un messaggio. Le proprietà consentono a un'applicazione, utilizzando i selettori dei messaggi, di fare in modo che un provider JMS selezioni o filtri i messaggi per suo conto, utilizzando criteri specifici dell'applicazione. Le proprietà definite dall'applicazione devono rispettare le seguenti regole:

- I nomi delle proprietà devono rispettare le regole per un identificativo del selettore messaggi.
- I valori delle proprietà possono essere booleani, byte, short, int, long, float, double e String.
- I prefissi nome JMSX e JMS_ sono riservati.

I valori delle proprietà vengono impostati prima di inviare un messaggio. Quando un client riceve un messaggio, le proprietà del messaggio sono di sola lettura. Se un client tenta di impostare le proprietà a questo punto, viene generata unaWriteableException MessageNot. Se viene richiamato clearProperties , le proprietà possono ora essere lette e scritte.

Un valore di proprietà potrebbe duplicare un valore in un corpo del messaggio. JMS non definisce una politica per ciò che potrebbe essere creato in una proprietà. Tuttavia, gli sviluppatori di applicazioni

devono essere consapevoli che i fornitori JMS probabilmente gestiscono i dati in un corpo del messaggio in modo più efficiente rispetto ai dati nelle proprietà del messaggio. Per ottenere prestazioni ottimali, le applicazioni devono utilizzare le proprietà del messaggio solo quando è necessario personalizzare un'intestazione del messaggio. Il motivo principale di questa operazione è il supporto della selezione di messaggi personalizzati.

Un selettore messaggi JMS consente a un cliente di specificare i messaggi a cui è interessato utilizzando l'intestazione del messaggio. Vengono recapitati solo i messaggi con intestazioni che corrispondono al selettore.

I selettori dei messaggi non possono fare riferimento ai valori del corpo del messaggio.

Un selettore di messaggi corrisponde a un messaggio quando il selettore viene valutato true quando il campo di intestazione del messaggio e i valori della proprietà vengono sostituiti per i corrispondenti identificativi nel selettore.

Un selettore di messaggi è una stringa, con sintassi basata su un sottoinsieme della sintassi dell'espressione condizionale SQL92 . L'ordine in cui viene valutato un selettore di messaggi è da sinistra a destra all'interno di un livello di precedenza. È possibile utilizzare le parentesi per modificare questo ordine. I valori letterali selettori predefiniti e i nomi degli operatori vengono scritti qui in maiuscolo; tuttavia, non sono sensibili al maiuscolo / minuscolo.

Un selettore può contenere:

- Valori letterali
  - Un letterale stringa è racchiuso tra virgolette. Una virgoletta doppia rappresenta una virgoletta. Gli esempi sono 'literal' e 'literal' '. Come i letterali stringa Java, questi utilizzano la codifica dei caratteri Unicode.
  - Una costante letterale numerica esatta è un valore numerico senza una virgola decimale, come 57, -957 e +62. Sono supportati i numeri compresi nell'intervallo Java long.
  - Un letterale numerico approssimativo è un valore numerico in notazione scientifica, come 7E3 o -57.9E2, oppure un valore numerico con un decimale, come 7., -95.7o +6.2. Sono supportati i numeri nell'intervallo di Java double.
  - I letterali booleani TRUE e FALSE.
- Identificativi:
  - Un identificativo è una sequenza di lunghezza illimitata di lettere Java e cifre Java, la prima delle quali deve essere una lettera Java. Una lettera è qualsiasi carattere per cui il metodo Character.isJavaLetter restituisce true. Sono inclusi _ e $. Una lettera o una cifra è qualsiasi carattere per cui il metodo Character.isJavaLetterOrDigit restituisce true.
  - Gli identificativi non possono essere i nomi NULL, TRUE o FALSE.
  - Gli identificatori non possono essere NOT, AND, OR, BETWEEN, LIKE, IN o IS.
  - Gli identificatori sono riferimenti di campi di intestazione o riferimenti di proprietà.
  - Gli identificatori sono sensibili al maiuscolo / minuscolo.
  - I riferimenti del campo di intestazione del messaggio sono limitati a:
    - JMSDeliveryMode
    - JMSPriority
    - JMSMessageID
    - JMSTimestamp
    - JMSCorrelationID
    - JMSType

    JMSMessageID, JMSTimestamp, JMSCorrelationIDe i valori JMSType possono essere null e, in tal caso, vengono considerati come un valore NULL.
  - Qualsiasi nome che inizia con JMSX è un nome di proprietà definito da JMS.

  - Qualsiasi nome che inizia con JMS_ è un nome di proprietà specifico del provider.
  - Qualsiasi nome che non inizia con JMS è un nome di proprietà specifico dell'applicazione. Se esiste un riferimento a una proprietà che non esiste in un messaggio, il suo valore è NULL. Se esiste, il suo valore è il valore della proprietà corrispondente.
- Lo spazio vuoto è lo stesso definito per Java: spazio, tabulazione orizzontale, avanzamento modulo e terminazione riga.
- Espressioni:
  - Un selettore è un'espressione condizionale. Un selettore che assume il valore di true corrisponde; un selettore che assume il valore di false o unknown non corrisponde.
  - Le espressioni aritmetiche sono composte da se stesse, operazioni aritmetiche, identificatori (con un valore trattato come una costante letterale numerica) e costanti letterali numeriche.
  - Le espressioni condizionali sono composte da se stesse, operazioni di confronto e operazioni logiche.
- La parentesi standard (), per impostare l'ordine in cui vengono valutate le espressioni, è supportata.
- Operatori logici in ordine di precedenza: NOT, AND, OR.
- Operatori di confronto: =,>,> =, <, < =, <> (non uguale).
  - È possibile confrontare solo i valori dello stesso tipo. Un'eccezione è che è valido per confrontare valori numerici esatti e valori numerici approssimati. (La conversione del tipo richiesta è definita dalle regole della promozione numerica Java.) Se si tenta di confrontare tipi diversi, il selettore è sempre false.
  - Il confronto tra stringhe e valori booleani è limitato a = e <>. Due stringhe sono uguali solo se contengono la stessa sequenza di caratteri.
- Operatori aritmetici in ordine di precedenza:
  - +, - unario.
  - *, /, moltiplicazione e divisione.
  - +, -, addizione e sottrazione.
  - Le operazioni aritmetiche su un valore NULL non sono supportate. Se vengono tentati, il selettore completo è sempre false.
  - Le operazioni aritmetiche devono utilizzare la promozione numerica Java.
- Operatore di confronto arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 e arithmetic-expr3 :
  - L'età BETWEEN 15 and 19 è equivalente all'età> = 15 AND age < = 19.
  - L'età NON COMPRESA tra 15 e 19 è equivalente all'età < 15 OR età> 19.
  - Se una delle espressioni di un'operazione BETWEEN è NULL, il valore dell'operazione è false. Se una delle espressioni di un'operazione NOT BETWEEN è NULL, il valore dell'operazione è true.
- identificativo [NOT] IN (string-literal1, string-literal2, ...) operatore di confronto in cui l'identificativo ha un valore stringa o NULL.
  - Il paese IN ("Regno Unito", "Stati Uniti", "Francia") è vero per "Regno Unito" e falso per "Perù". Equivale all'espressione (Paese = 'UK ') OR (Paese = 'US') OR (Paese = 'Francia ').
  - Il paese NOT IN ("Regno Unito", "Stati Uniti", "Francia") è falso per "Regno Unito" e vero per "Perù". È equivalente all'espressione NOT ((Paese = 'UK ') OR (Paese = 'US') OR (Paese = 'Francia ')).
  - Se l'identificativo di un'operazione IN o NOT IN è NULL, il valore dell'operazione è sconosciuto.
- l'operatore di confronto [NOT] LIKE valore - modello [ESCAPE carattere - escape], dove l'identificativo ha un valore stringa. pattern - value è una stringa letterale, dove _ sta per qualsiasi carattere singolo e% sta per qualsiasi sequenza di caratteri (inclusa la sequenza vuota). Tutti gli altri personaggi rappresentano se stessi. Il carattere escape facoltativo è una stringa di caratteri singoli, con un carattere che viene utilizzato per eseguire l'escape del significato speciale di _ e% nel valore del modello.
  - telefono LIKE '12%3' è true per 123 e 12993 e false per 1234.

  - la parola LIKE 'l_se' è true per "lose" e false per "loose".
  - LIKE '_ %' ESCAPE' \ ' è true per "_foo" e false per "bar".
  - telefono NOT LIKE '12%3' è false per 123 e 12993 e true per 1234.
  - Se l'identificativo di un'operazione LIKE o NOT LIKE è NULL, il valore dell'operazione è sconosciuto.
- L'operatore di confronto IS NULL verifica un valore campo di intestazione null o un valore di proprietà mancante.
  - nome_prop IS NULL.
- L'operatore di confronto IS NOT NULL verifica l'esistenza di un valore campo di intestazione non null o di un valore di proprietà.
  - nome_prop NON È NULL.

Il seguente selettore di messaggi seleziona i messaggi con un tipo di messaggio di auto, un colore di blu e un peso maggiore di 2500 libbre:

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

Come indicato nell'elenco precedente, i valori delle proprietà possono essere NULL. La valutazione delle espressioni del selettore che contengono valori NULL è definita dalla semantica SQL 92 NULL. Il seguente elenco fornisce una breve descrizione di queste semantiche:

- SQL considera un valore NULL come sconosciuto.
- Il confronto o l'aritmetica con un valore sconosciuto produce sempre un valore sconosciuto.
- L'operatore IS NULL converte un valore sconosciuto in un valore TRUE.
- L'operatore IS NOT NULL converte un valore sconosciuto in un valore FALSE.

Sebbene SQL supporti il confronto decimale fisso e l'aritmetica, i selettori dei messaggi JMS non lo fanno. Questo è il motivo per cui le costanti letterali numeriche esatte sono limitate a quelle senza un decimale. È anche il motivo per cui ci sono numeri con un decimale come rappresentazione alternativa per un valore numerico approssimativo.

I commenti SQL non sono supportati.

Associazione dei messaggi JMS sui messaggi WebSphere MQ

I messaggi WebSphere MQ sono composti da un descrittore di messaggi, un'intestazione MQRFH2 facoltativa e un corpo. Il contenuto di un messaggio JMS è in parte associato e in parte copiato in un messaggio WebSphere MQ .

Questo argomento descrive come la struttura del messaggio JMS descritta nella prima parte di questa sezione viene mappata su un messaggio WebSphere MQ . È di interesse per i programmatori che desiderano trasmettere messaggi tra le applicazioni JMS e WebSphere MQ tradizionali. È anche di interesse per le persone che desiderano manipolare i messaggi trasmessi tra due applicazioni JMS, ad esempio, in una implementazione di WebSphere Message Broker.

Questa sezione non si applica se un'applicazione utilizza una connessione in tempo reale ad un broker. Quando un'applicazione utilizza una connessione in tempo reale, tutte le comunicazioni vengono eseguite direttamente su TCP/IP; non sono coinvolti messaggi o code WebSphere MQ .

WebSphere I messaggi di MQ sono composti da tre componenti:

- MQMD (WebSphere MQ Message Descriptor)
- Un'intestazione WebSphere MQ MQRFH2
- Il contenuto del messaggio.

MQRFH2 è facoltativo e la sua inclusione in un messaggio in uscita è regolata da un indicatore nella classe di destinazione JMS. È possibile impostare questo indicatore utilizzando lo strumento di gestione JMS WebSphere MQ . Poiché MQRFH2 contiene informazioni specifiche di JMS, includerlo sempre nel messaggio quando il mittente sa che la destinazione di ricezione è un'applicazione JMS. In genere,

omettere MQRFH2 quando si invia un messaggio direttamente a una applicazione non JMS. Ciò è dovuto al fatto che un'applicazione di questo tipo non prevede un MQRFH2 nel messaggio WebSphere MQ .

Se un messaggio in ingresso non dispone di un'intestazione MQRFH2 , l'oggetto Coda o Argomento derivato dal campo di intestazione JMSReplyTo del messaggio, per impostazione predefinita, ha questo indicatore impostato in modo che anche un messaggio di risposta inviato alla coda o all'argomento non abbia un'intestazione MQRFH2 . È possibile disattivare questo comportamento includendo un'intestazione MQRFH2 in un messaggio di risposta solo se il messaggio originale ha un'intestazione MQRFH2 , impostando la proprietà TARGCLIENTMATCHING del factory di connessione su NO.

Figura 128 a pagina 813 mostra in che modo la struttura di un messaggio JMS viene trasformata in un messaggio WebSphere MQ e di nuovo:

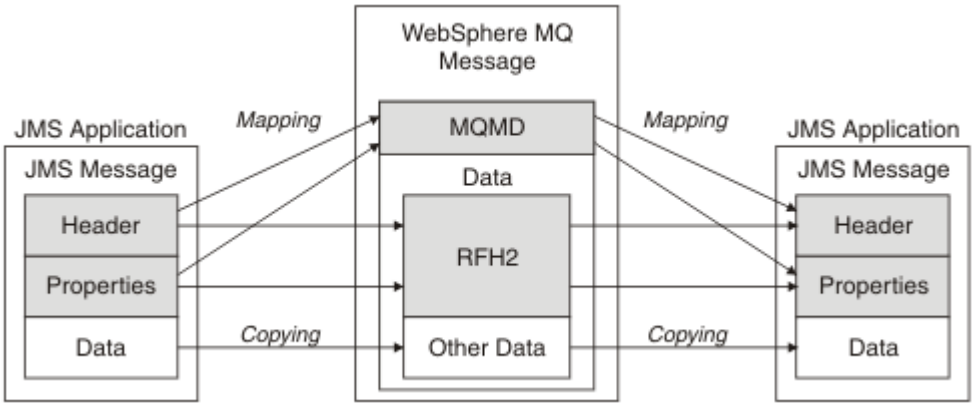


Figura 128. Modalità di trasformazione dei messaggi tra JMS e WebSphere MQ utilizzando l'intestazione MQRFH2

Le strutture si trasformano in due modi:

Associazione
Quando MQMD include un campo equivalente al campo JMS, il campo JMS viene associato al campo MQMD. Ulteriori campi MQMD vengono esposti come proprietà JMS, poiché un'applicazione JMS potrebbe dover acquisire o impostare questi campi durante la comunicazione con un'applicazione non JMS.

Copia in corso
Dove non esiste un equivalente MQMD, viene passato un campo o una proprietà dell'intestazione JMS, possibilmente trasformata, come un campo all'interno di MQRFH2.

L'intestazione MQRFH2 e JMS
Questa raccolta di argomenti descrive l'intestazione MQRFH Versione 2, che contiene i dati specifici JMS associati al contenuto del messaggio. MQRFH2 Versione 2 è un'intestazione estensibile e può contenere anche ulteriori informazioni non direttamente associate a JMS. Tuttavia, questa sezione riguarda solo l'utilizzo da parte di JMS.Per una descrizione completa, consultare MQRFH2 - Regole e intestazione di formattazione 2.

Esistono due parti dell'intestazione, una parte fissa e una parte variabile.

Parte fissa
La parte fissa è modellata sul modello di intestazione *standard* WebSphere MQ ed è composta dai seguenti campi:

StrucId (MQCHAR4)
Identificatore struttura.

Deve essere MQRFH_STRUC_ID (valore: "RFH ") (valore iniziale).

Viene definito anche MQRFH_STRUC_ID_ARRAY (valore: "R","F","H"," ").

Version (MQLONG)
Numero di versione della struttura.

Deve essere MQRFH_VERSION_2 (valore: 2) (valore iniziale).

StrucLength (MQLONG)
Lunghezza totale di MQRFH2, inclusi i campi di dati NameValue.

Il valore impostato in StrucLength deve essere un multiplo di 4 (i dati nei campi NameValueData possono essere riempiti con spazi).

Encoding (MQLONG)
Codifica dati.

Codifica di tutti i dati numerici nella porzione del messaggio che segue MQRFH2 (l'intestazione successiva o i dati del messaggio che seguono questa intestazione).

CodedCharSetId (MQLONG)
Coded character set identifier (CCSID)

Rappresentazione di qualsiasi dato carattere nella parte del messaggio che segue MQRFH2 (l'intestazione successiva o i dati del messaggio che seguono questa intestazione).

Formato (MQCHAR8)
Nome formato.

Nome formato per la parte del messaggio che segue MQRFH2.

Indicatori (MQLONG)
Indicatori.

MQRFH_NO_FLAGS = 0. Nessun indicatore impostato.

CCSID NameValue(MQLONG)
Il CCSID (coded character set identifier) per le stringhe di caratteri di dati NameValuecontenute in questa intestazione. I dati NameValuepossono essere codificati in una serie di caratteri diversa dalle altre stringhe di caratteri contenute nell'intestazione (StrucID e Format).

Se il CCSID di NameValueè un CCSID Unicode a 2 byte (1200, 13488 o 17584), l'ordine di byte di Unicode è lo stesso dell'ordine di byte dei campi numerici in MQRFH2. (Ad esempio, Version, StrucLengthe NameValueCCSID stesso.)

Tabella 104. Valori possibili per il campo CCSID NameValue

Valore	Significato
1200	UCS2 aperto
1208	UTF8
13488	Sottoinsieme UCS2 2.0
17584	Sottoinsieme UCS2 2.1 (include il simbolo Euro)

Parte variabile
La parte variabile segue la porzione fissa. La parte variabile contiene un numero variabile di cartelle MQRFH2 . Ogni cartella contiene un numero variabile di elementi o proprietà. Proprietà relative al gruppo di cartelle. Le intestazioni MQRFH2 create da JMS possono contenere una delle seguenti cartelle:

Cartella < mcd>

`mcd` contiene proprietà che descrivono il formato del messaggio. Ad esempio, la proprietà `Msd` del dominio del servizio di messaggi identifica un messaggio JMS come `JMSTextMessage`, `JMSBytesMessage`, `JMSStreamMessage`, `JMSMapMessage`, `JMSObjectMessage` o null.

La cartella `mcd` è sempre presente in un messaggio di JMS contenente un `MQRFH2`.

È sempre presente in un messaggio contenente un `MQRFH2` inviato da WebSphere Message Broker. Descrive il dominio, il formato, il tipo e la serie di un messaggio.

Tabella 105. Nome, sinonimo, tipo di dati e cartella della proprietà mcd			
Sinonimo proprietà	**Nome della proprietà**	**Tipo dati**	**Cartella**
	mcd.Msd	string	<mcd><Msd>*messageDomain*</Msd></mcd>
	mcd.Set	string	<mcd><Set>*messageDomain*</Set></mcd>
	mcd.Type	string	<mcd><Type>*messageDomain*</Type></mcd>
	mcd.Fmt	string	<mcd><Fmt>*messageDomain*</Fmt></mcd>

Non aggiungere le proprie proprietà nella cartella mcd.

Cartella < jms>

jms contiene campi di intestazione JMS e proprietà JMSX che non possono essere pienamente espresse nell'MQMD. La cartella jms è sempre presente in un MQRFH2 JMS.

La cartella < usr>

usr contiene le proprietà di JMS definite dall'applicazione associate al messaggio. La cartella usr è presente solo se un'applicazione ha impostato una proprietà definita dall'applicazione.

Cartella < mqext>

mqext contiene proprietà che vengono utilizzate solo da WebSphere Application Server. La cartella è presente solo se l'applicazione ha impostato almeno una delle proprietà definite da IBM .

Tabella 106. Nome, sinonimo, tipo di dati e cartella della proprietà mqext			
Sinonimo proprietà	**Nome della proprietà**	**Tipo dati**	**Cartella**
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>*armCorrelator*</Arm></mqext>
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>*wrmCorrelator*</Wrm></mqext>

Non aggiungere le proprie proprietà nella cartella mqext.

La cartella < mqps>

mqps contiene proprietà che vengono utilizzate solo dalla pubblicazione/sottoscrizione di IBM WebSphere MQ. La cartella è presente solo se l'applicazione ha impostato almeno una delle proprietà di pubblicazione/sottoscrizione integrate.

Tabella 107. Nome, sinonimo, tipo di dati e cartella della proprietà mqps			
Sinonimo proprietà	**Nome della proprietà**	**Tipo dati**	**Cartella**
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>

Tabella 107. Nome, sinonimo, tipo di dati e cartella della proprietà mqps (Continua)			
Sinonimo proprietà	**Nome della proprietà**	**Tipo dati**	**Cartella**
MQPubOptions	mqps.Pub	i8	<mqps><Pub>*publicationOptions*</Pub></mqps>
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>

Non aggiungere le proprie proprietà nella cartella mqps.

Tabella 108 a pagina 816 mostra un elenco completo di nomi di proprietà.

Tabella 108. Proprietà e cartelle MQRFH2 utilizzate da JMS				
Nome campo JMS	**Tipo Java**	**Nome cartella MQRFH2**	**Nome della proprietà**	**Tipo / valori**
JMSDestination	Destination	jms	Dst	stringa
JMSExpiration	completo	jms	Esp	i8
JMSPriority	int	jms	Pri	i4
JMSDeliveryMode	int	jms	Dlv	i4
JMSCorrelationID	Stringa	jms	Cid	stringa
JMSReplyTo	Destination	jms	Rto	stringa
JMSTimestamp	completo	jms	Tms	i8
JMSType	Stringa	mcd	Tipo, Serie, Fmt	stringa
JMSXGroupID	Stringa	jms	GID	stringa
JMSXGroupSeq	int	jms	Seq	i4
xxx (definito dall'utente)	Qualsiasi	USR	xxx	tutte
		mcd	MSD	jms_none jms_text jms_bytes jms_map jms_stream jms_object

NameValue(MQLONG)
Lunghezza in byte della stringa di dati NameValueche segue immediatamente questo campo di lunghezza (non include la propria lunghezza).

Dati NameValue(MQCHARn)

Una singola stringa di caratteri, la cui lunghezza in byte viene fornita dal precedente campo NameValueLength. Contiene una cartella contenente una sequenza di proprietà. Ciascuna proprietà è una tripletta nome / tipo/valore, contenuta in un elemento XML il cui nome è il nome della cartella, come segue:

```
 <foldername>
triplet1 triplet2 .....   tripletn </foldername>
```

La tag </foldername> di chiusura può essere seguita da spazi come caratteri di riempimento. Ogni tripletta è codificata utilizzando una sintassi simile a XML:

```
   <name dt='datatype'>value</name>
```

L'elemento dt='datatype' è facoltativo e viene omesso per molte proprietà, poiché il tipo di dati è predefinito. Se è incluso, uno o più caratteri spazio devono essere inclusi prima del tag dt= .

name
è il nome della proprietà; consultare Tabella 108 a pagina 816.

datatype
deve corrispondere, dopo il raggruppamento, a uno dei tipi di dati elencati in Tabella 109 a pagina 817.

value
è una rappresentazione stringa del valore da trasmettere, utilizzando le definizioni in Tabella 109 a pagina 817.

Un valore null viene codificato utilizzando la sintassi seguente:

```
<name dt='datatype' xsi:nil='true'></name>
```

Non utilizzare xsi:nil='false'.

Tabella 109. Tipi di dati proprietà

Tipo dati	Definizione
stringa	Qualsiasi sequenza di caratteri esclusi < e &
booleano	Il carattere 0 o 1 (0 = false, 1 = true)
bin.hex	Si tratta di cifre esadecimali che rappresentano ottetti.
i1	Un numero, espresso con cifre 0..9, con segno facoltativo (nessuna frazione o esponente). Deve essere compreso nell'intervallo tra -128 e 127 inclusi
i2	Un numero, espresso con cifre 0..9, con segno facoltativo (nessuna frazione o esponente). Deve essere compreso nell'intervallo tra -32768 e 32767 incluso
i4	Un numero, espresso con cifre 0..9, con segno facoltativo (nessuna frazione o esponente). Deve essere compreso tra -2147483648 e 2147483647 inclusi
i8	Un numero, espresso con cifre 0..9, con segno facoltativo (nessuna frazione o esponente). Deve essere compreso tra -9223372036854775808 e 9223372036854775807 inclusi
int	Un numero, espresso con cifre 0..9, con segno facoltativo (nessuna frazione o esponente). Deve essere nello stesso intervallo di i8. Può essere utilizzato al posto di uno dei tipi i * se il mittente non desidera associare una particolare precisione alla proprietà
r4	Numero a virgola mobile, grandezza < = 3.40282347E+38,> = 1.175E-37 espresso con cifre 0..9, segno facoltativo, cifre frazionarie facoltative, esponente facoltativo

Tabella 109. Tipi di dati proprietà (Continua)	
Tipo dati	**Definizione**
r8	Numero a virgola mobile, magnitudine < = 1.7976931348623E+308,> = 2.225E-307 espresso con cifre `0..9`, segno facoltativo, cifre frazionarie facoltative, esponente facoltativo

Un valore stringa può contenere spazi. È necessario utilizzare le seguenti sequenze di escape in un valore stringa:

- `&` per il carattere &
- < per il carattere <

È possibile utilizzare le seguenti sequenze di escape, ma non sono richieste:

- `>` per il carattere >
- `&apos;` per il carattere '
- `&quot;` per il carattere "

Campi e proprietà JMS con campi MQMD corrispondenti
Queste tabelle mostrano i campi MQMD equivalenti ai campi di intestazione JMS, alle proprietà JMS e alle proprietà del provider JMS.

Tabella 110 a pagina 818 elenca i campi di intestazione JMS e Tabella 111 a pagina 818 elenca le proprietà JMS associate direttamente ai campi MQMD. Tabella 112 a pagina 819 elenca le proprietà specifiche del fornitore e i campi MQMD a cui sono associati.

Tabella 110. Campi di intestazioni JMS associati ai campi MQMD			
Campo intestazione JMS	**Tipo Java**	**Campo MQMD**	**Tipo C**
JMSDeliveryMode	int	Persistenza	MQLONG
JMSExpiration	completo	Scadenza	MQLONG
JMSPriority	int	Priorit...	MQLONG
JMSMessageID	Stringa	MsgID	MQBYTE24
JMSTimestamp	completo	PutDate PutTime	MQCHAR8 MQCHAR8
JMSCorrelationID	Stringa	CorrelId	MQBYTE24

Tabella 111. Associazione delle proprietà JMS ai campi MQMD			
proprietà JMS	**Tipo Java**	**Campo MQMD**	**Tipo C**
JMSXUserID	Stringa	UserIdentifier	MQCHAR12
JMSXAppID	Stringa	PutApplName	MQCHAR28
JMSXDeliveryCount	int	BackoutCount	MQLONG
JMSXGroupID	Stringa	GroupId	MQBYTE24
JMSXGroupSeq	int	MsgSeqNumber	MQLONG

Tabella 112. Proprietà specifiche del fornitore JMS associate ai campi MQMD

proprietà specifica del provider JMS	**Tipo Java**	**Campo MQMD**	**Tipo C**
Eccezione JMS_IBM_Report_Exception	int	Prospetto	MQLONG
Scadenza report IBM JMS	int	Prospetto	MQLONG
COA report IBM JMS	int	Prospetto	MQLONG
COD report IBM JMS	int	Prospetto	MQLONG
PAN report IBM JMS	int	Prospetto	MQLONG
NAN report IBM JMS	int	Prospetto	MQLONG
ID_ms_ms_IBM_Report_Pass_Msg_ID	int	Prospetto	MQLONG
ID correlazione password report IBM JMS	int	Prospetto	MQLONG
JMS_IBM_Report_Discard_Msg	int	Prospetto	MQLONG
JMS_IBM_MsgType	int	MsgType	MQLONG
Feedback IBM JMS	int	Feedback	MQLONG
Formato_IBM_JMS	Stringa	Formato"1" a pagina 819	MQCHAR8
JMS_IBM_PutApplTipo	int	PutApplType	MQLONG
JMS_IBM_Encoding	int	Codifica	MQLONG
Serie di caratteristiche IBM JMS	Stringa	CodedCharacterSetId"2" a pagina 819	MQLONG
JMS_IBM_PutDate	Stringa	PutDate	MQCHAR8
JMS_IBM_PutTime	Stringa	PutTime	MQCHAR8
JMS_IBM_Last_Msg_In_Group	booleano	MsgFlags	MQLONG

Nota:

1. JMS_IBM_Format rappresenta il formato del corpo del messaggio. Ciò può essere definito dall'applicazione che imposta la proprietà JMS_IBM_Format del messaggio (notare che è presente un limite di 8 caratteri) o può essere impostato sul formato WebSphere MQ del corpo del messaggio appropriato per il tipo di messaggio JMS. JMS_IBM_Format si associa al campo Formato MQMD solo se il messaggio non contiene sezioni RFH o RFH2 . In un messaggio tipico, si associa al campo Formato di RFH2 immediatamente precedente il corpo del messaggio.
2. Il valore della proprietà JMS_IBM_Character_Set è un valore stringa che contiene l'equivalente della serie di caratteri Java per il valore numerico CodedCharacterSetId . Il campo MQMD CodedCharacterSetId è un valore numerico che contiene l'equivalente della stringa della serie di caratteri Java specificata dalla proprietà JMS_IBM_Character_Set.

Mappatura dei campi JMS sui campi WebSphere MQ (messaggi in uscita)
Queste tabelle mostrano il modo in cui i campi di proprietà e di intestazione JMS vengono mappati nei campi MQMD e MQRFH2 al momento di send () o publish ().

Tabella 113 a pagina 820 mostra in che modo i campi di intestazione JMS vengono mappati nei campi MQMD/RFH2 al momento di send () o publish (). Tabella 114 a pagina 820 mostra il modo in cui le proprietà JMS vengono associate nei campi MQMD/RFH2 al momento di send () o publish (). Tabella 115 a pagina 821 mostra in che modo le proprietà specifiche del provider JMS vengono associate ai campi MQMD al momento di send () o publish (),

Per i campi contrassegnati da Oggetto messaggio, il valore trasmesso è il valore contenuto nel messaggio JMS immediatamente prima dell'operazione send () o publish (). Il valore nel messaggio JMS viene lasciato invariato dall'operazione.

Per i campi contrassegnati come impostati dal metodo di invio, viene assegnato un valore quando viene eseguito send () o publish () (qualsiasi valore contenuto nel messaggio JMS viene ignorato). Il valore nel messaggio JMS viene aggiornato per mostrare il valore utilizzato.

I campi contrassegnati come Solo ricezione non vengono trasmessi e vengono lasciati invariati nel messaggio da send () o publish ().

Tabella 113. Mappatura del campo del messaggio in uscita

Nome campo intestazione JMS	**Campo MQMD utilizzato per la trasmissione**	**Intestazione**	**Impostato da**
JMSDestination		MQRFH2	Metodo di invio
JMSDeliveryMode	Persistenza	MQRFH2	Metodo di invio
JMSExpiration	Scadenza	MQRFH2	Metodo di invio
JMSPriority	Priorit...	MQRFH2	Metodo di invio
JMSMessageID	MsgID		Metodo di invio
JMSTimestamp	PutDate/PutTime		Metodo di invio
JMSCorrelationID	CorrelId	MQRFH2	Oggetto messaggio
JMSReplyTo	Gestore code ReplyToQ/ ReplyTo	MQRFH2	Oggetto messaggio
JMSType		MQRFH2	Oggetto messaggio
JMSRedelivered			Solo ricezione

Nota:

1. Il campo MQMD CodedCharacterSetId è un valore numerico che contiene l'equivalente della stringa della serie di caratteri Java specificata dalla proprietà JMS_IBM_Character_Set.

Tabella 114. Mappatura proprietà JMS del messaggio in uscita

Nome proprietà JMS	**Campo MQMD utilizzato per la trasmissione**	**Intestazione**	**Impostato da**
JMSXUserID	UserIdentifier		Metodo di invio
JMSXAppID	PutApplName		Metodo di invio
JMSXDeliveryCount			Solo ricezione
JMSXGroupID	GroupId	MQRFH2	Oggetto messaggio
JMSXGroupSeq	MsgSeqNumber	MQRFH2	Oggetto messaggio

Tabella 115. Associazione proprietà specifica del provider JMS del messaggio in uscita

Nome della proprietà specifica del provider JMS	Campo MQMD utilizzato per la trasmissione	Intestazione	Impostato da
Eccezione JMS_IBM_Report_Exception	Prospetto		Oggetto messaggio
Scadenza report IBM JMS	Prospetto		Oggetto messaggio
COA/COD report IBM JMS	Prospetto		Oggetto messaggio
NAN/PAN report IBM JMS	Prospetto		Oggetto messaggio
ID_ms_ms_IBM_Report_Pass_Msg_ID	Prospetto		Oggetto messaggio
ID correlazione password report IBM JMS	Prospetto		Oggetto messaggio
JMS_IBM_Report_Discard_Msg	Prospetto		Oggetto messaggio
JMS_IBM_MsgType	MsgType		Oggetto messaggio
Feedback IBM JMS	Feedback		Oggetto messaggio
Formato_IBM_JMS	Formato		Oggetto messaggio
JMS_IBM_PutApplTipo	PutApplType		Metodo di invio
JMS_IBM_Encoding	Codifica		Oggetto messaggio
Serie di caratteristiche IBM JMS	CodedCharacterSetId		Oggetto messaggio
JMS_IBM_PutDate	PutDate		Metodo di invio
JMS_IBM_PutTime	PutTime		Metodo di invio
JMS_IBM_Last_Msg_In_Group	MsgFlags		Oggetto messaggio

Associazione dei campi di intestazione JMS in send () o publish ()
Queste note sono relative alla mappatura dei campi JMS in send () o publish ().

JMSDestination a MQRFH2
Viene memorizzato come una stringa che serializza le caratteristiche salienti dell'oggetto destinazione, in modo che un JMS di ricezione possa ricostituire un oggetto di destinazione equivalente. Il campo MQRFH2 è codificato come URI (consultare "URI (Uniform Resource Identifier)" a pagina 881 per i dettagli della notazione URI).

Da JMSReplyTo a MQMD.ReplyToQ, ReplyTo, MQRFH2
Il nome della coda viene copiato in MQMD.ReplyToQ e il nome del gestore code viene copiato nei campi del gestore code ReplyTo. Le informazioni sull'estensione di destinazione (altri dettagli utili conservati nell'oggetto di destinazione) vengono copiati nel campo MQRFH2 . Il campo MQRFH2 è codificato come URI (consultare "URI (Uniform Resource Identifier)" a pagina 881 per i dettagli della notazione URI).

JMSDeliveryMode su MQMD.Persistence

Il valore JMSDeliveryMode viene impostato dal metodo send () o publish () o MessageProducer, a meno che l'oggetto di destinazione non lo sovrascriva. Il valore JMSDeliveryMode è associato a MQMD.Persistence come segue:

- Il valore JMS PERSISTENT è equivalente a MQPER_PERSISTENT
- Il valore JMS NON_PERSISTENT è equivalente a MQPER_NOT_PERSISTENT

Se la proprietà di persistenza di MQQueue non è impostata su WMQConstants.WMQ_PER_QDEF, il valore della modalità di consegna è codificato anche in MQRFH2.

JMSExpiration a/da MQMD.Expiry, MQRFH2

JMSExpiration memorizza l'ora di scadenza (la somma dell'ora corrente e dell'ora di scadenza), mentre MQMD memorizza l'ora di scadenza. Inoltre, JMSExpiration è espresso in millisecondi, ma MQMD.Expiry è in decimi di secondo.

- Se il metodo send () imposta una durata illimitata, MQMD.Expiry è impostata su MQEI_UNLIMITED e nessun JMSExpiration è codificato in MQRFH2.
- Se il metodo send () imposta una durata inferiore a 214748364.7 secondi (circa 7 anni), la durata viene memorizzata in MQMD.Expirye l'ora di scadenza (in millisecondi) vengono codificati come un valore i8 in MQRFH2.
- Se il metodo send () imposta una durata maggiore di 214748364.7 secondi, MQMD.Expiry è impostata su MQEI_UNLIMITED. Il tempo di scadenza reale in millisecondi è codificato come un valore i8 in MQRFH2.

JMSPriority per MQMD.Priority

Associare direttamente il valore JMSPriority (0-9) al valore della priorità MQMD (0-9). Se JMSPriority è impostato su un valore non predefinito, anche il livello di priorità viene codificato in MQRFH2.

JMSMessageID da MQMD.MessageID

Tutti i messaggi inviati da JMS hanno identificativi di messaggi univoci assegnati da WebSphere MQ. Il valore assegnato viene restituito in MQMD.MessageId dopo la chiamata MQPUT e viene passato nuovamente all'applicazione nel campo JMSMessageID . WebSphere MQ messageId è un valore binario di 24 byte, mentre JMSMessageID è una stringa. Il JMSMessageID è composto dal valore binario messageId convertito in una sequenza di 48 caratteri esadecimali, preceduti dal carattere ID:. JMS fornisce un suggerimento che può essere impostato per disabilitare la produzione di identificatori di messaggi. Questo suggerimento viene ignorato e viene assegnato un identificativo univoco in tutti i casi. Qualsiasi valore impostato nel campo JMSMessageId prima di send () viene sovrascritto.

Se si richiede la capacità di specificare l'MQMD MQMD.MessageID, è possibile eseguire questa operazione con una delle estensioni JMS WebSphere MQ descritte in “Lettura e scrittura del descrittore del messaggio da un'applicazione WebSphere MQ classes per JMS” a pagina 897.

JMSTimestamp su MQRFH2

Durante un invio, il campo JMSTimestamp viene impostato in base all'orologio della JVM. Questo valore è impostato in MQRFH2. Qualsiasi valore impostato nel campo JMSTimestamp prima di send () viene sovrascritto. Consultare anche le proprietà JMS_IBM_PutDate e JMS_IBMPutTime .

Da JMSType a MQRFH2

Questa stringa è impostata nel campo MQRFH2 mcd.Type . Se è in formato URI, può interessare anche i campi mcd.Set e mcd.Fmt . Vedi anche “Utilizzo di una connessione in tempo reale a un broker di WebSphere Event Broker o WebSphere Message Broker” a pagina 925.

JMSCorrelationID in MQMD.CorrelId, MQRFH2

Il JMSCorrelationID può contenere uno dei seguenti:

Un ID messaggio specifico del fornitore

Si tratta di un identificativo di messaggio da un messaggio precedentemente inviato o ricevuto, e quindi deve essere una stringa di 48 cifre esadecimali minuscole che hanno come prefisso *ID:* Il prefisso viene rimosso, i restanti caratteri vengono convertiti in binario e quindi impostati in MQMD.CorrelId . Nessun valore CorrelId è codificato in MQRFH2.

Un valore byte nativo del provider []
Il valore viene copiato in MQMD.CorrelId field - riempito con valori null o troncato a 24 byte, se necessario. Nessun valore CorrelId è codificato in MQRFH2.

Una stringa specifica dell'applicazione
Il valore viene copiato in MQRFH2. I primi 24 byte della stringa, in formato UTF8 , vengono scritti nell'MQMD MQMD.CorrelID.

Associazione dei campi delle proprietà JMS
Queste note fanno riferimento all'associazione dei campi delle proprietà JMS nei messaggi WebSphere MQ .

JMSXUserID da MQMD UserIdentifier
JMSXUserID è impostato al ritorno dalla chiamata di invio.

JMSXAppID dal nome MQMD PutAppl
JSMXAppID è impostato al ritorno dalla chiamata di invio.

JMSXGroupID a MQRFH2 (point - to - point)
Per i messaggi point-to-point, JMSXGroupID viene copiato nel campo MQMD GroupID . Se JMSXGroupID inizia con l'ID prefisso:, viene convertito in binario. Altrimenti, viene codificato come una stringa UTF8 . Il valore viene riempito o troncato se necessario ad una lunghezza di 24 byte. L'indicatore MQMF_MSG_IN_GROUP è impostato.

JMSXGroupID a MQRFH2 (pubblicazione / sottoscrizione)
Per i messaggi di pubblicazione / sottoscrizione, JMSXGroupID viene copiato in MQRFH2 come stringa.

JMSXGroupSeq MQMD MsgSeqNumero (point - to - point)
Per i messaggi point-to-point, JMSXGroupSeq viene copiato nel campo MQMD MsgSeqNumber. L'indicatore MQMF_MSG_IN_GROUP è impostato.

JMSXGroupSeq MQMD MsgSeqNumero (pubblicazione/sottoscrizione)
Per i messaggi di pubblicazione / sottoscrizione, JMSXGroupSeq viene copiato in MQRFH2 come i4.

Associazione di campi specifici del provider JMS
Le seguenti note fanno riferimento all'associazione dei campi specifici del provider JMS nei messaggi IBM WebSphere MQ .

Report JMS_IBM_Report_ < name> to MQMD Report
Un'applicazione JMS può impostare le opzioni del report MQMD, utilizzando le seguenti proprietà JMS_IBM_Report_XXX. Il singolo MQMD è associato a diverse proprietà JMS_IBM_Report_XXX. L'applicazione deve impostare il valore di queste proprietà sulle costanti IBM WebSphere MQ MQRO_ standard (incluse in com.ibm.mq.MQC). Quindi, ad esempio, per richiedere COD con Dati completi, l'applicazione deve impostare JMS_IBM_Report_COD sul valore CMQC.MQRO_COD_WITH_FULL_DATA.

Eccezione JMS_IBM_Report_Exception

MQRO_EXCEPTION o
MQRO_EXCEPTION_WITH_DATA o
MQRO_EXCEPTION_WITH_FULL_DATA

Scadenza report IBM JMS

MQRO_EXPIRATION o
MQRO_EXPIRATION_WITH_DATA o
MQRO_EXPIRATION_WITH_FULL_DATA

COA report IBM JMS

MQRO_COA o
MQRO_COA_WITH_DATA o
DATI_COA_WITH_MQRO_FULL_DATA

COD report IBM JMS

MQRO_COD o
MQRO_COD_WITH_DATA o
DAD_COD MQRO_WITH_FULL_DATA

PAN report IBM JMS
MQRO_PAN

NAN report IBM JMS
MQRO_NAN

ID_ms_ms_IBM_Report_Pass_Msg_ID
MQRO_PASS_MSG_ID

ID correlazione password report IBM JMS
ID CORREL_PASS_MQRO_

JMS_IBM_Report_Discard_Msg
MQRO_DISCARD_MSG

JMS_IBM_MsgType in MQMD MsgType
Il valore viene associato direttamente a MQMD MsgType. Se l'applicazione non ha impostato un valore esplicito di JMS_IBM_MsgType, viene utilizzato un valore predefinito. Questo valore predefinito è determinato come segue:

- Se JMSReplyTo è impostato su una destinazione coda IBM WebSphere MQ , MSGType è impostato sul valore MQMT_REQUEST
- Se JMSReplyTo non è impostato o è impostato su un valore diverso da una destinazione coda IBM WebSphere MQ , MsgType è impostato sul valore MQMT_DATAGRAM

Feedback da JMS_IBM_Feedback a MQMD
Il valore viene associato direttamente a MQMD Feedback.

JMS_IBM_Format in formato MQMD
Il valore si associa direttamente al formato MQMD.

Da JMS_IBM_Encoding a MQMD Encoding
Se impostata, questa proprietà sovrascrive la codifica numerica della coda di destinazione o dell'argomento.

JMS_IBM_Character_Set su MQMD CodedCharacterSetId
Se impostata, questa proprietà sovrascrive la proprietà della serie di caratteri codificati della coda di destinazione o dell'argomento.

JMS_IBM_PutDate da MQMD PutDate
Il valore di questa proprietà viene impostato, durante l'invio, direttamente dal campo PutDate in MQMD. Qualsiasi valore impostato nella proprietà JMS_IBM_PutDate prima di un invio viene sovrascritto. Questo campo è una stringa di otto caratteri, nel formato data IBM WebSphere MQ AAAAMMGG. Questa proprietà può essere utilizzata con la proprietà JMS_IBMPutTime per determinare l'ora in cui il messaggio è stato inserito in base al gestore code.

JMS_IBM_PutTime da MQMD PutTime
Il valore di questa proprietà viene impostato, durante l'invio, direttamente dal campo PutTime in MQMD. Qualsiasi valore impostato nella proprietà JMS_IBM_PutTime prima di un invio viene sovrascritto. Questo campo è una stringa di otto caratteri, nel formato ora IBM WebSphere MQ di HHMMSSTH. Questa proprietà può essere utilizzata con la proprietà JMS_IBMPutDate per determinare l'ora in cui il messaggio è stato inserito in base al gestore code.

JMS_IBM_Last_Msg_In_Group a MQMD MsgFlags
Per la messaggistica point - to - point, questo valore booleano si associa all'indicatore MQMF_LAST_MSG_IN_GROUP nel campo MQMD MsgFlags . Di solito viene utilizzato con le proprietà JMSXGroupID e JMSXGroupSeq per indicare a un'applicazione IBM WebSphere MQ legacy che questo messaggio è l'ultimo di un gruppo. Questa proprietà viene ignorata per la messaggistica di pubblicazione / sottoscrizione.

Associazione dei campi WebSphere MQ ai campi JMS (messaggi in entrata)
Queste tabelle mostrano il modo in cui i campi di proprietà e intestazione JMS vengono associati ai campi MQMD e MQRFH2 al momento di get () o receive ().

Tabella 116 a pagina 825 mostra il modo in cui i campi di intestazione JMS vengono associati ai campi MQMD/MQRFH2 al momento di get () o receive (). Tabella 117 a pagina 825 mostra il modo in cui i campi delle proprietà JMS vengono associati ai campi MQMD/MQRFH2 al momento di get () o receive (). Tabella 118 a pagina 826 mostra come vengono associate le proprietà specifiche del provider JMS.

Tabella 116. Mappatura campo intestazione JMS del messaggio in entrata

Nome campo intestazione JMS	**Campo MQMD richiamato da**	**Campo MQRFH2 richiamato da**
JMSDestination		jms.Dst o mqps.Top"1" a pagina 825
JMSDeliveryMode	Durata"2" a pagina 825	jms.Dlv"2" a pagina 825
JMSExpiration		jms.Exp
JMSPriority	Priorit...	
JMSMessageID	MsgID	
JMSTimestamp	PutDate"2" a pagina 825 PutTime"2" a pagina 825	jms.Tms"2" a pagina 825
JMSCorrelationID	CorrelId"2" a pagina 825	jms.Cid"2" a pagina 825
JMSReplyTo	ReplyToQ"2" a pagina 825 ReplyToGestore code"2" a pagina 825	jms.Rto"2" a pagina 825
JMSType		mcd.Type, mcd.Set, mcd.Fmt
JMSRedelivered	BackoutCount	

Nota:

1. Se sono impostati sia jms.Dst che mqps.Top , viene utilizzato il valore in jms.Dst .
2. Per le proprietà che possono avere valori richiamati da MQRFH2 o da MQMD, se entrambi sono disponibili, viene utilizzata l'impostazione in MQRFH2 .
3. Il valore della proprietà JMS_IBM_Character_Set è un valore stringa che contiene l'equivalente della serie di caratteri Java per il valore numerico CodedCharacterSetId .

Tabella 117. Mappatura delle proprietà dei messaggi in entrata

Nome proprietà JMS	**Campo MQMD richiamato da**	**Campo MQRFH2 richiamato da**
JMSXUserID	UserIdentifier	
JMSXAppID	PutApplName	
JMSXDeliveryCount	BackoutCount	
JMSXGroupID	GroupId"1" a pagina 826	jms.Gid"1" a pagina 826
JMSXGroupSeq	MsgSeqNumero"1" a pagina 826	jms.Seq"1" a pagina 826

Tabella 117. Mappatura delle proprietà dei messaggi in entrata (Continua)

Nome proprietà JMS	Campo MQMD richiamato da	Campo MQRFH2 richiamato da

Nota:

1. Per le proprietà che possono avere valori richiamati da MQRFH2 o da MQMD, se entrambi sono disponibili, viene utilizzata l'impostazione in MQRFH2 . Le proprietà vengono impostate dai valori MQMD solo se sono impostati gli indicatori di messaggio MQMF_MSG_IN_GROUP o MQMF_LAST_MSG_IN_GROUP.

Tabella 118. Mappatura delle proprietà JMS specifiche del fornitore di messaggi in entrata

Nome proprietà JMS	Campo MQMD richiamato da	Campo MQRFH2 richiamato da
Eccezione JMS_IBM_Report_Exception	Prospetto	
Scadenza report IBM JMS	Prospetto	
COA report IBM JMS	Prospetto	
COD report IBM JMS	Prospetto	
PAN report IBM JMS	Prospetto	
NAN report IBM JMS	Prospetto	
JMS_IBM_Report_ Pass_Msg_ID	Prospetto	
ID correlazione password report IBM JMS	Prospetto	
JMS_IBM_Report_Discard_Msg	Prospetto	
JMS_IBM_MsgType	MsgType	
Feedback IBM JMS	Feedback	
Formato_IBM_JMS	Formato	
JMS_IBM_PutApplTipo	PutApplType	
Codifica IBM JMS "1" a pagina 826	Codifica	
Serie di caratteristiche IBM JMS "1" a pagina 826	CodedCharacterSetId	
JMS_IBM_PutDate	PutDate	
JMS_IBM_PutTime	PutTime	
JMS_IBM_Last_Msg_In_Group	MsgFlags	

1. Impostato solo se il messaggio in entrata è un messaggio in byte.

Scambio di messaggi tra un'applicazione JMS e un'applicazione WebSphere MQ tradizionale
Questo argomento descrive cosa accade quando un'applicazione JMS scambia i messaggi con un'applicazione tradizionale WebSphere MQ che non può elaborare l'intestazione MQRFH2 .

. Figura 129 a pagina 827 mostra l'associazione.

L'amministratore indica che l'applicazione JMS sta comunicando con un'applicazione WebSphere MQ tradizionale impostando la proprietà TARGCLIENT della destinazione su *MQ*. Ciò indica che non deve

essere prodotta alcuna intestazione MQRFH2 . Se questa operazione non viene eseguita, l'applicazione ricevente deve essere in grado di gestire l'intestazione MQRFH2 .

L'associazione da JMS a MQMD indirizzata a un'applicazione WebSphere MQ tradizionale è uguale all'associazione da JMS a MQMD indirizzata a un'applicazione JMS. Se WebSphere MQ classes per JMS riceve un messaggio WebSphere MQ con il campo MQMD *Format* impostato su un valore diverso da MQFMT_RFH2, i dati vengono ricevuti da un'applicazione non - JMS. Se il formato è MQFMT_STRING, il messaggio viene ricevuto come messaggio di testo JMS. Altrimenti, viene ricevuto come messaggio di byte JMS. Poiché non esiste alcun MQRFH2, è possibile ripristinare solo le proprietà JMS trasmesse in MQMD.

Se WebSphere MQ classes per JMS riceve un messaggio che non dispone di un'intestazione MQRFH2 , la proprietà TARGCLIENT dell'oggetto Queue o Topic derivato dal campo di intestazione JMSReplyTo del messaggio è impostata su MQ per impostazione predefinita. Ciò significa che anche un messaggio di risposta inviato alla coda o all'argomento non dispone di un'intestazione MQRFH2 . È possibile disattivare questo comportamento includendo un'intestazione MQRFH2 in un messaggio di risposta solo se il messaggio originale ha un'intestazione MQRFH2 , impostando la proprietà TARGCLIENTMATCHING del factory di connessione su NO.

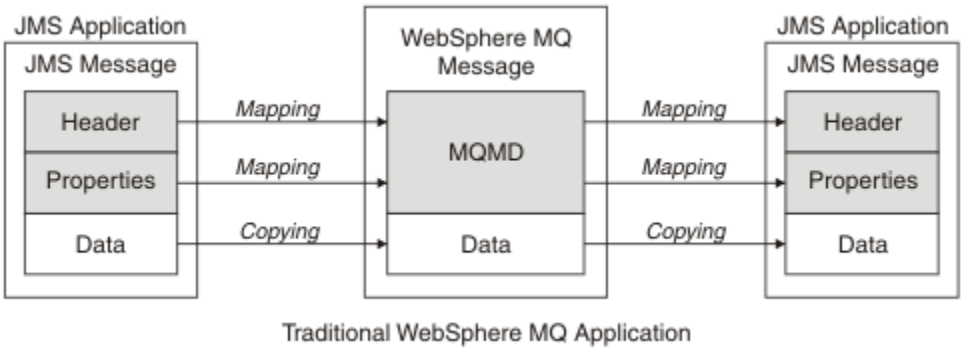


Figura 129. Modalità di trasformazione dei messaggi JMS in messaggi WebSphere MQ senza intestazione MQRFH2

Il contenuto del messaggio JMS
Questo argomento contiene informazioni sulla codifica del corpo del messaggio stesso. La codifica dipende dal tipo di messaggio JMS.

ObjectMessage
Un ObjectMessage è un oggetto serializzato da Java Runtime nel modo normale.

TextMessage
Un TextMessage è una stringa codificata. Per un messaggio in uscita, la stringa è codificata nella serie di caratteri fornita dall'oggetto di destinazione. Il valore predefinito è la codifica UTF8 (la codificazione UTF8 inizia con il primo carattere del messaggio; non esiste alcun campo di lunghezza all'inizio). È tuttavia possibile specificare qualsiasi altra serie di caratteri supportata dalle classi WebSphere MQ per JMS. Tali serie di caratteri vengono utilizzate principalmente quando si invia un messaggio a un'applicazione non - JMS.

Se la serie di caratteri è una serie a doppio byte (incluso UTF16), la specifica di codifica del numero intero dell'oggetto di destinazione determina l'ordine dei byte.

Un messaggio in entrata viene interpretato utilizzando la serie di caratteri e la codifica specificati nel messaggio stesso. Tali specifiche si trovano nell'ultima intestazione WebSphere MQ (o MQMD se non vi sono intestazioni). Per i messaggi JMS, l'ultima intestazione è generalmente MQRFH2.

BytesMessage
Un BytesMessage è, per impostazione predefinita, una sequenza di byte definita dalla specifica JMS 1.0.2 e dalla documentazione Java associata.

Per un messaggio in uscita assemblato dall'applicazione stessa, la proprietà di codifica dell'oggetto di destinazione può essere utilizzata per sovrascrivere le codifiche dei campi integer e floating point

contenuti nel messaggio. Ad esempio, è possibile richiedere che i valori a virgola mobile siano memorizzati in formato S/390 anziché IEEE).

Un messaggio in entrata viene interpretato utilizzando la codifica numerica specificata nel messaggio stesso. Questa specifica si trova nell'ultima intestazione WebSphere MQ (o MQMD se non vi sono intestazioni). Per i messaggi JMS, l'ultima intestazione è generalmente MQRFH2.

Se viene ricevuto un BytesMessage e viene inviato nuovamente senza modifiche, il relativo corpo viene trasmesso byte per byte, come è stato ricevuto. La proprietà di codifica dell'oggetto di destinazione non ha alcun effetto sul corpo. L'unica entità di tipo stringa che può essere inviata esplicitamente in un BytesMessage è una stringa UTF8 . Viene codificato in formato Java UTF8 e inizia con un campo di lunghezza di 2 byte. La proprietà della serie di caratteri dell'oggetto di destinazione non ha alcun effetto sulla codifica di un BytesMessagein uscita. Il valore della serie di caratteri in un messaggio WebSphere MQ in entrata non ha alcun effetto sull'interpretazione di tale messaggio come BytesMessageJMS.

Le applicazioni non Java non sono in grado di riconoscere la codifica Java UTF8 . Pertanto, affinché un'applicazione JMS invii un BytesMessage che contiene dati di testo, l'applicazione stessa deve convertire le sue stringhe in array di byte e scrivere tali array di byte in BytesMessage.

MapMessage
Un MapMessage è una stringa contenente triplette nome / tipo/valore XML codificate come:

```
<map>
  <elt name="elementname1" dt="datatype1">value1</elt>
  <elt name="elementname2" dt="datatype2">value2</elt>
  ...
</map>
```

dove `datatype` è uno dei tipi di dati elencati in Tabella 109 a pagina 817. Il tipo di dati predefinito è `string`e quindi l'attributo `dt="string"` viene omesso per gli elementi stringa.

La serie di caratteri utilizzata per codificare o interpretare la stringa XML che forma il corpo di un messaggio di associazione è determinata in base alle regole che si applicano a un messaggio di testo.

Le versioni di WebSphere MQ classes for JMS precedenti alla versione 5.3 codificano il contenuto di un messaggio di associazione nel seguente formato:

```
<map>
  <elementname1 dt="datatype1">value1</elementname1>
  <elementname2 dt="datatype2">value2</elementname2>
  ...
</map>
```

La versione 5.3 e le versioni successive di WebSphere MQ classes for JMS possono interpretare entrambi i formati, ma le versioni di WebSphere MQ classes for JMS precedenti alla versione 5.3 non possono interpretare il formato corrente.

Se un'applicazione deve inviare messaggi di associazione a un'altra applicazione che utilizza una versione di WebSphere MQ classes per JMS precedente alla Versione 5.3, l'applicazione mittente deve richiamare il metodo factory di connessione `setMapNameStyle(WMQConstants.WMQ_MAP_NAME_STYLE_COMPATIBLE)` per specificare che i messaggi di associazione vengono inviati nel formato precedente. Per impostazione predefinita, tutti i messaggi di associazione vengono inviati nel formato corrente.

StreamMessage
Un StreamMessage è come un messaggio di associazione, ma senza nomi elemento:

```
<stream>
  <elt dt="datatype1">value1</elt>
  <elt dt="datatype2">value2</elt>
  ...
</stream>
```

dove `datatype` è uno dei tipi di dati elencati in Tabella 109 a pagina 817. Il tipo di dati predefinito è `string`e quindi l'attributo `dt="string"` viene omesso per gli elementi stringa.

La serie di caratteri utilizzata per codificare o interpretare la stringa XML che costituisce il corpo di StreamMessage viene determinata in base alle regole che si applicano a TextMessage.

Il campo MQRFH2.format è impostato come segue:

MQFMT_NONE
per ObjectMessage, BytesMessageo messaggi senza corpo.

MQFMT_STRING
per TextMessage, StreamMessageo MapMessage.

Conversione messaggio JMS

La conversione dei dati del messaggio in JMS viene eseguita durante l'invio e la ricezione dei messaggi. WebSphere MQ esegue automaticamente la maggior parte della conversione dei dati. Converte il testo e i dati numerici quando si trasferisce un messaggio tra le applicazioni JMS. Il testo viene convertito quando si scambiano `JMSTextMessage` tra un'applicazione JMS e un'applicazione WebSphere MQ .

Se si sta pianificando di effettuare scambi di messaggi più complessi, i seguenti argomenti sono di interesse. Gli scambi di messaggi complessi includono:

- Trasferimento di messaggi non di testo tra un'applicazione WebSphere MQ e un'applicazione JMS.
- Scambio di dati di testo in formato byte.
- Conversione del testo nella tua applicazione.

Dati messaggio JMS

La conversione dei dati è necessaria per scambiare dati di testo e numerici tra le applicazioni, anche tra due applicazioni JMS. La rappresentazione interna del testo e dei numeri deve essere codificata in modo che possano essere trasferiti in un messaggio. La codifica forza una decisione su come vengono rappresentati i numeri e il testo. WebSphere MQ gestisce la codifica di testo e numeri nei messaggi JMS, ad eccezione di `JMSObjectMessage`, consultare “JMSObjectMessage” a pagina 836. Utilizza tre attributi del messaggio. I tre attributi sono `CodedCharacterSetId`, `Encoding`e `Format`.

Questi tre attributi del messaggio sono normalmente memorizzati nei campi dell'intestazione JMS, `MQRFH2`, di un messaggio JMS. Se il tipo di messaggio è un tipo di messaggio `MQ`, piuttosto che `JMS` , gli attributi vengono memorizzati nel descrittore del messaggio, `MQMD`. Gli attributi vengono utilizzati per convertire i dati del messaggio JMS. I dati del messaggio di JMS vengono trasferiti nella parte dei dati del messaggio di un messaggio di WebSphere MQ .

Proprietà messaggio JMS

Le proprietà del messaggio JMS, come `JMS_IBM_CHARACTER_SET`, vengono scambiate nella parte di intestazione `MQRFH2` di un messaggio JMS, a meno che il messaggio non sia stato inviato senza un `MQRFH2`. Solo `JMSTextMessage` e `JMSBytesMessage` possono essere inviati senza `MQRFH2`. Se una proprietà JMS viene memorizzata come una proprietà del messaggio WebSphere MQ nel descrittore del messaggio, `MQMD`, viene convertita come parte della conversione `MQMD` . Se una proprietà JMS è memorizzata in `MQRFH2`, viene memorizzata nella serie di caratteri specificati da `MQRFH2.NameValueCCSID`. Quando un messaggio viene inviato o ricevuto, le proprietà del messaggio vengono convertite nella relativa rappresentazione interna nella JVM. La conversione è da e verso la serie di caratteri del descrittore del messaggio o `MQRFH2.NameValueCCSID`. I dati numerici vengono convertiti in testo.

Conversione messaggio JMS

I seguenti argomenti contengono esempi e attività utili se si prevede di scambiare messaggi più complessi che richiedono la conversione.

Approcci di conversione dei messaggi JMS
Alcuni approcci di conversione dati sono aperti ai progettisti di applicazioni JMS. Questi approcci non sono esclusivi; alcune applicazioni probabilmente utilizzeranno una combinazione di questi approcci. Se

l'applicazione sta scambiando solo testo o messaggi solo con altre applicazioni JMS, normalmente non si considera la conversione dei dati. La conversione dei dati viene eseguita automaticamente da WebSphere WebSphere MQ.

È possibile porre una serie di domande su come avvicinarsi alla conversione dei messaggi:

È necessario pensare alla conversione dei messaggi?
In alcuni casi, come ad esempio i trasferimenti di messaggi da JMS a JMS e lo scambio di messaggi di testo con i programmi IBM WebSphere MQ , IBM WebSphere MQ esegue automaticamente le conversioni necessarie. È possibile che si desideri controllare la conversione dei dati per motivi di prestazioni o che si stia scambiando messaggi complessi con un formato predefinito. In casi come questi, è necessario comprendere la conversione dei messaggi e leggere i seguenti argomenti.

Che tipo di conversione ci sono?
Ci sono quattro tipi principali di conversione, che sono spiegati nelle sezioni seguenti:

1. “Conversione dati client JMS” a pagina 830
2. “Conversione dati applicazione” a pagina 831
3. “Conversione dati del gestore code” a pagina 831
4. “Conversione dati del canale dei messaggi” a pagina 832

Dove deve essere eseguita la conversione?
La sezione, “Scelta di un approccio alla conversione del messaggio: il ricevitore è corretto” a pagina 832, descrive il solito approccio di "ricezione". "Il destinatario fa bene" si applica anche alla conversione dati JMS.

Conversione dati client JMS

Client JMS[4]la conversione dei dati è la conversione di oggetti e primitive Java in byte in un messaggio JMS quando viene inviato a una destinazione e di nuovo la conversione quando viene ricevuto. La conversione dei dati del client JMS utilizza i metodi delle classi `JMSMessage` . I metodi sono elencati per il tipo di classe `JMSMessage` in Tabella 119 a pagina 833 .

La conversione da e verso la rappresentazione JVM interna di numeri e testo viene eseguita per i metodi read, get, set e write. La conversione viene eseguita quando il messaggio viene inviato e quando uno dei metodi read o get viene richiamato su un messaggio ricevuto.

La codepage e la codifica numerica utilizzate per scrivere o impostare il contenuto di un messaggio sono definite come attributi della destinazione. La codepage di destinazione e la codifica numerica possono essere modificate in modo amministrativo. Un'applicazione può anche sovrascrivere la codepage di destinazione e la codifica impostando le proprietà del messaggio che controllano la scrittura o l'impostazione del contenuto del messaggio.

Se si desidera convertire la codifica dei numeri quando un messaggio `JMSBytesMessage` viene inviato a una destinazione non definita come `Native` codifica, è necessario impostare la proprietà del messaggio `JMS_IBM_ENCODING` prima di inviare il messaggio. Se si sta seguendo il modello "il destinatario è corretto" o se si stanno scambiando messaggi tra applicazioni JMS, l'applicazione non deve impostare `JMS_IBM_ENCODING`. Nella maggior parte dei casi, è possibile lasciare la proprietà `Encoding` come `Native`.

Per i messaggi `JMSStreamMessage`, `JMSMapMessage`e `JMSTextMessage` , vengono utilizzate le proprietà dell'identificativo della serie di caratteri della destinazione. La codifica viene ignorata durante l'invio poiché i numeri vengono scritti in formato testo. Il programma di applicazione client JMS non deve impostare `JMS_IBM_CHARACTER_SET` prima di inviare il messaggio se la proprietà della serie di caratteri di destinazione da applicare.

Per ottenere i dati in un messaggio, un'applicazione richiama i metodi read o get del messaggio JMS. I metodi fanno riferimento alla codepage e alla codifica definiti nell'intestazione del messaggio precedente per creare correttamente gli oggetti e le primitive Java.

[4] "Client JMS" si riferisce alle classi WebSphere MQ per JMS che implementano l'interfaccia JMS, che viene eseguita in modalità client o bind.

La conversione dei dati del client JMS soddisfa le esigenze della maggior parte delle applicazioni JMS che scambiano messaggi tra un client JMS e l'altro. Non si codifica alcuna conversione dati esplicita. Non si utilizza la classe `java.nio.charset.Charset` , generalmente utilizzata quando si scrive il testo in un file. I metodi `writeString` e `setString` fanno la conversione per conto dell'utente.

Per ulteriori dettagli sulla conversione dati del client JMS, consultare “Conversione e codifica dei messaggi del client JMS” a pagina 843.

Conversione dati applicazione

Un'applicazione client JMS può eseguire la conversione dei dati carattere espliciti utilizzando la classe `java.nio.charset.Charset` ; consultare gli esempi in Figura 132 a pagina 835 e Figura 133 a pagina 835 . I dati stringa vengono convertiti in byte, utilizzando il metodo `getBytes` e inviati come byte. I byte vengono riconvertiti in testo utilizzando un costruttore `String` che utilizza un array di byte e un `Charset`. I dati carattere vengono convertiti utilizzando i metodi `encode` e `decode` `Charset` . Generalmente il messaggio viene inviato o ricevuto come `JMSBytesMessage`, poiché la parte del messaggio di un `JMSBytesMessage` non contiene altro che i dati scritti dall'applicazione[5]. È anche possibile inviare e ricevere byte utilizzando `JMSStreamMessage`, `JMSMapMessage`o `JMSObjectMessage`.

Non esistono metodi Java per codificare e decodificare byte che contengono dati numerici rappresentati in formati di codifica differenti. I dati numerici vengono codificati e decodificati automaticamente utilizzando i metodi di lettura e scrittura `JMSMessage` numerici. I metodi di lettura e scrittura utilizzano il valore dell'attributo `JMS_IBM_ENCODING` dei dati del messaggio.

Un uso tipico per la conversione dei dati dell'applicazione è se un client JMS invia o riceve un messaggio formattato da un'applicazione non JMS. Un messaggio formattato contiene dati di testo, numerici e byte organizzati in base alla lunghezza dei campi di dati. A meno che l'applicazione non JMS non abbia specificato il formato del messaggio come "MQSTR ", il messaggio viene creato come `JMSBytesMessage`. Per ricevere i dati dei messaggi formattati in `JMSBytesMessage` è necessario richiamare una sequenza di metodi. I metodi devono essere richiamati nello stesso ordine in cui i campi sono stati scritti nel messaggio. Se i campi sono numerici, è necessario conoscere la codifica e la lunghezza dei dati numerici. Se uno dei campi contiene dati di byte o di testo, è necessario conoscere la lunghezza dei dati di byte nel messaggio. Esistono due modi per convertire un messaggio formattato in un oggetto Java facile da utilizzare.

1. Creare una classe Java corrispondente al record, per incapsulare la lettura e la scrittura del messaggio. L'accesso ai dati nel record è con i metodi get e set della classe.
2. Creare una classe Java corrispondente al record estendendo la classe `com.ibm.mq.headers` . L'accesso ai dati nella classe è con gli accessor specifici del tipo del modulo, `getStringValue(`*`fieldname`*`);`

Consultare “Scambio di un record formattato con un'applicazione non JMS” a pagina 850.

Conversione dati del gestore code

In WebSphere MQ V7.0, la conversione della codepage può essere eseguita dal gestore code quando un programma client JMS riceve un messaggio. La conversione è uguale alla conversione eseguita per un programma C. Un programma C imposta `MQGMO_CONVERT` come opzione di parametro `MQGET` `GetMsgOpts` ; consultare Figura 131 a pagina 835. Un gestore code esegue la conversione per un programma client JMS che sta ricevendo un messaggio, se la proprietà di destinazione `WMQ_RECEIVE_CONVERSION` è impostata su `WMQ_RECEIVE_CONVERSION_QMGR`, anche il programma client JMS può impostare la proprietà di destinazione; consultare Figura 130 a pagina 832.

Prima di V7.0, le conversione venivano sempre eseguite dal client JMS. La conversione dei dati del client JMS è limitata alla conversione di sequenze di numeri e testo di tipo e lunghezza noti al client JMS. Non può convertire strutture dati; consultare “Scambio di un record formattato con un'applicazione non JMS” a pagina 850. In V7.0, fino al fix pack 7.0.1.5, se la conversione può essere eseguita dal gestore code, viene sempre eseguita dal gestore code. A partire da 7.0.1.5 in poi, il comportamento

[5] Un'eccezione: i dati scritti utilizzando `writeUTF` iniziano con un campo di lunghezza di 2 byte

di conversione predefinito ritorna allo stesso valore di V6.0e tutte le conversioni vengono eseguite dal client JMS. Da 7.0.1.5, o 7.0.1.4 con APAR IC72897, è possibile impostare una nuova opzione di destinazione, `WMQ_RECEIVE_CONVERSION`, per controllare dove viene eseguita la conversione e `WMQ_RECEIVE_CCSID`, per impostare la codepage di destinazione; consultare Figura 130 a pagina 832.

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Oppure,

```
((MQDestination)destination).setReceiveConversion
             (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Figura 130. Abilita conversione dati gestore code

Il vantaggio principale della conversione del gestore code si ha quando si scambiano messaggi con applicazioni non JMS. Se il campo `Format` nel messaggio è definito e la serie di caratteri di destinazione, o la codifica, è diversa dal messaggio, il gestore code esegue la conversione dei dati per l'applicazione di destinazione, se l'applicazione lo richiede. Il gestore code converte i dati del messaggio formattati in base a uno dei tipi di messaggio predefiniti di WebSphere MQ , ad esempio un'intestazione bridge CICS (`MQCIH`). Se il campo `Format` è definito dall'utente, il gestore code cerca un'uscita di conversione dati con il nome fornito nel campo `Format` .

La conversione dei dati del gestore code viene utilizzata al meglio con il modello di progettazione "Il destinatario fa bene" . Un client JMS di invio non deve eseguire la conversione. Un programma di ricezione non JMS si basa sull'uscita di conversione per garantire che il messaggio venga consegnato nella codepage e nella codifica richieste. Con un client JMS di invio e un destinatario non JMS, l'esempio si applica a IBM WebSphere MQ pre-e post-V7.0. Con IBM WebSphere MQ V7.0, è possibile richiamare l'exit di conversione anche per un programma di ricezione JMS.

È possibile creare un'uscita di conversione dati utilizzando il programma di utilità di uscita di conversione dati, **crtmqcvx**, per consentire al gestore code di convertire i propri dati formattati del record. È possibile creare un formato record personalizzato, utilizzare `com.ibm.mq.headers` per accedervi come classe Java e utilizzare la propria uscita di conversione per convertirlo. Su z/OS , il programma di utilità è denominato **CSQUCVX**e su IBM i, **CVTMQMDTA**. Consultare “Scambio di un record formattato con un'applicazione non JMS” a pagina 850.

Conversione dati del canale dei messaggi

WebSphere MQ I canali mittente, server, ricevente del cluster e mittente del cluster hanno un'opzione di conversione dei messaggi, `CONVERT`. Il contenuto di un messaggio può essere facoltativamente convertito quando viene inviato un messaggio. La conversione avviene all'estremità di invio del canale. La definizione ricevente del cluster viene utilizzata per definire automaticamente il corrispondente canale mittente del cluster.

La conversione dei dati da parte dei canali dei messaggi viene generalmente utilizzata se non è possibile utilizzare altre forme di conversione.

Scelta di un approccio alla conversione del messaggio: "il ricevitore è corretto"

L'approccio abituale nella progettazione dell'applicazione WebSphere MQ per la conversione del codice è "receiver rende bene". "Destinatario corretto" riduce il numero di conversioni di messaggi. Evita inoltre il problema di errori di canale imprevisti se la conversione del messaggio non riesce su un gestore code intermediario durante il trasferimento del messaggio. La regola "receiver make good" viene interrotta solo se esiste un motivo per cui il ricevitore non può essere corretto. Ad esempio, la piattaforma ricevente potrebbe non avere la serie di caratteri corretta.

"Il ricevitore fa bene" è anche una buona guida generale per applicazioni client JMS. Ma in casi specifici, la conversione al set di caratteri corretto all'origine può essere più efficiente. La conversione dalla rappresentazione interna JVM deve essere eseguita quando viene inviato un messaggio contenente tipi di testo o numerici. La conversione alla serie di caratteri richiesta dal destinatario, se il ricevente non è un client JMS, potrebbe eliminare la necessità per il ricevente non JMS di eseguire la conversione. Se il destinatario è un client JMS, lo convertirà di nuovo, comunque, per decodificare i dati del messaggio e creare oggetti e primitive Java.

La differenza tra le applicazioni client JMS e le applicazioni scritte in un linguaggio come C, è che Java deve eseguire la conversione dei dati. Un'applicazione Java deve convertire numeri e testo dalla loro rappresentazione interna in un formato codificato utilizzato nei messaggi.

Impostando la destinazione o le proprietà del messaggio, è possibile impostare la serie di caratteri e la codifica utilizzata da WebSphere MQ per codificare i numeri e il testo nei messaggi. Normalmente, la serie di caratteri viene lasciata come `1208` e la codifica come `Native`.

WebSphere MQ non converte gli array di byte. Per codificare stringhe e array di caratteri in array di byte, utilizzare il pacchetto `java.nio.charset` . `Charset` specifica la serie di caratteri utilizzata per convertire una stringa o una schiera di caratteri in una schiera di byte. È anche possibile decodificare una schiera di byte in una stringa o una schiera di caratteri utilizzando un `Charset`. Non è consigliabile affidarsi a `java.nio.charset.Charset.defaultCodePage` quando si codificano stringhe e array di caratteri. Il valore predefinito `Charset` è generalmente `windows-1252` in Windowse `UTF-8` in UNIX. `windows-1252` è una serie di caratteri a singolo byte e `UTF-8` è una serie di caratteri a più byte.

Generalmente, lasciare la serie di caratteri di destinazione e le proprietà di codifica ai valori predefiniti di `UTF-8` e `Native` quando si scambiano messaggi con altre applicazioni JMS. Se si stanno scambiando messaggi contenenti numeri o testo con un'applicazione JMS, scegliere uno dei tipi di messaggi `JMSTextMessage`, `JMSStreamMessage`, `JMSMapMessage`o `JMSObjectMessage` che si adattano al proprio scopo. Non ci sono altre attività di conversione da eseguire.

Se si scambiano messaggi con applicazioni non JMS che utilizzano un formato record, è più complicato. A meno che l'intero record non contenga testo e non possa essere trasferito come `JMSTextMessage`, è necessario codificare e decodificare il testo nell'applicazione. Impostare il tipo di messaggio di destinazione su `MQ`e utilizzare `JMSBytesMessage` per evitare che le classi IBM WebSphere MQ per JMS aggiungano ulteriori informazioni di intestazione e tag ai dati del messaggio. Utilizzare i metodi `JMSBytesMessage` per scrivere numeri e byte e la classe `Charset` converte esplicitamente il testo in schiere di byte. Una serie di fattori potrebbe influenzare la scelta della serie di caratteri:

- Prestazioni: È possibile ridurre il numero di conversioni trasformando il testo in un set di caratteri utilizzato sul maggior numero di server?
- Uniformità: trasferire tutti i messaggi nella stessa serie di caratteri.
- Ricchezza: Quali set di caratteri hanno tutti i punti di codice che le applicazioni devono utilizzare?
- Semplicità: le serie di caratteri a byte singolo sono più semplici da utilizzare rispetto alle serie di caratteri a lunghezza variabile e multibyte.

Consultare “Scambio di un record formattato con un'applicazione non JMS” a pagina 850. per esempi di conversione di messaggi scambiati con applicazioni non JMS.

Esempi

Tabella dei tipi di messaggio e dei tipi di conversione

Tabella 119. Tipi di messaggi e tipi di conversione

	Tipo di conversione			
Tipo messaggio	**Testo**	**Numerico**	**Altro**	**Nessuno**
`JMSObjectMessage`				`getObject` `setObject`

Tabella 119. Tipi di messaggi e tipi di conversione (Continua)

	Tipo di conversione			
Tipo messaggio	**Testo**	**Numerico**	**Altro**	**Nessuno**
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

Richiamo della conversione dati da un programma C

```
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                     */
```

Figura 131. Frammento di codice da amqsget0.c

Invio e ricezione di testo in un JMSBytesMessage

Il codice in Figura 132 a pagina 835 invia una stringa in un BytesMessage. Per semplicità, l'esempio invia una singola stringa, per cui un JMSTextMessage è più appropriato. Per ricevere un messaggio di testo in byte contenente una combinazione di tipi, è necessario conoscere la lunghezza della stringa in byte, denominata *TEXT_LENGTH* in Figura 133 a pagina 835. Anche per una stringa con un numero fisso di caratteri, la lunghezza della rappresentazione in byte potrebbe essere maggiore.

```
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);
```

Figura 132. Invio di un String in un 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);
```

Figura 133. Ricezione di un String da un JMSBytesMessage

Concetti correlati

Conversione e codifica dei messaggi del client JMS
Vengono elencati i metodi utilizzati per eseguire la conversione e la codifica dei messaggi del client JMS, con esempi di codice di ciascun tipo di conversione.

Conversione dati del gestore code
La conversione dei dati del gestore code è sempre stata disponibile per le applicazioni non - JMS che ricevono i messaggi dai client JMS. A partire da V7.0, i client JMS che ricevono messaggi utilizzano anche la conversione dei dati del gestore code. Da 7.0.1.5o 7.0.1.4 con APAR IC72897, la conversione dei dati del gestore code è facoltativa.

Attività correlate

Scambio di un record formattato con un'applicazione non JMS
Seguire la procedura suggerita in questa attività per progettare e creare un'uscita di conversione dati e un'applicazione client JMS che può scambiare messaggi con un'applicazione non JMS utilizzando JMSBytesMessage. Lo scambio di un messaggio formattato con un'applicazione non JMS può avvenire con o senza richiamare un'uscita di conversione dati.

Riferimenti correlati

Tipi di messaggio JMS e conversione
La scelta del tipo di messaggio influisce sull'approccio alla conversione del messaggio. L'interazione tra la conversione del messaggio e il tipo di messaggio è descritta per i tipi di messaggio JMS JMSObjectMessage, JMSTextMessage, JMSMapMessage JMSStreamMessagee JMSBytesMessage.

Tipi di messaggio JMS e conversione
La scelta del tipo di messaggio influisce sull'approccio alla conversione del messaggio. L'interazione tra la conversione del messaggio e il tipo di messaggio è descritta per i tipi di messaggio JMS JMSObjectMessage, JMSTextMessage, JMSMapMessage JMSStreamMessagee JMSBytesMessage.

JMSObjectMessage

JMSObjectMessage contiene un oggetto e tutti gli oggetti a cui fa riferimento, serializzati in un flusso di byte dalla JVM. Il testo viene serializzato in UTF-8e limitato a stringhe o schiere di caratteri di non più di 65534 byte. Un vantaggio di JMSObjectMessage è che le applicazioni non sono coinvolte in alcun problema di conversione dati purché utilizzino solo i metodi e gli attributi dell'oggetto. JMSObjectMessage fornisce la conversione dei dati per oggetti complessi senza che il programmatore dell'applicazione consideri come codificare un oggetto in un messaggio. Lo svantaggio dell'utilizzo di JMSObjectMessage è che può essere scambiato solo con altre applicazioni JMS. Scegliendo uno degli altri tipi di messaggi JMS, è possibile scambiare messaggi JMS con applicazioni non JMS.

"Invio e ricezione di un JMSObjectMessage" a pagina 839 mostra un String oggetto scambiato in un messaggio.

Un'applicazione client JMS può ricevere un JMSObjectMessage solo in un messaggio con un corpo in stile JMS. La destinazione deve specificare un corpo stile JMS.

JMSTextMessage

JMSTextMessage contiene una singola stringa di testo. Quando viene inviato un messaggio di testo, il testo Format è impostato su "MQSTR ", WMQConstants.MQFMT_STRING. Il CodedCharacterSetId del testo è impostato sull'identificativo della serie di caratteri codificati definito per la destinazione. Il testo viene codificato in CodedCharacterSetId da WebSphere MQ. I campi CodedCharacterSetId e Format sono impostati nel descrittore del messaggio, MQMD, o nei campi JMS in un MQRFH2. Se il messaggio è definito come avente uno stile del corpo del messaggio WMQ_MESSAGE_BODY_MQ o lo stile del contenuto non è specificato, ma la destinazione è WMQ_TARGET_DEST_MQ, vengono impostati i campi del descrittore del messaggio. In alternativa, il messaggio ha un RFH2 JMS e i campi sono impostati nella parte fissa di MQRFH2.

Un'applicazione può sostituire il CCSID (coded character set identifier) definito per una destinazione. È necessario impostare la proprietà del messaggio JMS_IBM_CHARACTER_SET su un CCSID (coded character set identifier); consultare l'esempio in "Invio e ricezione di un JMSTextmessage" a pagina 839.

Quando il client JMS richiama la conversione del gestore code del metodo consumer.receive è facoltativa. La conversione del gestore code è abilitata impostando la proprietà di destinazione WMQ_RECEIVE_CONVERSION su WMQ_RECEIVE_CONVERSION_QMGR. Il gestore code converte il messaggio di testo dal JMS_IBM_CHARACTER_SET specificato per il messaggio prima di trasferirlo al client JMS. La serie di caratteri del messaggio convertito è 1208, UTF-8, a meno che la destinazione non abbia un WMQ_RECEIVE_CCSIDdiverso. Il CodedCharacterSetId nel messaggio che fa riferimento a JMSTextMessage viene aggiornato all'ID della serie di caratteri di destinazione. Il testo viene decodificato dalla serie di caratteri di destinazione in Unicode mediante il metodo getText ; consultare l'esempio in "Invio e ricezione di un JMSTextmessage" a pagina 839.

Un JMSTextMessage può essere inviato in un corpo di messaggio in stile MQ, senza un'intestazione JMS MQRFH2 . Il valore degli attributi di destinazione, WMQ_MESSAGE_BODY e WMQ_TARGET_DEST determinano lo stile del messaggio, a meno che non venga sovrascritto dall'applicazione. L'applicazione può sovrascrivere i valori impostati sulla destinazione richiamando destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ) o destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ).

Se si invia un JMSTextMessage con un corpo di stile MQ inviandolo a una destinazione con WMQ_MESSAGE_BODY impostato su WMQ_MESSAGE_BODY_MQ, non è possibile riceverlo come JMSTextMessage dalla stessa destinazione. Tutti i messaggi ricevuti da una destinazione con WMQ_MESSAGE_BODY impostato su WMQ_MESSAGE_BODY_MQ vengono ricevuti come JMSBytesMessage. Se si tenta di ricevere il messaggio come JMSTextMessage , viene generata un'eccezione, ClassCastException: com.ibm.jms.JMSBytesMessage cannot be cast to javax.jms.TextMessage.

Nota: Il testo in JMSBytesMessage non viene convertito dal client JMS. Il client può ricevere solo il testo nel messaggio come schiera di byte. Se la conversione del gestore code è abilitata, il testo viene convertito dal gestore code, ma il client JMS deve ancora riceverlo come array di byte in un JMSBytesMessage.

È generalmente preferibile utilizzare la proprietà WMQ_TARGET_DEST per controllare se un JMSTextMessage viene inviato con uno stile del corpo MQ o JMS. È quindi possibile ricevere il messaggio da una destinazione che ha WMQ_TARGET_DEST impostato su WMQ_TARGET_DEST_MQ o WMQ_TARGET_DEST_JMS. WMQ_TARGET_DEST non ha effetto sul ricevitore.

JMSMapMessage e JMSStreamMessage

Questi due messaggi JMS sono simili. È possibile leggere e scrivere tipi primitivi nei messaggi utilizzando metodi basati sulle interfacce DataInputStream e DataOutputStream ; consultare “Tabella dei tipi di messaggio e dei tipi di conversione” a pagina 841. I dettagli sono descritti in “Conversione e codifica dei messaggi del client JMS” a pagina 843. Ogni primitiva è contrassegnata; consultare “Il contenuto del messaggio JMS” a pagina 827.

I dati numerici vengono letti e scritti nel messaggio codificato come testo XML. Non viene fatto riferimento alla proprietà di destinazione, JMS_IBM_ENCODING. I dati di testo vengono trattati come il testo in un JMSTextMessage. Se si dovesse esaminare il contenuto del messaggio creato dall'esempio in Figura 138 a pagina 840, tutti i dati del messaggio sarebbero in EBCDIC come sono stati inviati con un valore della serie di caratteri 37.

È possibile inviare più elementi in un JMSMapMessage o JMSStreamMessage.

È possibile richiamare i singoli elementi di dati per nome da un JMSMapMessageo per posizione da un JMSStreamMessage. Ogni elemento viene decodificato quando viene richiamato un metodo get o read utilizzando il valore di CodedCharacterSetId memorizzato nel messaggio. Se il metodo utilizzato per richiamare l'elemento restituisce un tipo diverso dal tipo inviato, il tipo viene convertito. Se il tipo non può essere convertito, viene generata un'eccezione. Consultare Classe JMSStreamMessage per i dettagli. L'esempio riportato in “Invio di dati in un JMSStreamMessage e JMSMapMessage” a pagina 840 illustra la conversione del tipo e il richiamo del contenuto JMSMapMessage fuori sequenza.

Il campo MQRFH2.format per JMSMapMessage e JMSStreamMessage è impostato su "MQSTR ". Se la proprietà di destinazione WMQ_RECEIVE_CONVERSION è impostata su WMQ_RECEIVE_CONVERSION_QMGR, i dati del messaggio vengono convertiti dal gestore code prima di essere inviati al client JMS. MQRFH2.CodedCharacterSetId del messaggio è il WMQ_RECEIVE_CCSID della destinazione. MQRFH2.Encoding è Native. Se WMQ_RECEIVE_CONVERSION è WMQ_RECEIVE_CONVERSION_CLIENT_MSG il CodedCharacterSetId e Encoding di MQRFH2 è il valore impostato dal mittente.

Un'applicazione client JMS può ricevere un JMSMapMessage o un JMSStreamMessage solo in un messaggio che ha un corpo in stile JMS e da una destinazione che non specifica un corpo in stile MQ .

JMSBytesMessage

Un JMSBytesMessage può contenere più tipi primitivi. È possibile leggere e scrivere tipi primitivi nei messaggi utilizzando metodi basati sulle interfacce DataInputStream e DataOutputStream ; consultare "Tabella dei tipi di messaggio e dei tipi di conversione" a pagina 841. I dettagli sono descritti in "Tipi di messaggio JMS e conversione" a pagina 836.

La codifica dei dati numerici nel messaggio è controllata dal valore di JMS_IBM_ENCODING impostato prima della scrittura di dati numerici in JMSBytesMessage. Un'applicazione può sovrascrivere la codifica Native predefinita definita per JMSBytesMessage impostando la proprietà del messaggio JMS_IBM_ENCODING.

I dati di testo possono essere letti e scritti in UTF-8 utilizzando readUTF e writeUTFoppure in Unicode utilizzando i metodi readChar e writeChar . Non ci sono metodi che utilizzano CodedCharacterSetId. In alternativa, il client JMS può codificare e decodificare il testo in byte utilizzando la classe Charset . Trasferisce i byte tra la JVM e il messaggio senza che le classi WebSphere MQ per JMS eseguano alcuna conversione; consultare "Invio e ricezione di testo in un JMSBytesMessage" a pagina 840.

Un JMSBytesMessage inviato a un'applicazione MQ viene generalmente inviato in un corpo del messaggio in stile MQ, senza un'intestazione JMS MQRFH2 . Se viene inviato a un'applicazione JMS, lo stile del corpo del messaggio è in genere JMS. Il valore degli attributi di destinazione, WMQ_MESSAGE_BODY e WMQ_TARGET_DEST determinano lo stile del messaggio, a meno che non venga sovrascritto dall'applicazione. L'applicazione può sovrascrivere i valori impostati sulla destinazione richiamando destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ) o destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ).

Se si invia un JMSBytesMessage con un corpo in stile MQ , è possibile ricevere il messaggio da una destinazione che definisce uno stile del corpo del messaggio MQ o JMS. Se si invia un JMSBytesMessage con un corpo dello stile JMS, è necessario ricevere il messaggio da una destinazione che definisce uno stile del corpo del messaggio JMS. In caso contrario, il MQRFH2 viene considerato come parte dei dati del messaggio utente, il che potrebbe non essere quello previsto.

Se un messaggio ha uno stile di corpo MQ o JMS, il modo in cui viene ricevuto non è influenzato dall'impostazione WMQ_TARGET_DEST.

Il messaggio potrebbe essere trasformato in un secondo momento, dal gestore code, se viene fornito un Format per i dati del messaggio e la conversione dei dati del gestore code è abilitata. Non utilizzare il campo formato se non per specificare il formato dei dati del messaggio o lasciarlo vuoto, MQConstants.MQFMT_NONE

È possibile inviare più elementi in JMSBytesMessage. Ogni elemento numerico viene convertito quando il messaggio viene inviato utilizzando la codifica definita per il messaggio.

È possibile richiamare i singoli elementi di dati da JMSBytesMessage. Richiamare i metodi di lettura nello stesso ordine in cui vengono richiamati i metodi di scrittura per creare il messaggio. Ogni elemento numerico viene convertito quando il messaggio viene richiamato utilizzando il valore Encoding memorizzato nel messaggio.

A differenza di JMSMapMessage e JMSStreamMessage, JMSBytesMessage contiene solo dati scritti dall'applicazione. Non vengono memorizzati ulteriori dati nei dati del messaggio, come ad esempio i tag XML utilizzati per definire gli elementi in JMSMapMessage e JMSStreamMessage. Per questo motivo, utilizzare JMSBytesMessage per trasferire i messaggi formattati per altre applicazioni.

La conversione tra JMSBytesMessage e DataInputStream e DataOutputStream è utile in alcune applicazioni. Il codice basato sull'esempio, "Lettura e scrittura di messaggi utilizzando DataInputStream e DataOutputStream" a pagina 840, è necessario per utilizzare il package com.ibm.mq.header con JMS.

Esempi

Invio e ricezione di un 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
```

Figura 134. Invio e ricezione di un JMSObjectMessage

Invio e ricezione di un JMSTextmessage

Un messaggio di testo non può contenere testo in serie di caratteri differenti. L'esempio mostra il testo in diverse serie di caratteri, inviato in due diversi messaggi.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Figura 135. Invia messaggio di testo nella serie di caratteri definita dalla destinazione

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Figura 136. Invia messaggio di testo in ccsid 37

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Figura 137. Ricevi messaggio di testo

Invio di dati in un JMSStreamMessage e 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
```

Figura 138. Invio di dati in JMSStreamMessage e JMSMapMessage

Invio e ricezione di testo in un JMSBytesMessage

Il codice in Figura 139 a pagina 840 invia una stringa in un BytesMessage. Per semplicità, l'esempio invia una singola stringa, per cui un JMSTextMessage è più appropriato. Per ricevere un messaggio di testo in byte contenente una combinazione di tipi, è necessario conoscere la lunghezza della stringa in byte, denominata *TEXT_LENGTH* in Figura 140 a pagina 840. Anche per una stringa con un numero fisso di caratteri, la lunghezza della rappresentazione in byte potrebbe essere maggiore.

```
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);
```

Figura 139. Invio di un String in un 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);
```

Figura 140. Ricezione di un String da un JMSBytesMessage

Lettura e scrittura di messaggi utilizzando DataInputStream e DataOutputStream

Il codice in Figura 141 a pagina 841 crea un JMSBytesMessage utilizzando un 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);
```

Figura 141. Inviare un JMSBytesMessage utilizzando un DataOutputStream

L'istruzione che imposta la proprietà JMS_IBM_ENCODING è commentata. L'istruzione è valida se si scrive direttamente in un JMSBytesMessage, ma non ha alcun effetto quando si scrive in DataOutputStream. I numeri scritti in DataOutputStream sono codificati nella codifica Native . L'impostazione JMS_IBM_ENCODING non ha alcun effetto.

Il codice in Figura 142 a pagina 841 riceve un JMSBytesMessage utilizzando un 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)));
```

Figura 142. Ricevere un JMSBytesMessage utilizzando un DataInputStream

La codepage viene stampata utilizzando la proprietà codepage dei dati del messaggio di input, JMS_IBM_CHARACTER_SET. Sull'immissione JMS_IBM_CHARACTER_SET è una codepage Java e non un CCSID (coded character set identifier) numerico.

Tabella dei tipi di messaggio e dei tipi di conversione

Tabella 120. Tipi di messaggi e tipi di conversione

	Tipo di conversione			
Tipo messaggio	**Testo**	**Numerico**	**Altro**	**Nessuno**
JMSObjectMessage				getObject setObject
JMSTextMessage	getText setText			

Tabella 120. Tipi di messaggi e tipi di conversione (Continua)

	Tipo di conversione			
Tipo messaggio	**Testo**	**Numerico**	**Altro**	**Nessuno**
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

Concetti correlati

Approcci di conversione dei messaggi JMS

Alcuni approcci di conversione dati sono aperti ai progettisti di applicazioni JMS. Questi approcci non sono esclusivi; alcune applicazioni probabilmente utilizzeranno una combinazione di questi approcci. Se l'applicazione sta scambiando solo testo o messaggi solo con altre applicazioni JMS, normalmente non si considera la conversione dei dati. La conversione dei dati viene eseguita automaticamente da WebSphere WebSphere MQ.

Conversione e codifica dei messaggi del client JMS

Vengono elencati i metodi utilizzati per eseguire la conversione e la codifica dei messaggi del client JMS, con esempi di codice di ciascun tipo di conversione.

Conversione dati del gestore code

La conversione dei dati del gestore code è sempre stata disponibile per le applicazioni non - JMS che ricevono i messaggi dai client JMS. A partire da V7.0, i client JMS che ricevono messaggi utilizzano anche la conversione dei dati del gestore code. Da 7.0.1.5o 7.0.1.4 con APAR IC72897, la conversione dei dati del gestore code è facoltativa.

Attività correlate

Scambio di un record formattato con un'applicazione non JMS

Seguire la procedura suggerita in questa attività per progettare e creare un'uscita di conversione dati e un'applicazione client JMS che può scambiare messaggi con un'applicazione non JMS utilizzando `JMSBytesMessage`. Lo scambio di un messaggio formattato con un'applicazione non JMS può avvenire con o senza richiamare un'uscita di conversione dati.

Conversione e codifica dei messaggi del client JMS

Vengono elencati i metodi utilizzati per eseguire la conversione e la codifica dei messaggi del client JMS, con esempi di codice di ciascun tipo di conversione.

La conversione e la codifica si verificano quando gli oggetti o le primitive Java vengono letti o scritti in e da messaggi JMS. La conversione viene denominata conversione dati client JMS per distinguerla dalla conversione dati del gestore code e dalla conversione dati dell'applicazione. La conversione avviene rigorosamente quando i dati vengono letti o scritti in un messaggio JMS. Il testo viene convertito in e dalla rappresentazione interna Unicode a 16 bit[6]alla serie di caratteri utilizzata per il testo nei messaggi. I dati numerici vengono convertiti in tipi numerici primitivi Java nella codifica definita per il messaggio. Se viene eseguita la conversione e quale tipo di conversione viene eseguito, dipende dal tipo di messaggio JMS e dall'operazione di lettura o scrittura.

Tabella 121 a pagina 843 categorizza i metodi di lettura e scrittura per diversi tipi di messaggi JMS in base al tipo di conversione eseguita. I tipi di conversione sono descritti nel testo che segue la tabella.

Tabella 121. Tipi di messaggi e tipi di conversione

	Tipo di conversione			
Tipo messaggio	**Testo**	**Numerico**	**Altro**	**Nessuno**
`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] Alcune rappresentazioni Unicode richiedono più di 16 bit. Consultare un riferimento Java SE.

Tabella 121. Tipi di messaggi e tipi di conversione (Continua)				
	Tipo di conversione			
Tipo messaggio	**Testo**	**Numerico**	**Altro**	**Nessuno**
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

Testo

Il valore predefinito CodedCharacterSetId per una destinazione è 1208, UTF-8. Per impostazione predefinita, il testo viene convertito da Unicode e inviato come stringa di testo UTF-8 . In ricezione, il testo viene convertito dal set di caratteri codificati nel messaggio ricevuto dal client in Unicode.

I metodi setText e writeString convertono il testo da Unicode nella serie di caratteri definita per la destinazione. Un'applicazione può sostituire la serie di caratteri di destinazione impostando la proprietà del messaggio JMS_IBM_CHARACTER_SET. JMS_IBM_CHARACTER_SET, quando si invia un messaggio deve essere un CCSID (coded character set identifier) numerico[7].

I frammenti di codice in “Invio e ricezione di un JMSTextmessage” a pagina 846 inviano due messaggi. Uno viene inviato nella serie di caratteri definita per la destinazione e l'altro nella serie di caratteri 37, definita dall'applicazione.

I metodi getText e readString convertono il testo nel messaggio dalla serie di caratteri definita nel messaggio in Unicode. I metodi utilizzano la codepage definita nella proprietà del messaggio JMS_IBM_CHARACTER_SET. La codepage è associata da MQRFH2.CodedCharacterSetId a meno che il messaggio non sia un messaggio di MQe non abbia MQRFH2. Se il messaggio è un messaggio di tipo MQ, senza MQRFH2, la codepage viene associata da MQMD.CodedCharacterSetId.

Il frammento di codice in Figura 147 a pagina 847 riceve il messaggio che è stato inviato alla destinazione. Il testo nel messaggio viene riconvertito dalla codepage IBM037 in Unicode.

Nota: Un modo semplice per controllare che il testo sia convertito in un set di caratteri codificati 37 consiste nell'utilizzare WebSphere MQ Explorer. Sfoglia la coda e mostra le proprietà del messaggio prima che venga richiamato.

Contrasto del frammento di codice in Figura 146 a pagina 847 con il frammento di codice non corretto in Figura 143 a pagina 845. Nel frammento non corretto la stringa di testo viene convertita due volte, una dall'applicazione e di nuovo da WebSphere MQ.

[7] Quando si riceve un messaggio JMS_IBM_CHARACTER_SET è un nome di codepage Java Charset .

```
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);
```

Figura 143. Conversione codepage non corretta

Il metodo `writeUTF` converte il testo da Unicode a 1208, UTF-8. La stringa di testo ha come prefisso una lunghezza di 2 byte. La lunghezza massima della stringa di testo è 65534 byte. Il metodo `readUTF` legge un elemento in un messaggio scritto dal metodo `writeUTF` . Legge esattamente il numero di byte scritti dal metodo `writeUTF` .

Numerico

La codifica numerica predefinita per una destinazione è `Native`. La costante di codifica `Native` per Java ha il valore 273, `x'00000111'`, che è lo stesso per tutte le piattaforme. In ricezione, i numeri nel messaggio vengono trasformati correttamente in primitive Java numeriche. La trasformazione usa la codifica definita nel messaggio e il tipo restituito dal metodo read.

Il metodo di invio converte i numeri aggiunti a un messaggio da `set` e `write` nella codifica numerica definita per la destinazione. La codifica di destinazione può essere sovrascritta per un messaggio da un'applicazione che imposta la proprietà del messaggio `JMS_IBM_ENCODING`; ad esempio:

```
message.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
WMQConstants.WMQ_ENCODING_INTEGER_REVERSED);
```

I metodi numerici `get` e `read` convertono i numeri nel messaggio dalla codifica numerica definita nel messaggio. Convertono i numeri nel tipo specificato dal metodo `read` o `get` ; consultare La proprietà ENCODING . I metodi utilizzano la codifica definita in `JMS_IBM_ENCODING`. La codifica viene associata da `MQRFH2.Encoding` a meno che il messaggio non sia un messaggio di tipo MQe non abbia `MQRFH2`. Se il messaggio è un messaggio di MQdi tipo, senza `MQRFH2`, i metodi utilizzano la codifica definita in `MQMD.Encoding`.

L'esempio in Figura 148 a pagina 847 mostra un'applicazione che codifica un numero nel formato di destinazione e lo invia in un `JMSStreamMessage`. Confrontare l'esempio in Figura 148 a pagina 847 con quello in Figura 149 a pagina 847. La differenza è che `JMS_IBM_ENCODING` deve essere impostato in un `JMSBytesMessage`.

Nota: Un modo semplice per controllare che il numero sia codificato correttamente consiste nell'utilizzare WebSphere MQ Explorer. Sfoglia la coda e mostra le proprietà del messaggio prima che venga utilizzato.

Altro

I metodi `boolean` codificano `true` e `false` come `x'01'` e `x'00'` in un `JMSByteMessage`, `JMSStreamMessagee` `JMSMapMessage`.

I metodi UTF codificano e decodificano Unicode in stringhe di testo UTF-8 . Le stringhe sono limitate a meno di 65536 caratteri e sono precedute dal campo di lunghezza di 2 byte.

I metodi Object avvolgono i tipi primitivi come oggetti. I tipi numerici e di testo vengono codificati o convertiti come se i tipi primitivi fossero stati letti o scritti utilizzando i metodi numerici e di testo.

Nessuno

I metodi `readByte`, `readBytes`, `readUnsignedByte`, `writeBytee` `writeBytes` ottengono o inserendo singoli byte o array di byte, tra l'applicazione e il messaggio senza conversione. I metodi `readChar` e `writeChar` ottengono e inserendo caratteri Unicode a 2 byte tra l'applicazione e il messaggio senza conversione.

Utilizzando i metodi `readBytes` e `writeBytes` , l'applicazione può eseguire la propria conversione di punti di codice, come in “Invio e ricezione di testo in un JMSBytesMessage” a pagina 847.

WebSphere MQ non esegue alcuna conversione di codepage nel client poiché il messaggio è JMSBytesMessagee perché vengono utilizzati i metodi readBytes e writeBytes . Tuttavia, se i byte rappresentano il testo, assicurarsi che la codepage utilizzata dall'applicazione corrisponda alla serie di caratteri codificati della destinazione. Il messaggio potrebbe essere riconvertito da un'uscita di conversione del gestore code. Un'altra possibilità è che il programma client JMS ricevente possa seguire la convenzione di conversione di qualsiasi array di byte che rappresenta il testo nel messaggio in stringhe o caratteri utilizzando la proprietà JMS_IBM_CHARACTER_SET nel messaggio.

In questo esempio il client utilizza la serie di caratteri codificati di destinazione per la propria conversione:

```
bytes.writeBytes("In the destination code page".getBytes(
      CCSID.getCodepage(((MQDestination) destination)
           .getIntProperty(WMQConstants.WMQ_CCSID))));
```

In alternativa, il client potrebbe aver scelto una codepage e quindi impostare la serie di caratteri codificati corrispondente nella proprietà JMS_IBM_CHARACTER_SET del messaggio. Le classi WebSphere MQ per Java utilizzano JMS_IBM_CHARACTER_SET per impostare il campo CodedCharacterSetId nelle proprietà JMS in MQRFH2o nel descrittore del messaggio, MQMD:

```
String codePage = CCSID.getCodepage(37);
message.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage);[8]
```

Se un array di byte viene scritto in JMSStringMessage o JMSMapMessage, WebSphere MQ classes for JMS non esegue la conversione dei dati, poiché i byte vengono immessi come dati esadecimali e non come testo in JMSStringMessage e JMSMapMessage.

Se i byte rappresentano i caratteri nell'applicazione, è necessario considerare quali punti di codice leggere e scrivere nel messaggio. Il codice in Figura 144 a pagina 846 segue la convenzione di utilizzo della serie di caratteri codificati di destinazione. Se si crea la stringa utilizzando la serie di caratteri predefinita per la JVM, il contenuto dei byte dipende dalla piattaforma. Una JVM su Windows in genere ha un Charset predefinito di windows-1252e UNIX, UTF-8. L'interscambio tra Windows e UNIX richiede la selezione di una codepage esplicita per lo scambio di testo come byte.

```
StreamMessage smo = producer.session.createStreamMessage();
smo.writeBytes("123".getBytes(CCSID.getCodepage(((MQDestination) destination)
           .getIntProperty(WMQConstants.WMQ_CCSID))));
```

Figura 144. Scrittura di byte che rappresentano una stringa in JMSStreamMessage utilizzando la serie di caratteri di destinazione

Esempi

Invio e ricezione di un JMSTextmessage

Un messaggio di testo non può contenere testo in serie di caratteri differenti. L'esempio mostra il testo in diverse serie di caratteri, inviato in due diversi messaggi.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Figura 145. Invia messaggio di testo nella serie di caratteri definita dalla destinazione

[8] SetStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage) currently accepts only numeric character set identifiers.

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Figura 146. Invia messaggio di testo in ccsid 37

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Figura 147. Ricevi messaggio di testo

Esempi di codifica

Esempi che mostrano un numero inviato nella codifica definisce una destinazione. Notare che è necessario impostare la proprietà JMS_IBM_ENCODING di un JMSBytesMessage sul valore specificato per la destinazione.

```
StreamMessage smo = session.createStreamMessage();
smo.writeInt(256);
producer.send(smo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println(smi.readInt());
...
256
```

Figura 148. Invio di un numero utilizzando la codifica di destinazione in un 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
```

Figura 149. Invio di un numero utilizzando la codifica di destinazione in un JMSBytesMessage

Invio e ricezione di testo in un JMSBytesMessage

Il codice in Figura 150 a pagina 848 invia una stringa in un BytesMessage. Per semplicità, l'esempio invia una singola stringa, per cui un JMSTextMessage è più appropriato. Per ricevere un messaggio di testo in byte contenente una combinazione di tipi, è necessario conoscere la lunghezza della stringa in byte, denominata *TEXT_LENGTH* in Figura 151 a pagina 848. Anche per una stringa con un numero fisso di caratteri, la lunghezza della rappresentazione in byte potrebbe essere maggiore.

```
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);
```

Figura 150. Invio di un `String` in un `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);
```

Figura 151. Ricezione di un `String` da un `JMSBytesMessage`

Concetti correlati

Approcci di conversione dei messaggi JMS

Alcuni approcci di conversione dati sono aperti ai progettisti di applicazioni JMS. Questi approcci non sono esclusivi; alcune applicazioni probabilmente utilizzeranno una combinazione di questi approcci. Se l'applicazione sta scambiando solo testo o messaggi solo con altre applicazioni JMS, normalmente non si considera la conversione dei dati. La conversione dei dati viene eseguita automaticamente da WebSphere WebSphere MQ.

Conversione dati del gestore code

La conversione dei dati del gestore code è sempre stata disponibile per le applicazioni non - JMS che ricevono i messaggi dai client JMS. A partire da V7.0, i client JMS che ricevono messaggi utilizzano anche la conversione dei dati del gestore code. Da 7.0.1.5o 7.0.1.4 con APAR IC72897, la conversione dei dati del gestore code è facoltativa.

Attività correlate

Scambio di un record formattato con un'applicazione non JMS

Seguire la procedura suggerita in questa attività per progettare e creare un'uscita di conversione dati e un'applicazione client JMS che può scambiare messaggi con un'applicazione non JMS utilizzando `JMSBytesMessage`. Lo scambio di un messaggio formattato con un'applicazione non JMS può avvenire con o senza richiamare un'uscita di conversione dati.

Riferimenti correlati

Tipi di messaggio JMS e conversione

La scelta del tipo di messaggio influisce sull'approccio alla conversione del messaggio. L'interazione tra la conversione del messaggio e il tipo di messaggio è descritta per i tipi di messaggio JMS `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage` `JMSStreamMessagee` `JMSBytesMessage`.

Conversione dati del gestore code

La conversione dei dati del gestore code è sempre stata disponibile per le applicazioni non - JMS che ricevono i messaggi dai client JMS. A partire da V7.0, i client JMS che ricevono messaggi utilizzano anche la conversione dei dati del gestore code. Da 7.0.1.5o 7.0.1.4 con APAR IC72897, la conversione dei dati del gestore code è facoltativa.

Il gestore code può convertire i dati carattere e numerici nei dati del messaggio utilizzando i valori di `CodedCharacterSetId`, `Encoding`e `Format` impostati per i dati del messaggio. Per applicazioni non JMS la funzione di conversione è sempre stata disponibile impostando l'opzione `GetMessage`, `GMO_CONVERT`. La funzione di conversione del gestore code non è stata disponibile per un'applicazione JMS che riceve un messaggio fino a V7.0.

È possibile utilizzare la conversione del gestore code, prima di V7.0, con un'applicazione client JMS che invia un messaggio. Il client JMS crea un record formattato, imposta gli attributi `CodedCharacterSetId`, `Encoding`e `Format` corrispondenti ai dati inseriti nel messaggio. Un'applicazione ricevente non JMS

legge il messaggio utilizzando GMO_CONVERTe fa sì che venga richiamata un'uscita di conversione dati scritta dall'utente. L'uscita di conversione dati è una libreria condivisa con il nome impostato nel campo `Format` .

A partire da V7.0, il gestore code è in grado di convertire i messaggi inviati a client JMS. Da 7.0.0.0 a 7.0.1.4 incluso, la conversione del gestore code viene richiamata sempre per client JMS. Da 7.0.1.5, o da 7.0.1.4 con APAR IC72897 applicato, la conversione del gestore code è controllata impostando la proprietà di destinazione, `WMQ_RECEIVE_CONVERSION`, su `WMQ_RECEIVE_CONVERSION_QMGR`o `WMQ_RECEIVE_CONVERSION_CLIENT_MSG`. `WMQ_RECEIVE_CONVERSION_CLIENT_MSG` è l'impostazione predefinita, che corrisponde al comportamento di WebSphere MQ V6.0, che non supporta la conversione dei dati del gestore code per client JMS. L'applicazione può modificare l'impostazione della destinazione:

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Oppure,

```
((MQDestination)destination).setReceiveConversion
             (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Figura 152. Abilita conversione dati gestore code

La conversione dei dati del gestore code per un client JMS si verifica quando il client richiama un metodo `consumer.receive` . I dati di testo vengono trasformati in UTF-8 (1208) per impostazione predefinita. I metodi read e get successivi decodificano il testo nei dati ricevuti da UTF-8, creando primitive di testo Java nella relativa codifica Unicode interna. UTF-8 non è l'unica serie di caratteri di destinazione dalla conversione dei dati del gestore code. È possibile scegliere un CCSID differente impostando la proprietà destinazione `WMQ_RECEIVE_CCSID` .

Un'applicazione può anche modificare l'impostazione della destinazione, ad esempio impostandola su 437, DOS - US:

```
((MQDestination)destination).setIntProperty
             (WMQConstants.WMQ_RECEIVE_CCSID, 437);
```

Oppure,

```
((MQDestination)destination).setReceiveCCSID(437);
```

Figura 153. Imposta set di caratteri codificati di destinazione per la conversione del gestore code

Il motivo per cui si modifica `WMQ_RECEIVE_CCSID` è specializzato; il CCSID scelto non fa differenza per gli oggetti di testo creati nella JVM. Tuttavia, alcune JVM, su alcune piattaforme, potrebbero non essere in grado di gestire la conversione dal CCSID del testo nel messaggio in Unicode. L'opzione fornisce una scelta di CCSID per qualsiasi testo consegnato al client nel messaggio. Alcune piattaforme client JMS hanno avuto problemi con il testo del messaggio consegnato in UTF-8.

Il codice JMS equivale al testo in grassetto nel codice C in Figura 154 a pagina 850,

```
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                     */
```

Figura 154. Frammento di codice da `amqsget0.c`

Nota:

La conversione del gestore code viene eseguita solo sui dati del messaggio che hanno un formato WebSphere MQ noto. MQSTR, o MQCIH sono esempi di formati noti predefiniti. Un formato noto può anche essere definito dall'utente, purché sia stata fornita un'uscita di conversione dati.

I messaggi creati come JMSTextMessage, JMSMapMessage e JMSStreamMessage, hanno un formato MQSTR e possono essere convertiti dal gestore code.

Concetti correlati

Approcci di conversione dei messaggi JMS
Alcuni approcci di conversione dati sono aperti ai progettisti di applicazioni JMS. Questi approcci non sono esclusivi; alcune applicazioni probabilmente utilizzeranno una combinazione di questi approcci. Se l'applicazione sta scambiando solo testo o messaggi solo con altre applicazioni JMS, normalmente non si considera la conversione dei dati. La conversione dei dati viene eseguita automaticamente da WebSphere WebSphere MQ.

Conversione e codifica dei messaggi del client JMS
Vengono elencati i metodi utilizzati per eseguire la conversione e la codifica dei messaggi del client JMS, con esempi di codice di ciascun tipo di conversione.

“Richiamo dell'uscita di conversione dati” a pagina 418
Un'uscita di conversione dati è un'uscita scritta dall'utente che riceve il controllo durante l'elaborazione di una chiamata MQGET.

Attività correlate

Scambio di un record formattato con un'applicazione non JMS
Seguire la procedura suggerita in questa attività per progettare e creare un'uscita di conversione dati e un'applicazione client JMS che può scambiare messaggi con un'applicazione non JMS utilizzando JMSBytesMessage. Lo scambio di un messaggio formattato con un'applicazione non JMS può avvenire con o senza richiamare un'uscita di conversione dati.

Riferimenti correlati

Tipi di messaggio JMS e conversione
La scelta del tipo di messaggio influisce sull'approccio alla conversione del messaggio. L'interazione tra la conversione del messaggio e il tipo di messaggio è descritta per i tipi di messaggio JMS JMSObjectMessage, JMSTextMessage, JMSMapMessage JMSStreamMessagee JMSBytesMessage.

Scambio di un record formattato con un'applicazione non JMS
Seguire la procedura suggerita in questa attività per progettare e creare un'uscita di conversione dati e un'applicazione client JMS che può scambiare messaggi con un'applicazione non JMS utilizzando

JMSBytesMessage. Lo scambio di un messaggio formattato con un'applicazione non JMS può avvenire con o senza richiamare un'uscita di conversione dati.

Prima di iniziare

È possibile progettare una soluzione più semplice per lo scambio di messaggi con un'applicazione non JMS utilizzando un JMSTextMessage. Eliminare questa possibilità prima di seguire i passi in questa attività.

Informazioni su questa attività

Un client JMS è più semplice da scrivere se non è coinvolto nei dettagli di formattazione dei messaggi JMS scambiati con altri client JMS. Se il tipo di messaggio è JMSTextMessage, JMSMapMessage, JMSStreamMessageo JMSObjectMessage, WebSphere MQ si occupa dei dettagli di formattazione del messaggio. WebSphere MQ gestisce le differenze nelle code page e nella codifica numerica su piattaforme differenti.

È possibile utilizzare questi tipi di messaggi per scambiare messaggi con applicazioni non JMS. A tale scopo, è necessario comprendere in che modo questi messaggi vengono creati dalle classi WebSphere MQ per JMS. È possibile modificare l'applicazione non JMS per interpretare i messaggi; consultare “Associazione dei messaggi JMS sui messaggi WebSphere MQ” a pagina 812.

Un vantaggio dell'utilizzo di uno di questi tipi di messaggi è che la programmazione client JMS non dipende dal tipo di applicazione con cui sta scambiando i messaggi. Uno svantaggio è che potrebbe richiedere una modifica ad un altro programma e potrebbe non essere possibile modificare l'altro programma.

Un approccio alternativo consiste nel scrivere un'applicazione client JMS che possa gestire i formati di messaggio esistenti. Spesso i messaggi esistenti sono in formato fisso e contengono una combinazione di dati, testo e numeri non formattati. Utilizzare i passi di questa attività e il client di esempio JMS in “Scrittura di classi per incapsulare un layout di record in un JMSBytesMessage” a pagina 854come punto di partenza per la creazione di un client JMS che possa scambiare record formattati con applicazioni non JMS.

Procedura

1. Definire il layout del record oppure utilizzare una delle classi di intestazione WebSphere MQ predefinite.

   Per la gestione delle intestazioni predefinite di WebSphere MQ , consultare Gestione delle intestazioni dei messaggi WebSphere MQ.

   Figura 155 a pagina 852 è un esempio di layout di record a lunghezza fissa definito dall'utente, che può essere elaborato dal programma di utilità di conversione dati.
2. Creare l'uscita di conversione dati.

   Seguire le istruzioni in Scrittura di un programma di uscita di conversione dati per scrivere un'uscita di conversione dati.

   Per provare l'esempio in “Scrittura di classi per incapsulare un layout di record in un JMSBytesMessage” a pagina 854, denominare l'uscita di conversione dati MYRECORD.
3. Scrivere classi Java per incapsulare il layout del record e inviare e ricevere il record. Due approcci possibili sono:
   - Scrivere una classe in cui leggere e scrivere il file JMSBytesMessage che contiene il record; consultare “Scrittura di classi per incapsulare un layout di record in un JMSBytesMessage” a pagina 854.
   - Scrivere una classe che estenda com.ibm.mq.header.Header per definire la struttura dati del record; consultare Creazione di classi per nuovi tipi di intestazione.
4. Decidere in quale serie di caratteri codificati scambiare i messaggi.

Consultare “Scelta di un approccio alla conversione del messaggio: il ricevitore è corretto” a pagina 832.

5. Configurare la destinazione per lo scambio di messaggi di tipo MQ, senza un'intestazione JMS MQRFH2 .

   Sia la destinazione di invio che quella di ricezione devono essere configurate per lo scambio di messaggi di tipo MQ. È possibile utilizzare la stessa destinazione sia per l'invio che per la ricezione.

   L'applicazione può sovrascrivere la proprietà del corpo del messaggio di destinazione:

   ```
   ((MQDestination)destination).setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ```

   L'esempio in “Scrittura di classi per incapsulare un layout di record in un JMSBytesMessage” a pagina 854 sovrascrive la proprietà del corpo del messaggio di destinazione, verificando che venga inviato un messaggio in stile MQ.

6. Verifica della soluzione con applicazioni JMS e non JMS

   Strumenti utili per testare un'uscita di conversione dati sono:

   - Il programma di esempio `amqsgetc0.c` è utile per verificare la ricezione di un messaggio inviato da un client JMS. Consultare le modifiche suggerite per utilizzare l'intestazione di esempio, `RECORD.h`, in Figura 156 a pagina 853. Con le modifiche, `amqsgetc0.c` riceve un messaggio inviato dal client JMS di esempio, `TryMyRecord.java`; consultare “Scrittura di classi per incapsulare un layout di record in un JMSBytesMessage” a pagina 854.
   - Il programma di esplorazione di WebSphere MQ , `amqsbcg0.c`, è utile per esaminare il contenuto dell'intestazione del messaggio, dell'intestazione JMS, `MQRFH2`e il contenuto del messaggio.
   - Il programma **`rfhutil`** , precedentemente disponibile in SupportPac IH03, consente di catturare e memorizzare i messaggi di test nei file e quindi utilizzarli per gestire i flussi di messaggi. I messaggi di output possono anche essere letti e visualizzati in diversi formati. I formati includono due tipi di XML e la corrispondenza con un copybook COBOL. I dati possono essere in EBCDIC o ASCII. È possibile aggiungere un'intestazione RFH2 al messaggio prima che venga inviato.

   Se si tenta di ricevere i messaggi utilizzando il programma di esempio `amqsgetc0.c` modificato e si riceve un errore con codice di errore `2080`, verificare se il messaggio ha un `MQRFH2`. Le modifiche presuppongono che il messaggio sia stato inviato a una destinazione che non specifica `MQRFH2`.

Esempi

```
struct RECORD { MQCHAR StrucID[4];
                MQLONG Version;
                MQLONG StructLength;
                MQLONG Encoding;
                MQLONG CodeCharSetId;
                MQCHAR Format[8];
                MQLONG Flags;
                MQCHAR RecordData[32];
    };
```

Figura 155. RECORD.h

- Dichiarare la struttura dati 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);
```

- Modificare la chiamata MQGET per utilizzare RECORD,

  1. Prima della modifica:

```
    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. Dopo la modifica:

```
    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           */
```

- Modificare l'istruzione di stampa,

  1. Da:

```
buffer[messlen] = '\0';            /* add terminator          */
printf("message <%s>\n", buffer);
```

  2. A:

```
/* buffer[messlen] = '\0';            add terminator          */
printf("ccsid <%d>, flags <%d>, message <%32.32s>\n \0",
        md.CodedCharSetId, record.Flags, record.RecordData);
```

Figura 156. Modificare amqsget0.c

Concetti correlati

Approcci di conversione dei messaggi JMS
Alcuni approcci di conversione dati sono aperti ai progettisti di applicazioni JMS. Questi approcci non sono esclusivi; alcune applicazioni probabilmente utilizzeranno una combinazione di questi approcci. Se l'applicazione sta scambiando solo testo o messaggi solo con altre applicazioni JMS, normalmente non si considera la conversione dei dati. La conversione dei dati viene eseguita automaticamente da WebSphere WebSphere MQ.

Conversione e codifica dei messaggi del client JMS
Vengono elencati i metodi utilizzati per eseguire la conversione e la codifica dei messaggi del client JMS, con esempi di codice di ciascun tipo di conversione.

Conversione dati del gestore code
La conversione dei dati del gestore code è sempre stata disponibile per le applicazioni non - JMS che ricevono i messaggi dai client JMS. A partire da V7.0, i client JMS che ricevono messaggi utilizzano anche la conversione dei dati del gestore code. Da 7.0.1.5o 7.0.1.4 con APAR IC72897, la conversione dei dati del gestore code è facoltativa.

Riferimenti correlati

Tipi di messaggio JMS e conversione
La scelta del tipo di messaggio influisce sull'approccio alla conversione del messaggio. L'interazione tra la conversione del messaggio e il tipo di messaggio è descritta per i tipi di messaggio JMS `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage` `JMSStreamMessagee` `JMSBytesMessage`.

Programma di utilità per la creazione del codice di uscita di conversione

Scrittura di classi per incapsulare un layout di record in un `JMSBytesMessage`
Lo scopo di questa attività è di esplorare, per esempio, come combinare la conversione dei dati e un layout di record fisso in un `JMSBytesMessage`. Nell'attività, si creano alcune classi Java per scambiare una struttura di record di esempio in un `JMSBytesMessage`. È possibile modificare l'esempio per scrivere classi per scambiare altre strutture di record.

Un `JMSBytesMessage` è la scelta migliore del tipo di messaggio JMS per lo scambio di record di tipi di dati misti con programmi non JMS. Non dispone di ulteriori dati inseriti nel corpo del messaggio dal provider JMS. È quindi la scelta migliore del tipo di messaggio da utilizzare se un programma client JMS interagisce con un programma IBM WebSphere MQ esistente. La sfida principale nell'utilizzo di un `JMSBytesMessage` viene fornita con la corrispondenza della codifica e della serie di caratteri previsti dall'altro programma. Una soluzione è creare una classe che incapsula il record. Una classe che incapsula la lettura e la scrittura di un `JMSBytesMessage`, per un tipo di record specifico, rende più semplice l'invio e la ricezione di record in formato fisso in un programma JMS. Catturando gli aspetti generali dell'interfaccia in una classe astratta, gran parte della soluzione può essere riutilizzata per formati di record diversi. Diversi formati di record possono essere implementati in classi che estendono la classe generica astratta.

Un approccio alternativo consiste nell'estendere la classe `com.ibm.mq.headers.Header` . La classe `Header` dispone di metodi, come `addMQLONG`, per creare un formato record in modo più dichiarativo. Uno svantaggio dell'utilizzo della classe `Header` è ottenere e l'impostazione degli attributi utilizza un'interfaccia interpretativa più complicata. Entrambi gli approcci risultano nella stessa quantità di codice dell'applicazione.

Un `JMSBytesMessage` può incapsulare solo un formato singolo, oltre a un `MQRFH2`, in un messaggio, a meno che ogni record non utilizzi lo stesso formato, serie di caratteri codificati e codifica. Il formato, la codifica e la serie di caratteri di un `JMSBytesMessage` sono proprietà di tutti i messaggi che seguono `MQRFH2`. L'esempio è scritto supponendo che un `JMSBytesMessage` contenga solo un record utente.

Prima di iniziare

1. Il tuo livello di competenza: devi avere familiarità con la programmazione Java e JMS. Non vengono fornite istruzioni sull'impostazione dell'ambiente di sviluppo Java. È vantaggioso aver scritto un programma per scambiare un `JMSTextMessage`, `JMSStreamMessageo` `JMSMapMessage`. È possibile quindi visualizzare le differenze nello scambio di un messaggio utilizzando un `JMSBytesMessage`.
2. L'esempio richiede IBM WebSphere MQ V7.0.
3. L'esempio è stato creato utilizzando la prospettiva Java del workbench Eclipse . Richiede JRE 6.0 o superiore. È possibile utilizzare la prospettiva Java in IBM WebSphere MQ Explorer per sviluppare ed eseguire le classi Java. In alternativa, utilizzare il proprio ambiente di sviluppo Java.
4. L'utilizzo di Esplora risorse di IBM WebSphere MQ rende più semplice l'impostazione dell'ambiente di test e il debug rispetto all'utilizzo dei programmi di utilità della riga comandi.

Informazioni su questa attività

Si è guidati attraverso la creazione di due classi: RECORD e MyRecord. Insieme queste due classi racchiudono un record a formato fisso. Hanno metodi per ottenere e impostare gli attributi. Il metodo get legge il record da un JMSBytesMessage e il metodo put scrive un record in JMSBytesMessage.

Lo scopo dell'attività non è creare una classe di qualità di produzione che è possibile riutilizzare. È possibile scegliere di utilizzare gli esempi nell'attività per iniziare a utilizzare le proprie classi. Lo scopo dell'attività è quello di fornire note di guida, principalmente sull'utilizzo di serie di caratteri, formati e codifica, quando si utilizza un JMSBytesMessage. Ogni fase nella creazione delle classi viene spiegata e vengono descritti gli aspetti dell'utilizzo di JMSBytesMessage, a volte trascurati.

La classe RECORD è astratta e definisce alcuni campi comuni per un record utente. I campi comuni sono modellati sul layout di intestazione IBM WebSphere MQ standard con un campo di tipo eye catcher, una versione e una lunghezza. I campi di codifica, set di caratteri e formato, trovati in molte intestazioni IBM WebSphere MQ , vengono omessi. Un'altra intestazione non può seguire un formato definito dall'utente. La classe MyRecord , che estende la classe RECORD , lo fa estendendo letteralmente il record con ulteriori campi utente. Un JMSBytesMessage, creato dalle classi, può essere elaborato dall'uscita di conversione dati del gestore code.

"Classi utilizzate per eseguire l'esempio" a pagina 861 include un elenco completo di RECORD e MyRecord. Include anche elenchi delle classi "scaffolding" aggiuntive per verificare RECORD e MyRecord. Le classi extra sono:

TryMyRecord
Il programma principale per testare RECORD e MyRecord.

EndPoint
Una classe astratta che incapsula la connessione, la destinazione e la sessione JMS in una singola classe. La sua interfaccia soddisfa le esigenze di verifica delle classi RECORD e MyRecord . Non è un modello di progettazione stabilito per la scrittura di applicazioni JMS.

Nota: La classe Endpoint include questa riga di codice dopo la creazione di una destinazione:

```
((MQDestination)destination).setReceiveConversion
                            (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

In V7.0, da V7.0.1.5, è necessario attivare la conversione del gestore code. Per impostazione predefinita, questa funzione è disabilitata. In V7.0, la conversione del gestore code V7.0.1.4 è abilitata per impostazione predefinita e questa riga di codice causa un errore.

MyProducer e MyConsumer
Classi che estendono EndPointe creano un MessageConsumer e un MessageProducer, connessi e pronti ad accettare richieste.

Insieme, tutte le classi costituiscono un'applicazione completa che è possibile creare e sperimentare, per comprendere come utilizzare la conversione dati in un JMSBytesMessage.

Procedura

1. Creare una classe astratta per incapsulare i campi standard in un'intestazione IBM WebSphere MQ , con un costruttore predefinito. Successivamente, si estende la classe per adattare l'intestazione ai propri requisiti.

```
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();
    }
```

Nota:

a. Gli attributi, da `structID` a `nextFormat`, sono elencati nell'ordine in cui sono disposti in un'intestazione di messaggio IBM WebSphere MQ standard.

b. Gli attributi `format`, `messageEncoding`e `messageCharset`, descrivono l'intestazione stessa e non fanno parte dell'intestazione.

c. È necessario decidere se memorizzare il CCSID (coded character set identifier) o la serie di caratteri del record. Java utilizza serie di caratteri e i messaggi IBM WebSphere MQ utilizzano identificativi di serie di caratteri codificati. Il codice di esempio utilizza serie di caratteri.

d. `int` viene serializzato in `MQLONG` da IBM WebSphere MQ. `MQLONG` è di 4 byte.

2. Creare i getter e i setter per gli attributi privati.

a) Creare o generare i getter:

```
    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) Creare o generare i setter:

```
    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. Crea un costruttore per creare un'istanza RECORD da una `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());
    }
```

Nota:

a. `messageCharset` e `messageEncoding`vengono catturati dalle proprietà del messaggio, poiché sovrascrivono i valori impostati per la destinazione. `format` non è aggiornato. L'esempio non effettua alcuna verifica degli errori. Se viene richiamato il costruttore `Record(BytesMessage)` , si presume che `JMSBytesMessage` sia un messaggio di tipo `RECORD` . La linea "`setStructID(new String(structID, getMessageCharset()))`" imposta l'eye catcher.

b. Le righe di codice che completano i campi deserializzati del metodo nel messaggio, in ordine, aggiornando i valori predefiniti impostati nell'istanza RECORD.

4. Creare un metodo put per scrivere i campi di intestazione in un 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;
}
```

Nota:

a. MyProducer incapsula JMS Connection, Destination, Sessione MessageProducer in una singola classe. MyConsumer, utilizzato successivamente, incapsula JMS Connection, Destination, Sessione MessageConsumer in una classe singola.

b. Per un JMSBytesMessage, se la codifica è diversa da Native, la codifica deve essere impostata nel messaggio. La codifica di destinazione viene copiata nell'attributo di codifica del messaggio, JMS_IBM_CHARACTER_SET, e salvata come attributo della classe RECORD .

   i) "setMessageEncoding(myProducer.getEncoding());" chiama "(((MQDestination) destination).getIntProperty(WMQConstants.WMQ_ENCODING));" per ottenere la codifica di destinazione.

   ii) "Bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getMessageEncoding());" imposta la codifica del messaggio.

c. La serie di caratteri utilizzata per trasformare il testo in byte si ottiene dalla destinazione e viene salvata come attributo della classe RECORD . Non è impostato nel messaggio, perché non viene utilizzato dalle classi IBM WebSphere MQ per JMS durante la scrittura di un JMSBytesMessage.

   Chiamate "messageCharset = myProducer.getCharset();"

```
public String getCharset() throws UnsupportedEncodingException,
        JMSException {
    return CCSID.getCodepage(getCCSID());
}
```

   Richiama la serie di caratteri Java da un CCSID (coded character set identifier).

   "CCSID.getCodepage(ccsid)" si trova nel pacchetto com.ibm.mq.headers. ccsid si ottiene da un altro metodo in MyProducer, che interroga la destinazione:

```
public int getCCSID() throws JMSException {
    return (((MQDestination) destination)
            .getIntProperty(WMQConstants.WMQ_CCSID));
}
```

d. "myProducer.setMQClient(true);" sovrascrive le impostazioni di destinazione per il tipo di client, forzandolo su un client IBM WebSphere MQ MQI. È possibile che si preferisca omettere questa riga di codice, poiché oscura un errore di configurazione di gestione.

   "myProducer.setMQClient(true);" chiama:

```
((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ); }
if (!getMQDest()) setMQBody();
```

   Il codice ha l'effetto collaterale di impostare lo stile del corpo IBM WebSphere MQ su non specificato, se deve sovrascrivere un'impostazione di JMS.

   Nota:

Le classi IBM WebSphere MQ per JMS scrivono il formato, la codifica e l'identificativo della serie di caratteri del messaggio nel descrittore del messaggio, MQMD, o nell'intestazione JMS, MQRFH2. Dipende dal fatto che il messaggio abbia un corpo di stile IBM WebSphere MQ . Non impostare i campi MQMD manualmente.

Esiste un metodo per impostare manualmente le proprietà del descrittore del messaggio. Utilizza le proprietà JMS_IBM_MQMD_* . È necessario impostare la proprietà di destinazione WMQ_MQMD_WRITE_ENABLED per impostare le proprietà JMS_IBM_MQMD_* :

```
((MQDestination)destination).setMQMDWriteEnabled(true);
```

È necessario impostare la proprietà di destinazione, WMQ_MQMD_READ_ENABLED, per leggere le proprietà.

Utilizzare JMS_IBM_MQMD_* solo se si assume il controllo completo sull'intero payload del messaggio. Diversamente dalle proprietà JMS_IBM_* , le proprietà JMS_IBM_MQMD_* non controllano il modo in cui le classi IBM WebSphere MQ per JMS creano un messaggio JMS. È possibile creare proprietà del descrittore del messaggio in conflitto con le proprietà del messaggio JMS.

e. Le righe di codice che completano il metodo serializzano gli attributi nella classe come campi nel messaggio.

Gli attributi stringa vengono riempiti con spazi. Le stringhe vengono convertite in byte utilizzando la serie di caratteri definita per il record e troncate alla lunghezza dei campi del messaggio.

5. Completare la classe aggiungendo le importazioni.

```
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. Creare una classe per estendere la classe RECORD per includere ulteriori campi. Includere un costruttore predefinito.

```
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);
    }
```

Nota:

a. La sottoclasse RECORD , MyRecord, personalizza l'elemento eye catcher, il formato e la lunghezza dell'intestazione.

7. Creare o generare i getter e i setter.

a) Creare i getter:

```
    public int getFlags() { return flags; }
    public String getRecordData() { return recordData; }Â .
```

b) Creare i setter:

```
    public void setFlags(int flags) {
        this.flags = flags; }
    public void setRecordData(String recordData) {
        this.recordData = recordData; }
}
```

8. Crea un costruttore per creare un'istanza MyRecord da una 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()));
    }
```

Nota:

a. I campi che costituiscono il modello di messaggio standard vengono letti per primi dalla classe RECORD .

b. Il testo recordData viene convertito in String utilizzando la proprietà della serie di caratteri del messaggio.

9. Creare un metodo statico per ottenere un messaggio da un consumer e creare una nuova istanza MyRecord .

```
    public static MyRecord get(MyConsumer myConsumer) throws JMSException,
            MQDataException, IOException {
        BytesMessage message = (BytesMessage) myConsumer.receive();
        return new MyRecord(message);
    }
```

Nota:

a. Nell'esempio, per brevità, il costruttore MyRecord(BytesMessage) viene richiamato dal metodo get statico. Generalmente, è possibile separare la ricezione del messaggio dalla creazione di una nuova istanza MyRecord .

10. Creare un metodo put per accodare i campi cliente a JMSBytesMessage contenente un'intestazione del messaggio.

```
    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;
    }
```

Nota:

a. Il metodo richiama nel codice la serializzazione degli attributi nella classe MyRecord come campi nel messaggio.

- L'attributo recordData String viene riempito con spazi vuoti, convertito in byte utilizzando la serie di caratteri definita per il record e troncato alla lunghezza dei campi RecordData .

11. Completare la classe aggiungendo le istruzioni include.

```
package com.ibm.mq.id;
import java.io.IOException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.headers.MQDataException;
```

Risultati

Risultati:

- I risultati dell'esecuzione della classe `TryMyRecord` :
  - Invio di un messaggio nella serie di caratteri codificata 37 e utilizzo di un'exit di conversione del gestore code:

```
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
```

  - Invio del messaggio nella serie di caratteri codificati 37 e *non* utilizzando un'exit di conversione del gestore code:

```
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
```

- I risultati della modifica della classe `TryMyRecord` per non ricevere il messaggio e per riceverlo invece utilizzando l'esempio `amqsget0.c` modificato. L'esempio modificato accetta un record formattato; consultare Figura 156 a pagina 853 in “Scambio di un record formattato con un'applicazione non JMS” a pagina 850.
  - Invio di un messaggio nella serie di caratteri codificata 37 e utilizzo di un'exit di conversione del gestore code:

```
Sample AMQSGET0 start
ccsid <850>, flags <1>, message <ABCDEFGHIJKLMNOPQRSTUVWXYZ012345>
no more messages
Sample AMQSGET0 end
```

  - Invio del messaggio nella serie di caratteri codificati 37 e *non* utilizzando un'exit di conversione del gestore code:

```
Sample AMQSGET0 start
MQGET ended with reason code 2110
ccsid <37>, flags <1>, message <--+-+ãÃ++ÐÊËÈiÐÎÐ+ÔÒõõµþÞÚ-±=¼¶§>
no more messages
Sample AMQSGET0 end
```

Per provare l'esempio e provare diverse codepage e un'uscita di conversione dati. Creare le classi Java, configurare IBM WebSphere MQed eseguire il programma principale, `TryMyRecord`; consultare Figura 157 a pagina 861.

1. Configurare IBM WebSphere MQ e JMS per eseguire l'esempio. Le istruzioni sono per l'esecuzione dell'esempio su Windows.

   1. Creare un gestore code

```
crtmqm -sa -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1
```

   2. Creare una coda

```
echo DEFINE QL('Q1') REPLACE | runmqsc QM1
```

   3. Creare una directory JNDI

```
cd c:\
md JNDI-Directory
```

   4. Passare alla directory bin JMS
   Il programma di amministrazione JMS deve essere eseguito da qui. Il percorso è *MQ_INSTALLATION_PATH*\java\bin.

5. Creare le seguenti definizioni JMS in un file denominato `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. Eseguire il programma JMSAdmin per creare le risorse JMS

```
JMSAdmin < JMSQM1Q1.txt
```

2. È possibile creare, modificare e sfogliare le definizioni create utilizzando Esplora risorse di IBM WebSphere MQ .
3. Eseguire `TryMyRecord`.

Classi utilizzate per eseguire l'esempio

Le classi elencate nelle figure da Figura 157 a pagina 861 a Figura 162 a pagina 865 sono disponibili anche in un file compresso; scaricare jm25529_.zip o 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());
    }
}
```

Figura 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; }
}
```

Figura 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; }
}
```

Figura 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); }
    }
```

Figura 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); }
}
```

Figura 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();  }
}
```

Figura 162. MyConsumer

Creazione e configurazione di factory di connessione e destinazioni in un'applicazione WebSphere MQ per JMS

Un'applicazione WebSphere MQ può creare factory di connessione e destinazioni richiamandole come oggetti gestiti da uno spazio dei nomi JNDI (Java Naming and Directory Interface), utilizzando le estensioni JMS IBM o utilizzando le estensioni JMS WebSphere MQ . Un'applicazione può anche utilizzare le estensioni JMS IBM o le estensioni JMS WebSphere MQ per impostare le proprietà delle factory di connessione e delle destinazioni.

I factory di connessione e le destinazioni sono punti di avvio nel flusso della logica di un'applicazione JMS. Un'applicazione utilizza un oggetto ConnectionFactory per creare una connessione a un server di messaggistica e utilizza un oggetto coda o argomento come destinazione per inviare messaggi o un'origine da cui ricevere messaggi. Un'applicazione deve quindi creare almeno una factory di connessione e una o più destinazioni. Dopo aver creato una factory di connessione o una destinazione, l'applicazione potrebbe dover configurare l'oggetto impostando una o più proprietà.

In sintesi, un'applicazione può creare e configurare factory di connessione e destinazioni nei modi seguenti:

Utilizzo di JNDI per richiamare gli oggetti gestiti
Un amministratore può utilizzare lo strumento di amministrazione JMS di WebSphere MQ o WebSphere MQ Explorer per creare e configurare le factory di connessione e le destinazioni come oggetti gestiti in uno spazio dei nomi JNDI. Un'applicazione può quindi richiamare gli oggetti gestiti dallo spazio nomi JNDI. Dopo aver richiamato un oggetto amministrato, l'applicazione può, se necessario, impostare o modificare una o più proprietà utilizzando le estensioni JMS IBM o le estensioni JMS WebSphere MQ .

Utilizzo delle estensioni JMS IBM JMS
Un'applicazione può utilizzare le estensioni JMS IBM per creare in modo dinamico factory di connessione e destinazioni durante il runtime. L'applicazione crea innanzitutto un oggetto factory JmsFactory, quindi utilizza i metodi di questo oggetto per creare factory di connessione e destinazioni. Dopo aver creato una factory di connessione o una destinazione, l'applicazione può utilizzare i metodi ereditati dall'interfaccia di contesto JmsPropertyper impostare le relative proprietà. In alternativa, l'applicazione può utilizzare un URI (uniform resource identifier) per specificare una o più proprietà di una destinazione quando crea la destinazione.

Utilizzo delle estensioni WebSphere MQ JMS
Un'applicazione può inoltre utilizzare le estensioni JMS di WebSphere MQ per creare dinamicamente factory di connessione e destinazioni in fase di runtime. L'applicazione utilizza i costruttori forniti per creare factory di connessione e destinazioni. Dopo aver creato un factory di connessione o una destinazione, l'applicazione può utilizzare i metodi dell'oggetto per impostare le relative proprietà. In alternativa, l'applicazione può utilizzare un URI per specificare una o più proprietà di una destinazione quando crea la destinazione.

Utilizzo di JNDI per richiamare gli oggetti gestiti in una applicazione JMS

Per richiamare gli oggetti gestiti da uno spazio nomi JNDI (Java Naming and Directory Interface), un'applicazione JMS deve creare un contesto iniziale e quindi utilizzare il metodo lookup () per recuperare gli oggetti.

Prima che un'applicazione possa richiamare gli oggetti gestiti da uno spazio nomi JNDI, un amministratore deve prima creare gli oggetti gestiti. L'amministratore può utilizzare lo strumento di amministrazione JMS di WebSphere MQ o WebSphere MQ Explorer per creare e gestire gli oggetti gestiti in uno spazio dei nomi JNDI. Per informazioni su come utilizzare lo strumento di gestione JMS WebSphere MQ , consultare “Utilizzo dello strumento di amministrazione JMS WebSphere MQ” a pagina 936. Per informazioni su come utilizzare WebSphere MQ Explorer, consultare la guida fornita con WebSphere MQ Explorer. Un server delle applicazioni, tuttavia, generalmente fornisce il proprio repository per gli oggetti gestiti e i propri strumenti per la creazione e la gestione degli oggetti.

Per richiamare gli oggetti gestiti da un namespace JNDI, un'applicazione deve prima creare un contesto iniziale, come mostrato nel seguente esempio:

```
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);
```

In questo codice, le variabili stringa `url` e `icf` hanno i seguenti significati:

URL
L'URL (uniform resource locator) del servizio directory. L'URL può avere uno dei formati seguenti:

- `ldap://`*`hostname`*`/`*`contextName`*, per un servizio di directory basato su un server LDAP
- `file:/`*`directoryPath`*, per un servizio di directory basato sul file system locale

ICF
Il nome classe del factory di contesto iniziale, che può essere uno dei seguenti valori:

- `com.sun.jndi.ldap.LdapCtxFactory`, per un servizio di directory basato su un server LDAP
- `com.sun.jndi.fscontext.RefFSContextFactory`, per un servizio di directory basato sul file system locale

Notare che alcune combinazioni di un package JNDI e di un fornitore di servizi LDAP (Lightweight Directory Access Protocol) possono causare l'errore LDAP 84. Per risolvere questo problema, inserire la seguente riga di codice prima della chiamata a InitialDirContext ():

```
environment.put(Context.REFERRAL, "throw");
```

Dopo aver ottenuto un contesto iniziale, l'applicazione può recuperare gli oggetti gestiti dallo spazio nomi JNDI utilizzando il metodo lookup (), come mostrato nel seguente esempio:

```
ConnectionFactory factory;
Queue queue;
Topic topic;
.
.
.
factory = (ConnectionFactory)ctx.lookup("cn=myCF");
queue = (Queue)ctx.lookup("cn=myQ");
topic = (Topic)ctx.lookup("cn=myT");
```

Questo codice richiama i seguenti oggetti da un namespace basato su LDAP:

- Un oggetto ConnectionFactory collegato con il nome myCF
- Un oggetto Queue collegato con il nome myQ
- Un oggetto Argomento collegato con nome myT

Utilizzo delle estensioni JMS IBM JMS

WebSphere MQ classes per JMS contiene una serie di estensioni all'API JMS denominate IBM estensioni JMS. Un'applicazione può utilizzare queste estensioni per creare factory di connessione e destinazioni in modo dinamico al runtime e per impostare le proprietà delle classi WebSphere MQ per gli oggetti JMS. Le estensioni possono essere utilizzate con qualsiasi provider di messaggistica.

Le estensioni JMS IBM sono una serie di interfacce e classi nei seguenti pacchetti:

- com.ibm.msg.client.jms
- com.ibm.msg.client.services

I package possono essere trovati in `com.ibm.mqjms.jar` , che si trova in `<MQ_Install_Dir>/java/lib`.

Queste estensioni forniscono la seguente funzione:

- Un meccanismo basato su factory per la creazione di factory di connessioni e destinazioni in modo dinamico durante il runtime, invece di richiamarli come oggetti gestiti da uno spazio dei nomi JNDI (Java Naming and Directory Interface)
- Una serie di metodi per l'impostazione delle proprietà delle classi WebSphere MQ per oggetti JMS
- Una serie di classi di eccezioni con metodi per ottenere informazioni dettagliate su un problema
- Una serie di metodi per il controllo della traccia
- Una serie di metodi per ottenere informazioni sulla versione delle classi WebSphere MQ per JMS

Per quanto riguarda la creazione dinamica di factory di connessione e destinazioni in fase di runtime e l'impostazione e l'acquisizione delle relative proprietà, le estensioni JMS IBM forniscono una serie alternativa di interfacce alle estensioni JMS WebSphere MQ . Tuttavia, mentre le estensioni JMS di WebSphere MQ sono specifiche per il provider di messaggistica WebSphere MQ , le estensioni JMS di IBM non sono specifiche per WebSphere MQ e possono essere utilizzate con qualsiasi provider di messaggistica all'interno dell'architettura a livelli descritta in “Un'architettura a livelli” a pagina 800.

L'interfaccia com.ibm.msg.client.wmq.WMQConstants contiene definizioni di costanti, che un'applicazione può utilizzare quando si impostano le proprietà delle classi WebSphere MQ per gli oggetti JMS utilizzando le estensioni JMS IBM . L'interfaccia contiene costanti per il provider di messaggistica WebSphere MQ e le costanti JMS che sono indipendenti da qualsiasi provider di messaggistica.

Gli esempi di codice che seguono presuppongono che siano state eseguite le seguenti istruzioni di importazione:

```
import com.ibm.msg.client.jms.*;
import com.ibm.msg.client.services.*;
import com.ibm.msg.client.wmq.WMQConstants;
```

Creazione di factory di connessione e destinazioni

Prima che un'applicazione possa creare factory di connessione e destinazioni utilizzando le estensioni JMS IBM , è necessario creare un oggetto factory JmsFactory. Per creare un oggetto Factory JmsFactory, l'applicazione richiama il metodo getInstance() della classe Factory JmsFactory, come mostrato nel seguente esempio:

```
JmsFactoryFactory ff = JmsFactoryFactory.getInstance(JmsConstants.WMQ_PROVIDER);
```

Il parametro sulla chiamata getInstance() è una costante che identifica il provider di messaggistica WebSphere MQ come provider di messaggistica scelto. L'applicazione può quindi utilizzare l'oggetto factory JmsFactoryper creare factory di connessione e destinazioni.

Per creare una factory di connessione, l'applicazione richiama il metodo createConnectionFactory () dell'oggetto Factory JmsFactory, come mostrato nel seguente esempio:

```
JmsConnectionFactory factory = ff.createConnectionFactory();
```

Questa istruzione crea un oggetto JmsConnectionFactory con i valori predefiniti per tutte le relative proprietà, il che significa che l'applicazione si connette al gestore code predefinito in modalità bind. Se si desidera che un'applicazione si connetta in modalità client o a un gestore code diverso dal gestore code predefinito, l'applicazione deve impostare le proprietà appropriate dell'oggetto factory JmsConnectionprima di creare la connessione. Per informazioni su come svolgere questa procedura, consultare “Impostazione delle proprietà delle classi WebSphere MQ per gli oggetti JMS” a pagina 869.

La classe factory JmsFactorycontiene anche metodi per creare factory di connessione dei tipi seguenti:

- JmsQueueConnectionFactory
- JmsTopicConnectionFactory
- Factory JmsXAConnection
- JmsXAQueueConnectionFactory
- JmsXATopicConnectionFactory

Per creare un oggetto Queue, l'applicazione richiama il metodo createQueue() dell'oggetto Factory JmsFactory, come mostrato nel seguente esempio:

```
JmsQueue q1 = ff.createQueue("Q1");
```

Questa istruzione crea un oggetto JmsQueue con i valori predefiniti per tutte le relative proprietà. L'oggetto rappresenta una coda WebSphere MQ denominata Q1 appartenente al gestore code locale. Questa coda può essere una coda locale, una coda alias o una definizione di coda remota.

Il metodo createQueue() può anche accettare un URI (uniform resource identifier) della coda come parametro. Un URI coda è una stringa che specifica il nome di una coda WebSphere MQ e, facoltativamente, il nome del gestore code che possiede la coda e una o più proprietà dell'oggetto JmsQueue . La seguente istruzione contiene un esempio di URI della coda:

```
JmsQueue q2 = ff.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

L'oggetto JmsQueue creato da questa istruzione rappresenta una coda WebSphere MQ denominata Q2 di proprietà del gestore code QM2e tutti i messaggi inviati a questa destinazione sono persistenti e hanno una priorità di 5. Per ulteriori informazioni sugli URI della coda, consultare “URI (Uniform Resource Identifier)” a pagina 881. Per una modalità alternativa di impostazione delle proprietà di un oggetto JmsQueue , consultare “Impostazione delle proprietà delle classi WebSphere MQ per gli oggetti JMS” a pagina 869.

Per creare un oggetto Topic, un'applicazione può utilizzare il metodo createTopic() dell'oggetto Factory JmsFactory, come mostrato nel seguente esempio:

```
JmsTopic t1 = ff.createTopic("Sport/Football/Results");
```

Questa istruzione crea un oggetto JmsTopic con i valori predefiniti per tutte le sue proprietà. L'oggetto rappresenta un argomento chiamato Sport / Calcio/Risultati.

Il metodo createTopic() può accettare anche un URI argomento come parametro. Un URI dell'argomento è una stringa che specifica il nome di un argomento e, facoltativamente, una o più proprietà dell'oggetto JmsTopic . Le seguenti istruzioni contengono un esempio di URI argomento:

```
String s1 = "topic://Sport/Tennis/Results?persistence=1&priority=0";
JmsTopic t2 = ff.createTopic(s1);
```

L'oggetto JmsTopic creato da queste istruzioni rappresenta un argomento denominato Sport / Tennis/ Results e tutti i messaggi inviati a questa destinazione sono non persistenti e hanno una priorità 0. Per ulteriori informazioni sugli URI argomento, consultare “URI (Uniform Resource Identifier)” a pagina 881. Per un modo alternativo di impostare le proprietà di un oggetto JmsTopic , vedere “Impostazione delle proprietà delle classi WebSphere MQ per gli oggetti JMS” a pagina 869.

Dopo che un'applicazione ha creato una factory di connessione o una destinazione, tale oggetto può essere utilizzato solo con il fornitore di messaggistica selezionato.

Impostazione delle proprietà delle classi WebSphere MQ per gli oggetti JMS

Per impostare le proprietà delle classi WebSphere MQ per oggetti JMS utilizzando le estensioni JMS di IBM , un'applicazione utilizza i metodi dell'interfaccia com.ibm.msg.client.JmsPropertyContext .

Per ogni tipo di dati Java, l'interfaccia di contesto JmsPropertycontiene un metodo per impostare il valore di una proprietà con quel tipo di dati e un metodo per ottenere il valore di una proprietà con quel tipo di dati. Ad esempio, un'applicazione richiama il metodo setIntProperty () per impostare una proprietà con un valore intero e richiama il metodo getIntProperty () per ottenere una proprietà con un valore intero.

Le istanze delle classi nel pacchetto com.ibm.mq.jms ereditano anche i metodi dell'interfaccia di contesto JmsProperty. Un'applicazione può quindi utilizzare questi metodi per impostare le proprietà di oggetti MQConnectionFactory, MQQueue e MQTopic.

Quando un'applicazione crea un oggetto WebSphere MQ classes for JMS, tutte le proprietà con valori predefiniti vengono impostate automaticamente. Quando un'applicazione imposta una proprietà, il nuovo valore sostituisce qualsiasi valore precedente della proprietà. Una volta impostata una proprietà, non è possibile eliminarla, ma è possibile modificarne il valore.

Se un'applicazione tenta di impostare una proprietà su un valore non valido per la proprietà, WebSphere MQ classes per JMS genera un'eccezione JMSException. Se un'applicazione tenta di ottenere una proprietà che non è stata impostata, il funzionamento è quello descritto nella specifica JMS. WebSphere MQ classes per JMS genera un'eccezione NumberFormatper tipi di dati primitivi e restituisce null per i tipi di dati a cui si fa riferimento.

Oltre alle proprietà predefinite di un oggetto WebSphere MQ classes for JMS, un'applicazione può impostare le proprie proprietà. Queste proprietà definite dall'applicazione vengono ignorate da WebSphere MQ classes per JMS.

Per ulteriori informazioni sulle proprietà delle classi WebSphere MQ per gli oggetti JMS, vedere Proprietà degli oggetti IBM WebSphere MQ classes for JMS.

Il seguente codice è un esempio di come impostare le proprietà utilizzando le estensioni JMS di IBM . Il codice imposta cinque proprietà di una factory di connessione.

```
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");
```

L'effetto dell'impostazione di queste proprietà è che l'applicazione si connette al gestore code QM1 in modalità client, utilizzando un canale MQI denominato QM1.SVR. Il gestore code è in esecuzione su un sistema con il nome host HOST1e il listener per il gestore code è in attesa nella porta numero 1415. Questa connessione e altre connessioni del gestore code associate alle sessioni al di sotto di essa, hanno il nome dell'applicazione "Applicazione personale" associato ad esse.

Nota: I gestori code in esecuzione su piatteforme z/OS non supportano l'impostazione dei nomi delle applicazioni, pertanto questa impostazione viene ignorata.

L'interfaccia di contesto JmsPropertycontiene anche il metodo setObjectProperty (), che un'applicazione può utilizzare per impostare le proprietà. Il secondo parametro del metodo è un oggetto che incapsula il valore della proprietà. Ad esempio, il seguente codice crea un oggetto Integer che incapsula il numero intero 1415 e quindi richiama setObjectProperty () per impostare la proprietà PORT di una factory di connessione sul valore 1415:

```
Integer port = new Integer(1415);
factory.setObjectProperty(WMQConstants.WMQ_PORT, port);
```

Questo codice è quindi equivalente alla seguente istruzione:

```
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
```

Al contrario, il metodo getObjectProperty () restituisce un oggetto che incapsula il valore di una proprietà.

Conversione implicita di un valore di proprietà da un tipo di dati a un altro

Quando un'applicazione utilizza un metodo dell'interfaccia di contesto JmsPropertyper impostare o ottenere la proprietà di un oggetto WebSphere MQ classes per JMS, il valore della proprietà può essere implicitamente convertito da un tipo di dati ad un altro.

Ad esempio, la seguente istruzione imposta la proprietà PRIORITY dell'oggetto JmsQueue q1:

```
q1.setStringProperty(WMQConstants.WMQ_PRIORITY, "5");
```

La proprietà PRIORITY ha un valore intero, quindi la chiamata setStringProperty () converte implicitamente la stringa "5" (il valore di origine) nel numero intero 5 (il valore di destinazione), che diventa il valore della proprietà PRIORITY.

Al contrario, la seguente istruzione richiama la proprietà PRIORITY dell'oggetto JmsQueue q1:

```
String s1 = q1.getStringProperty(WMQConstants.WMQ_PRIORITY);
```

Il numero intero 5 (il valore di origine), che è il valore della proprietà PRIORITY, viene implicitamente convertito nella stringa "5" (il valore di destinazione) dalla chiamata getStringProperty ().

Le conversioni supportate dalle classi WebSphere MQ per JMS sono riportate in Tabella 122 a pagina 870.

Tabella 122. Conversioni supportate da un tipo di dati a un altro

Tipo di dati di origine	**Tipi di dati di destinazione supportati**
booleano	Stringa
byte	int, long, short, String
char	Stringa
doppio	Stringa
mobile	double, Stringa
int	long, stringa

Tabella 122. Conversioni supportate da un tipo di dati a un altro (Continua)	
Tipo di dati di origine	**Tipi di dati di destinazione supportati**
completo	Stringa
breve	int, long, Stringa
Stringa	booleano, byte, double, float, int, long, short

Le regole generali che disciplinano le conversioni supportate sono le seguenti:

- I valori numerici possono essere convertiti da un tipo di dati ad un altro purché non vengano persi dati durante la conversione. Ad esempio, un valore con tipo di dati `int` può essere convertito in un valore con tipo di dati `long`, ma non può essere convertito in un valore con tipo di dati `short`.
- Un valore di qualsiasi tipo di dati può essere convertito in una stringa.
- Una stringa può essere convertita in un valore di qualsiasi altro tipo di dati (eccetto `char`) purché la stringa sia nel formato corretto per la conversione. Se un'applicazione tenta di convertire una stringa che non è nel formato corretto, WebSphere MQ classes per JMS genera un'eccezione NumberFormat.
- Se un'applicazione tenta una conversione non supportata, WebSphere MQ classes per JMS genera un'eccezione MessageFormat.

Le regole specifiche per convertire un valore da un tipo di dati ad un altro sono le seguenti:

- Quando si converte un valore booleano in una stringa, il valore `true` viene convertito nella stringa "true"e il valore `false` viene convertito nella stringa "false".
- Quando si converte una stringa in un valore booleano, la stringa "true" (non sensibile al maiuscolo / minuscolo) viene convertita in `true`e la stringa "false" (non sensibile al maiuscolo / minuscolo) viene convertita in `false`. Qualsiasi altra stringa viene convertita in `false`.
- Quando si converte una stringa in un valore con tipo di dati `byte`, `int`, `long`o `short`, la stringa deve avere il seguente formato:

  [*spazi*] [*segno*]*cifre*

  I significati dei componenti della stringa sono i seguenti:

  spazi vuoti
  Caratteri vuoti iniziali facoltativi.

  segno
  Un segno più (+) o meno (-) facoltativo.

  cifre
  Una sequenza contigua di cifre (0-9). Deve essere presente almeno una cifra.

  Dopo la sequenza di cifre, la stringa può contenere altri caratteri che non sono cifre, ma la conversione si arresta non appena viene raggiunto il primo di questi caratteri. Si presuppone che la stringa rappresenti un numero intero decimale.

  Se la stringa non è nel formato corretto, WebSphere MQ classes for JMS genera un'eccezione NumberFormat.
- Quando si converte una stringa in un valore con tipo di dati `double` o `float`, la stringa deve avere il formato seguente:

  [*blank*] [*sign*]*cifre*[*e_char*[*e_sign*]*e_digits*]

  I significati dei componenti della stringa sono i seguenti:

  spazi vuoti
  Caratteri vuoti iniziali facoltativi.

  segno
  Un segno più (+) o meno (-) facoltativo.

cifre
Una sequenza contigua di cifre (0-9). Deve essere presente almeno una cifra.

e_car
Un carattere esponente, che è *E* o *e*.

e_firma
Un segno più (+) o meno (-) facoltativo per l'esponente.

e_cifre
Una sequenza contigua di cifre (0-9) per l'esponente. Deve essere presente almeno una cifra se la stringa contiene un carattere esponente.

Dopo la sequenza di cifre o i caratteri facoltativi che rappresentano un esponente, la stringa può contenere altri caratteri che non sono cifre, ma la conversione si interrompe non appena viene raggiunto il primo di questi caratteri. Si presuppone che la stringa rappresenti un numero a virgola mobile decimale con un esponente che sia una potenza di 10.

Se la stringa non è nel formato corretto, WebSphere MQ classes for JMS genera un'eccezione NumberFormat.

- Quando si converte un valore numerico (incluso un valore con tipo di dati `byte`) in una stringa, il valore viene convertito nella rappresentazione stringa del valore come numero decimale, non nella stringa contenente il carattere ASCII per tale valore. Ad esempio, il numero intero 65 viene convertito nella stringa "65", non nella stringa "A".

Impostazione di più di una proprietà in una singola chiamata

L'interfaccia di contesto JmsPropertycontiene anche il metodo setBatchProperties (), che un'applicazione può utilizzare per impostare più di una proprietà in una singola chiamata. Il parametro del metodo è un oggetto mappa che incapsula una serie di coppie nome - valore della proprietà.

Ad esempio, il seguente codice utilizza il metodo setBatchProperties () per impostare le stesse cinque proprietà di una factory di connessione come mostrato in “Impostazione delle proprietà delle classi WebSphere MQ per gli oggetti JMS” a pagina 869. Il codice crea un'istanza della classe HashMap , che implementa l'interfaccia 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);
```

Il secondo parametro del metodo Map.put() deve essere un oggetto. Pertanto, un valore di proprietà con un tipo di dati primitivo deve essere incapsulato all'interno di un oggetto o rappresentato da una stringa, come mostrato nell'esempio.

Il metodo setBatchProperties () convalida ciascuna proprietà. Se il metodo setBatchProperties () non può impostare una proprietà perché, ad esempio, il suo valore non è valido, nessuna delle proprietà specificate viene impostata.

Nomi e valori delle proprietà

Se un'applicazione utilizza i metodi dell'interfaccia di contesto JmsPropertyper impostare e richiamare le proprietà delle classi WebSphere MQ per gli oggetti JMS, l'applicazione può specificare i nomi e i valori delle proprietà in uno dei seguenti modi. Ognuno degli esempi di accompagnamento mostra come impostare la proprietà PRIORITY dell'oggetto JmsQueue q1 in modo che un messaggio inviato alla coda abbia la priorità specificata nella chiamata send ().

Utilizzo dei nomi e valori delle proprietà definiti come costanti nell'interfaccia com.ibm.msg.client.wmq.WMQConstants

La seguente istruzione è un esempio di come specificare i nomi e valori delle proprietà in questo modo:

```
q1.setIntProperty(WMQConstants.WMQ_PRIORITY, WMQConstants.WMQ_PRI_APP);
```

Utilizzo dei nomi e dei valori delle proprietà che possono essere utilizzati negli URI (uniform resource identifier) della coda e dell'argomento

La seguente istruzione è un esempio di come specificare i nomi e valori delle proprietà in questo modo:

```
q1.setIntProperty("priority", -2);
```

Solo i nomi e valori delle proprietà delle destinazioni possono essere specificati in questo modo.

Utilizzo dei nomi e dei valori delle proprietà riconosciuti dallo strumento di gestione JMS WebSphere MQ

La seguente istruzione è un esempio di come specificare i nomi e valori delle proprietà in questo modo:

```
q1.setStringProperty("PRIORITY", "APP");
```

Anche la forma breve del nome della proprietà è accettabile, come mostrato nella seguente istruzione:

```
q1.setStringProperty("PRI", "APP");
```

Quando un'applicazione ottiene una proprietà, il valore restituito dipende dal modo in cui l'applicazione specifica il nome della proprietà. Ad esempio, se un'applicazione specifica la costante WMQConstants.WMQ_PRIORITY come nome della proprietà, il valore restituito è il numero intero -2:

```
int n1 = getIntProperty(WMQConstants.WMQ_PRIORITY);
```

Lo stesso valore viene restituito se l'applicazione specifica la stringa "priority" come nome della proprietà:

```
int n2 = getIntProperty("priority");
```

Tuttavia, se l'applicazione specifica la stringa "PRIORITY" o "PRI" come nome della proprietà, il valore restituito è la stringa "APP":

```
String s1 = getStringProperty("PRI");
```

Internamente, WebSphere MQ classes for JMS memorizza i nomi e valori delle proprietà come valori letterali definiti nell'interfaccia com.ibm.msg.client.wmq.WMQConstants . Questo è il formato canonico definito per i nomi e i valori delle proprietà. Come regola generale, se un'applicazione imposta le proprietà utilizzando uno degli altri due modi di specificare i nomi e i valori delle proprietà, le classi WebSphere MQ per JMS devono convertire i nomi e i valori dal formato di input specificato nel formato canonico. Allo stesso modo, se un'applicazione acquisisce le proprietà utilizzando uno degli altri due modi per specificare i nomi e i valori delle proprietà, le classi WebSphere MQ per JMS devono convertire i nomi dal formato di input specificato nel formato canonico e convertire i valori dal formato canonico nel formato di output richiesto. L'esecuzione di queste conversioni potrebbe avere implicazioni per le prestazioni.

I nomi e i valori delle proprietà restituiti dalle eccezioni, nei file di traccia o nel log di WebSphere MQ per JMS sono sempre in formato canonico.

Utilizzo dell'interfaccia Mappa

L'interfaccia di contesto JmsPropertyestende l'interfaccia java.util.Map . Un'applicazione può quindi utilizzare i metodi dell'interfaccia Mappa per accedere alle proprietà di un oggetto WebSphere MQ classes per JMS.

Ad esempio, il seguente codice stampa i nomi e i valori di tutte le proprietà di una factory di connessione. Il codice utilizza solo i metodi dell'interfaccia Mappa per ottenere nomi e valori delle proprietà.

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

L'utilizzo dei metodi dell'interfaccia Mappa non ignora le conversioni o le convalide delle proprietà.

Utilizzo delle estensioni WebSphere MQ JMS

WebSphere MQ classes per JMS contiene una serie di estensioni all'API di JMS denominate WebSphere MQ estensioni JMS. Un'applicazione può utilizzare queste estensioni per creare factory di connessione e destinazioni in modo dinamico al runtime e per impostare le proprietà delle factory di connessione e delle destinazioni.

WebSphere MQ classes per JMS contiene una serie di classi nei pacchetti com.ibm.jms e com.ibm.mq.jms. Queste classi implementano le interfacce JMS e contengono le estensioni JMS WebSphere MQ . Gli esempi di codice che seguono presuppongono che questi pacchetti siano stati importati dalle seguenti istruzioni:

```
import com.ibm.jms.*;
import com.ibm.mq.jms.*;
```

Un'applicazione può utilizzare le estensioni JMS di WebSphere MQ per eseguire le seguenti funzioni:

- Creare le factory di connessione e le destinazioni in modo dinamico durante il runtime, invece di richiamarle come oggetti gestiti da uno spazio dei nomi JNDI (Java Naming and Directory Interface)
- Impostare le proprietà delle factory di connessione e delle destinazioni

Creazione factory di connessione

Per creare un factory di connessioni, un'applicazione può utilizzare il costruttore MQConnectionFactory , come mostrato nel seguente esempio:

```
MQConnectionFactory factory = new MQConnectionFactory();
```

Questa istruzione crea un oggetto MQConnectionFactory con i valori predefiniti per tutte le sue proprietà, il che significa che l'applicazione si connette al gestore code predefinito in modalità di bind. Se si desidera che un'applicazione si connetta in modalità client o a un gestore code diverso dal gestore code predefinito, l'applicazione deve impostare le proprietà appropriate dell'oggetto MQConnectionFactory prima di creare la connessione. Per informazioni su come svolgere questa procedura, consultare “Impostazione delle proprietà delle factory di connessione” a pagina 874.

Un'applicazione può creare factory di connessione dei seguenti tipi in modo simile:

- Factory di MQQueueConnection
- Factory MQTopicConnection
- MQXAConnectionFactory
- Factory MQXAQueueConnection
- Factory MQXATopicConnection

Impostazione delle proprietà delle factory di connessione

Un'applicazione pu impostare le propriet ... di una produzione connessioni richiamando i metodi appropriati della produzione connessioni. Il factory di connessione può essere un oggetto gestito o un oggetto creato dinamicamente in fase di runtime.

Si consideri il seguente codice, ad esempio:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_CLIENT);
factory.setQueueManager("QM1");
factory.setHostName("HOST1");
factory.setPort(1415);
factory.setChannel("QM1.SVR");
```

Questo codice crea un oggetto MQConnectionFactory e imposta cinque proprietà dell'oggetto. L'effetto dell'impostazione di queste proprietà è che l'applicazione si connette al gestore code QM1 in modalità client utilizzando un canale MQI denominato QM1.SVR. Il gestore code è in esecuzione su un sistema con il nome host HOST1e il listener per il gestore code è in attesa nella porta numero 1415.

Per una connessione in tempo reale a un broker, un'applicazione può utilizzare il codice seguente:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_DIRECT);
factory.setHostName("HOST2");
factory.setPort(1507);
```

Questo codice presuppone che il broker sia in esecuzione su un sistema con nome host HOST2 e in ascolto sul numero di porta 1507.

Un'applicazione che utilizza una connessione in tempo reale a un broker può utilizzare solo lo stile di pubblicazione / sottoscrizione della messaggistica. Non può utilizzare lo stile point-to-point della messaggistica.

Sono valide solo alcune combinazioni di proprietà di una produzione connessioni. Per informazioni sulle combinazioni valide, consultare Dipendenze tra proprietà di WebSphere MQ classes per oggetti JMS .

Per ulteriori informazioni sulle proprietà di una factory di connessione e sui metodi utilizzati per impostarne le proprietà, vedere Proprietà degli oggetti IBM WebSphere MQ classes for JMS.

Creazione di destinazioni

Per creare un oggetto Queue, un'applicazione può utilizzare il costruttore MQQueue, come mostrato nel seguente esempio:

```
MQQueue q1 = new MQQueue("Q1");
```

Questa istruzione crea un oggetto MQQueue con i valori predefiniti per tutte le relative proprietà. L'oggetto rappresenta una coda WebSphere MQ denominata Q1 appartenente al gestore code locale. Questa coda può essere una coda locale, una coda alias o una definizione di coda remota.

Un formato alternativo del costruttore MQQueue ha due parametri, come mostrato nel seguente esempio:

```
MQQueue q2 = new MQQueue("QM2", "Q2");
```

L'oggetto MQQueue creato da questa istruzione rappresenta una coda WebSphere MQ denominata Q2 di proprietà del gestore code QM2. Il gestore code identificato in questo modo può essere il gestore code locale o un gestore code remoto. Se si tratta di un gestore code remoto, WebSphere MQ deve essere configurato in modo che, quando l'applicazione invia un messaggio a questa destinazione, Websphere MQ possa instradare il messaggio dal gestore code locale al gestore code remoto.

Il costruttore MQQueue può anche accettare un URI (uniform resource identifier) della coda come singolo parametro. Un URI della coda è una stringa che specifica il nome di una coda di WebSphere MQ e, facoltativamente, il nome del gestore code proprietario della coda e una o più proprietà dell'oggetto MQQueue. La seguente istruzione contiene un esempio di URI della coda:

```
MQQueue q3 = new MQQueue("queue://QM3/Q3?persistence=2&priority=5");
```

L'oggetto MQQueue creato da questa istruzione rappresenta una coda WebSphere MQ denominata Q3 di proprietà del gestore code QM3e tutti i messaggi inviati a questa destinazione sono persistenti e

hanno una priorità di 5. Per ulteriori informazioni sugli URI della coda, consultare “URI (Uniform Resource Identifier)” a pagina 881. Per un modo alternativo di impostare le proprietà di un oggetto MQQueue, vedere “Impostazione delle proprietà delle destinazioni” a pagina 876.

Per creare un oggetto Argomento, un'applicazione può utilizzare il costruttore MQTopic, come mostrato nel seguente esempio:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
```

Questa istruzione crea un oggetto MQTopic con i valori predefiniti per tutte le relative proprietà. L'oggetto rappresenta un argomento chiamato Sport / Calcio/Risultati.

Il costruttore MQTopic può anche accettare un URI argomento come parametro. Un URI argomento è una stringa che specifica il nome di un argomento e, facoltativamente, una o più proprietà dell'oggetto MQTopic. La seguente istruzione contiene un esempio di un URI argomento:

```
MQTopic t2 = new MQTopic("topic://Sport/Tennis/Results?persistence=1&priority=0");
```

L'oggetto MQTopic creato da questa istruzione rappresenta un argomento denominato Sport / Tennis/ Results e tutti i messaggi inviati a questa destinazione sono non persistenti e hanno una priorità 0. Per ulteriori informazioni sugli URI argomento, consultare “URI (Uniform Resource Identifier)” a pagina 881. Per un modo alternativo di impostare le proprietà di un oggetto MQTopic, consultare “Impostazione delle proprietà delle destinazioni” a pagina 876.

Impostazione delle proprietà delle destinazioni

Un'applicazione può impostare le proprietà di una destinazione richiamando i metodi appropriati della destinazione. La destinazione può essere un oggetto gestito o un oggetto creato dinamicamente al runtime.

Si consideri il seguente codice, ad esempio:

```
MQQueue q1 = new MQQueue("Q1");
.
q1.setPersistence(WMQConstants.WMQ_PER_PER);
q1.setPriority(5);
```

Questo codice crea un oggetto MQQueue e imposta due proprietà dell'oggetto. L'effetto dell'impostazione di queste proprietà è che tutti i messaggi inviati alla destinazione sono persistenti e hanno una priorità di 5.

Un'applicazione può impostare le proprietà dell'oggetto MQTopic in modo simile, come mostrato nel seguente esempio:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
.
t1.setPersistence(WMQConstants.WMQ_PER_NON);
t1.setPriority(0);
```

Questo codice crea un oggetto MQTopic e quindi imposta due proprietà dell'oggetto. L'effetto dell'impostazione di queste proprietà è che tutti i messaggi inviati alla destinazione sono non persistenti e hanno una priorità pari a 0.

Per ulteriori informazioni relative alle proprietà di una destinazione e ai metodi utilizzati per impostarne le proprietà, consultare Proprietà degli oggetti IBM WebSphere MQ classes for JMS.

Creazione di una connessione in un'applicazione JMS

Per creare un collegamento, un'applicazione JMS utilizza un oggetto ConnectionFactory per creare un collegamento e quindi avvia il collegamento.

Per creare un oggetto Connection, un'applicazione utilizza il metodo createConnection() di un oggetto ConnectionFactory , come mostrato nel seguente esempio:

```
ConnectionFactory factory;
Connection connection;
.
.
.
connection = factory.createConnection();
```

Quando si crea una connessione JMS, IBM WebSphere MQ classes for JMS crea un handle di connessione (Hconn) e avvia una conversazione con il gestore code.

L'interfaccia factory QueueConnectione l'interfaccia factory TopicConnectionereditano il metodo createConnection() dall'interfaccia di ConnectionFactory . È quindi possibile utilizzare il metodo createConnection() per creare un oggetto specifico del dominio, come mostrato nel seguente esempio:

```
QueueConnectionFactory qcf;
Connection connection;
.
.
.
connection = qcf.createConnection();
```

Questo frammento di codice crea un oggetto QueueConnection . Un'applicazione può ora eseguire un'operazione indipendente dal dominio su questo oggetto o un'operazione applicabile solo al dominio point - to - point. Tuttavia, se l'applicazione tenta di eseguire un'operazione applicabile solo al dominio di pubblicazione / sottoscrizione, viene generata un'eccezione IllegalStatecon il seguente messaggio:

```
JMSMQ1112: Operation for a domain specific object was not valid.
           Operation createProducer() is not valid for type com.ibm.mq.jms.MQTopic
```

Ciò si verifica perché la connessione è stata creata da un factory di connessione specifico del dominio.

Nota: Tenere presente che l'ID processo dell'applicazione viene utilizzato come identità utente predefinita da trasmettere al gestore code. Se l'applicazione è in esecuzione in modalità di trasporto client, questo ID processo deve esistere, con le autorizzazioni pertinenti, sul server. Se si desidera utilizzare un'identità differente, utilizzare il metodo createConnection(nome utente, password).

La specifica JMS indica che una connessione viene creata nello stato `stopped` . Fino a quando non viene avviata una connessione, un utente di messaggi associato alla connessione non può ricevere alcun messaggio. Per avviare una connessione, un'applicazione utilizza il metodo start () di un oggetto Connection, come mostrato nel seguente esempio:

```
connection.start();
```

Creazione di una sessione in un'applicazione JMS

Per creare una sessione, un'applicazione JMS utilizza il metodo createSession() di un oggetto Connection.

Il metodo createSession() ha due parametri:

1. Un parametro che specifica se la sessione è transattiva o meno
2. Un parametro che specifica la modalità di riconoscimento per la sessione

Ad esempio, il seguente codice crea una sessione che non è sottoposta a transazione e ha una modalità di riconoscimento AUTO_ACKNOWLEDGMENT:

```
Session session;
.
boolean transacted = false;
session = connection.createSession(transacted, Session.AUTO_ACKNOWLEDGE);
```

Quando viene creata una sessione JMS, IBM WebSphere MQ classes for JMS crea un handle di connessione (Hconn) e avvia una conversazione con il gestore code.

Un oggetto Session e qualsiasi oggetto MessageProducer o MessageConsumer da esso creato, non possono essere utilizzati contemporaneamente da thread differenti di un'applicazione a più thread. Il

modo più semplice per garantire che questi oggetti non vengano utilizzati contemporaneamente è creare un oggetto Session separato per ciascun thread.

Sessioni transazionali in applicazioni JMS

Le applicazioni JMS possono eseguire transazioni locali creando prima una sessione transattiva. Un'applicazione può eseguire il commit o il rollback di una transazione.

Le applicazioni JMS possono eseguire transazioni locali. Una transazione locale è una transazione che implica modifiche solo alle risorse del gestore code a cui è connessa l'applicazione. Per eseguire le transazioni locali, un'applicazione deve prima creare una sessione sottoposta a transazione richiamando il metodo createSession() di un oggetto Connection, specificando come parametro che la sessione viene sottoposta a transazione. Successivamente, tutti i messaggi inviati e ricevuti all'interno della sessione vengono raggruppati in una sequenza di transazioni. Una transazione termina quando l'applicazione esegue il commit o il rollback dei messaggi inviati e ricevuti dall'inizio della transazione.

Per eseguire il commit di una transazione, un'applicazione richiama il metodo commit () dell'oggetto Session. Quando viene eseguito il commit di una transazione, tutti i messaggi inviati all'interno della transazione diventano disponibili per il recapito ad altre applicazioni e tutti i messaggi ricevuti all'interno della transazione vengono riconosciuti in modo che il server di messaggistica non tenti di consegnarli nuovamente all'applicazione. Nel dominio point - to - point, il server di messaggistica rimuove anche i messaggi ricevuti dalle code.

Per eseguire il rollback di una transazione, un'applicazione richiama il metodo rollback () dell'oggetto Session. Quando viene eseguito il rollback di una transazione, tutti i messaggi inviati all'interno della transazione vengono eliminati dal server di messaggistica e tutti i messaggi ricevuti all'interno della transazione diventano nuovamente disponibili per la consegna. Nel dominio point - to - point, i messaggi ricevuti vengono reinseriti nelle loro code e diventano nuovamente visibili ad altre applicazioni.

Una nuova transazione viene avviata automaticamente quando un'applicazione crea una sessione sottoposta a transazione o richiama il metodo commit () o rollback (). Pertanto, una sessione transazionale ha sempre una transazione attiva.

Quando un'applicazione chiude una sessione transatta, si verifica un rollback implicito. Quando un'applicazione chiude una connessione, si verifica un rollback implicito per tutte le sessioni transazionali della connessione.

Se un'applicazione termina senza chiudere una connessione, si verifica anche un rollback implicito per tutte le sessioni transazionali della connessione.

Una transazione è interamente contenuta all'interno di una sessione transatta. Una transazione non può estendere le sessioni. Ciò significa che non è possibile per un'applicazione inviare e ricevere messaggi in due o più sessioni transazionali e quindi eseguire il commit o il rollback di tutte queste azioni come una singola transazione.

Modalità di riconoscimento delle sessioni JMS

Ogni sessione non sottoposta a transazione dispone di una modalità di riconoscimento che determina il modo in cui vengono riconosciuti i messaggi ricevuti dall'applicazione. Sono disponibili tre modalità di riconoscimento e la scelta della modalità di riconoscimento influisce sulla progettazione dell'applicazione.

Se non viene eseguita la transazione di una sessione, il modo in cui vengono riconosciuti i messaggi ricevuti dall'applicazione è determinato dalla modalità di riconoscimento della sessione. Le tre modalità di riconoscimento sono descritte nei seguenti paragrafi:

AUTO_RICONOSCIMENTO

La sessione riconosce automaticamente ogni messaggio ricevuto dall'applicazione.

Se i messaggi vengono consegnati in modo sincrono all'applicazione, la sessione conferma la ricezione di un messaggio ogni volta che una chiamata di ricezione viene completata correttamente. Se i messaggi vengono consegnati in modo asincrono, la sessione conferma la ricezione di un messaggio ogni volta che una chiamata al metodo onMessage() di un listener di messaggi viene completata correttamente.

Se l'applicazione riceve correttamente un messaggio, ma un errore impedisce il verificarsi del riconoscimento, il messaggio diventa nuovamente disponibile per la consegna. L'applicazione deve quindi essere in grado di gestire un messaggio che viene riconsegnato.

DUPS_OK_RICONOSCI

La sessione riconosce i messaggi ricevuti dall'applicazione nel momento in cui viene selezionata.

L'utilizzo di questa modalità di riconoscimento riduce la quantità di lavoro che la sessione deve eseguire, ma un malfunzionamento che impedisce la conferma di ricezione del messaggio potrebbe causare la ridisponibilità di più di un messaggio per la consegna. L'applicazione deve quindi essere in grado di gestire i messaggi che vengono riconsegnati.

Limitazione: Nelle modalità AUTO_CONOSCERE e DUPS_OK_CONOSCERE, JMS non supporta un'applicazione che genera un'eccezione non gestita in un listener di messaggi. Ciò significa che i messaggi vengono sempre riconosciuti quando il listener dei messaggi viene restituito, indipendentemente dal fatto che sia stato elaborato correttamente (a condizione che gli errori non siano irreversibili e non impediscano all'applicazione di continuare). Se si richiede un controllo più fine della conferma di ricezione del messaggio, utilizzare le modalità CLIENT_ACKNOWLEDGEMENT o transazionali, che forniscono all'applicazione il controllo completo delle funzioni di conferma di ricezione.

CLIENT_RICONOSCIMENTO

L'applicazione riconosce i messaggi ricevuti richiamando il metodo Acknowledge della classe Message.

L'applicazione può confermare la ricezione di ciascun messaggio singolarmente oppure può ricevere un batch di messaggi e richiamare il metodo Acknowledge solo per l'ultimo messaggio ricevuto. Quando il metodo di riconoscimento viene richiamato, vengono riconosciuti tutti i messaggi ricevuti dall'ultima volta che il metodo è stato richiamato.

Insieme a una di queste modalità di riconoscimento, un'applicazione può arrestare e riavviare la consegna dei messaggi in una sessione richiamando il metodo Recover della classe Session. I messaggi ricevuti ma precedentemente non riconosciuti vengono riconsegnati. Tuttavia, potrebbero non essere consegnati nella stessa sequenza in cui sono stati precedentemente consegnati. Nel frattempo, potrebbero essere arrivati messaggi con priorità più elevata e alcuni dei messaggi originali potrebbero essere scaduti. Nel dominio point - to - point, alcuni dei messaggi originali potrebbero essere stati utilizzati da un'altra applicazione.

Un'applicazione può determinare se un messaggio viene riconsegnato esaminando il contenuto del campo di intestazione JMSRefornita del messaggio. L'applicazione esegue questa operazione richiamando il metodo getJMSRedelivered() della classe Message.

Creazione di destinazioni in una applicazione JMS

Invece di richiamare le destinazioni come oggetti gestiti da uno spazio nomi JNDI (Java Naming and Directory Interface), un'applicazione JMS può utilizzare una sessione per creare le destinazioni in modo dinamico durante il runtime. Un'applicazione può utilizzare un URI (uniform resource identifier) per identificare una coda WebSphere MQ o un argomento e, facoltativamente, per specificare una o più proprietà di un oggetto Coda o Argomento.

Utilizzo di una sessione per creare oggetti Coda

Per creare un oggetto Coda, un'applicazione può utilizzare il metodo createQueue() di un oggetto Session, come mostrato nel seguente esempio:

```
Session session;
.
Queue q1 = session.createQueue("Q1");
```

Questo codice crea un oggetto Coda con i valori predefiniti per tutte le relative proprietà. L'oggetto rappresenta una coda WebSphere MQ denominata Q1 appartenente al gestore code locale. Questa coda può essere una coda locale, una coda alias o una definizione di coda remota.

Il metodo createQueue() accetta anche un URI della coda come parametro. Un URI della coda è una stringa che specifica il nome di una coda WebSphere MQ e, facoltativamente, il nome del gestore code che possiede la coda e una o più proprietà dell'oggetto Coda. La seguente istruzione contiene un esempio di URI della coda:

```
Queue q2 = session.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

L'oggetto Queue creato da questa istruzione rappresenta una coda WebSphere MQ denominata Q2 di proprietà di un gestore code denominato QM2e tutti i messaggi inviati a questa destinazione sono persistenti e hanno una priorità di 5. Il gestore code identificato in questo modo può essere il gestore code locale o un gestore code remoto. Se si tratta di un gestore code remoto, WebSphere MQ deve essere configurata in modo che, quando l'applicazione invia un messaggio a questa destinazione, Websphere MQ possa instradare il messaggio dal gestore code locale al gestore code QM2. Per ulteriori informazioni sugli URI, consultare “URI (Uniform Resource Identifier)” a pagina 881.

Si noti che il parametro nel metodo createQueue() contiene informazioni specifiche del provider. Pertanto, l'utilizzo del metodo createQueue() per creare un oggetto Queue, invece di richiamare un oggetto Queue come oggetto gestito da uno spazio dei nomi JNDI, potrebbe rendere l'applicazione meno portabile.

Un'applicazione può creare un oggetto TemporaryQueue utilizzando il metodo createTemporaryQueue () di un oggetto Session, come mostrato nel seguente esempio:

```
TemporaryQueue q3 = session.createTemporaryQueue();
```

Anche se una sessione viene utilizzata per creare una coda temporanea, l'ambito di una coda temporanea è la connessione utilizzata per creare la sessione. Qualsiasi sessione della connessione può creare produttori di messaggi e consumer di messaggi per la coda temporanea. La coda temporanea rimane fino al termine della connessione o fino a quando l'applicazione non elimina esplicitamente la coda temporanea utilizzando il metodo TemporaryQueue.delete (), a seconda di quale sia la prima.

Quando un'applicazione crea una coda temporanea, WebSphere MQ classes for JMS crea una coda dinamica nel gestore code a cui è connessa l'applicazione. La proprietà TEMPMODEL della factory di connessione specifica il nome della coda modello utilizzata per creare la coda dinamica e la proprietà TEMPQPREFIX della factory di connessione specifica il prefisso utilizzato per formare il nome della coda dinamica.

Utilizzo di una sessione per creare oggetti argomento

Per creare un oggetto Argomento, un'applicazione può utilizzare il metodo createTopic() di un oggetto Session, come mostrato nel seguente esempio:

```
Session session;
.
Topic t1 = session.createTopic("Sport/Football/Results");
```

Questo codice crea un oggetto argomento con i valori predefiniti per tutte le proprietà. L'oggetto rappresenta un argomento chiamato Sport / Calcio/Risultati.

Il metodo createTopic() accetta anche un URI argomento come parametro. Un URI argomento è una stringa che specifica il nome di un argomento e, facoltativamente, una o più proprietà dell'oggetto Argomento. Il codice riportato di seguito contiene un esempio di URI argomento:

```
String uri = "topic://Sport/Tennis/Results?persistence=1&priority=0";
Topic t2 = session.createTopic(uri);
```

L'oggetto Argomento creato da questo codice rappresenta un argomento denominato Sport / Tennis/ Results e tutti i messaggi inviati a questa destinazione non sono persistenti e hanno una priorità 0. Per ulteriori informazioni sugli URI argomento, consultare “URI (Uniform Resource Identifier)” a pagina 881.

Si noti che il parametro nel metodo () createTopiccontiene informazioni specifiche del provider. Pertanto, l'utilizzo del metodo createTopic() per creare un oggetto argomento, invece di richiamare un oggetto argomento come oggetto gestito da uno spazio dei nomi JNDI, potrebbe rendere l'applicazione meno portabile.

Un'applicazione può creare un oggetto TemporaryTopic utilizzando il metodo createTemporaryTopic () di un oggetto Session, come mostrato nel seguente esempio:

```
TemporaryTopic t3 = session.createTemporaryTopic();
```

Sebbene una sessione venga utilizzata per creare un argomento temporaneo, l'ambito di un argomento temporaneo è la connessione utilizzata per creare la sessione. Qualsiasi sessione della connessione può creare produttori di messaggi e consumer di messaggi per l'argomento temporaneo. L'argomento temporaneo rimane attivo fino al termine della connessione o fino a quando l'applicazione non elimina esplicitamente l'argomento temporaneo utilizzando il metodo TemporaryTopic.delete (), a seconda di quale delle due modalità si verifica prima.

Quando un'applicazione crea un argomento temporaneo, WebSphere MQ classes for JMS crea un argomento con un nome che inizia con i caratteri TEMP/*tempTopicPrefix*, dove *tempTopicPrefix* è il valore della proprietà TEMPTOPICPREFIX della factory di connessione.

URI (Uniform Resource Identifier)

Un URI della coda è una stringa che specifica il nome di una coda WebSphere MQ e, facoltativamente, il nome del gestore code proprietario della coda e una o più proprietà dell'oggetto Coda creato dall'applicazione. Un URI argomento è una stringa che specifica il nome di un argomento e, facoltativamente, una o più proprietà dell'oggetto Argomento creato dall'applicazione.

Un URI coda ha il formato seguente:

```
queue://[qMgrName]/qName[?propertyName1=propertyValue1
                         &propertyName2=propertyValue2
                         &...]
```

Un URI argomento ha il seguente formato:

```
topic://topicName[?propertyName1=propertyValue1
                  &propertyName2=propertyValue2
                  &...]
```

Le variabili in questi formati hanno i significati seguenti:

QmgrName
Il nome del gestore code proprietario della coda identificata dall'URI.

Il gestore code può essere il gestore code locale o un gestore code remoto. Se si tratta di un gestore code remoto, WebSphere MQ deve essere configurato in modo che, quando un'applicazione invia un messaggio alla coda, Websphere MQ può instradare il messaggio dal gestore code locale al gestore code remoto.

Se non viene specificato alcun nome, viene utilizzato il gestore code locale.

qName
Il nome della coda WebSphere MQ .

La coda può essere una coda locale, una coda alias o una definizione di coda remota.

Per le regole per la creazione di nomi coda, consultare Regole per la denominazione degli oggetti IBM WebSphere MQ .

topicName
Il nome dell'argomento.

Per le regole per la creazione dei nomi argomento, consultare Regole per la denominazione degli oggetti IBM WebSphere MQ. Evitare l'utilizzo dei caratteri jolly +, #, * e? nei nomi argomento. I

nomi degli argomenti che contengono questi caratteri possono causare risultati imprevisti quando si sottoscrivono. Consultare Utilizzo di stringhe di argomenti.

***propertyName1*, *propertyName2*, ...**
I nomi delle proprietà dell'oggetto Coda o Argomento creato dall'applicazione. Tabella 123 a pagina 882 elenca i nomi di proprietà validi che possono essere utilizzati in un URI.

Se non viene specificata alcuna proprietà, l'oggetto Coda o Argomento ha i valori predefiniti per tutte le relative proprietà.

***propertyValue1*, *propertyValue2*, ...**
I valori delle proprietà dell'oggetto Coda o Argomento creato dall'applicazione. Tabella 123 a pagina 882 elenca i valori di proprietà validi che possono essere utilizzati in un URI.

Le parentesi ([]) indicano un componente facoltativo e i puntini di sospensione (...) indicano che l'elenco delle coppie nome - valore della proprietà, se presenti, può contenere una o più coppie nome - valore.

Tabella 123 a pagina 882 elenca i nomi di proprietà validi e i valori validi che possono essere utilizzati negli URI di argomenti e code. Sebbene lo strumento di gestione JMS di WebSphere MQ utilizzi costanti simboliche per i valori delle proprietà, gli URI non possono contenere costanti simboliche.

Tabella 123. Nomi di proprietà e valori validi da utilizzare negli URI di argomenti e code

Nome della proprietà	**Descrizione**	**Valori validi**
CCSID	Come vengono rappresentati i dati di caratteri nel contenuto di un messaggio quando WebSphere MQ classes per JMS inoltra il messaggio alla destinazione	• Qualsiasi CCSID supportato da WebSphere MQ.
codifica	Il modo in cui i dati numerici nel contenuto di un messaggio vengono rappresentati quando WebSphere MQ classes per JMS inoltra il messaggio alla destinazione	• Qualsiasi valore valido per il campo *Codifica* in un descrittore di messaggi WebSphere MQ .
Scadenza	Il TTL (time to live) per i messaggi inviati alla destinazione	• -2 - Come specificato nella chiamata send () o, se non specificato nella chiamata send (), la durata predefinita del produttore del messaggio. • 0 - Un messaggio inviato alla destinazione non scade mai. • Un numero intero positivo che indica la durata in millisecondi.

Tabella 123. Nomi di proprietà e valori validi da utilizzare negli URI di argomenti e code (Continua)		
Nome della proprietà	**Descrizione**	**Valori validi**
supporto multicast	L'impostazione multicast per un argomento quando si utilizza una connessione in tempo reale a un broker	Il seguente elenco contiene valori validi. Associato a ciascun valore è il valore corrispondente della proprietà MULTICAST come utilizzato nello strumento di gestione JMS WebSphere MQ . Per una descrizione della proprietà MULTICAST e dei relativi valori validi, vedere Proprietà degli oggetti IBM WebSphere MQ classes for JMS. • -1 - ASCF • 0 - DISABILITATO • 3 - NOTR • 5 - AFFIDABILE • 7 - ABILITATO
persistenza	La persistenza dei messaggi inviati alla destinazione	• -2 - Come specificato nella chiamata send () o, se non specificato nella chiamata send (), la persistenza predefinita del produttore del messaggio. • -1 - Come specificato dall'attributo *DefPersistence* dell'argomento o della coda WebSphere MQ . • 1 - Non persistente. • 2 - Permanente. • 3 - Equivalente al valore HIGH per la proprietà PERSISTENCE utilizzato nello strumento di amministrazione JMS WebSphere MQ . Per una spiegazione di questo valore, consultare “Messaggi persistenti JMS” a pagina 906.
priorità	La priorità dei messaggi inviati alla destinazione	• -2 - Come specificato nella chiamata send () o, se non specificato nella chiamata send (), la priorità predefinita del produttore del messaggio. • -1 - Come specificato dall'attributo *DefPriority* della coda o argomento WebSphere MQ . • Un numero intero compreso tra 0 e 9 che specifica la priorità dei messaggi inviati alla destinazione.
targetClient	Se i messaggi inviati alla destinazione contengono un'intestazione MQRFH2	• 0 - I messaggi contengono un'intestazione MQRFH2 . • 1 - I messaggi non contengono un'intestazione MQRFH2 .

Ad esempio, il seguente URI identifica una coda WebSphere MQ denominata Q1 appartenente al gestore code locale. Un oggetto coda creato utilizzando questo URI ha i valori predefiniti per tutte le proprie proprietà.

```
queue:///Q1
```

Il seguente URI individua una coda WebSphere MQ denominata Q2 di proprietà di un gestore code denominato QM2. Tutti i messaggi inviati a questa destinazione hanno una priorità di 6. Le restanti proprietà dell'oggetto Coda creato utilizzando questo URI hanno i valori predefiniti.

```
queue://QM2/Q2?priority=6
```

Il seguente URI identifica un argomento denominato Sport / Atletica/Risultati. Tutti i messaggi inviati a questa destinazione sono non persistenti e hanno una priorità 0. Le restanti proprietà dell'oggetto Argomento creato utilizzando questo URI hanno i propri valori predefiniti.

```
topic://Sport/Athletics/Results?persistence=1&priority=0
```

Invio di messaggi in un'applicazione JMS

Prima che un'applicazione JMS possa inviare messaggi ad una destinazione, è necessario creare un oggetto MessageProducer per la destinazione. Per inviare un messaggio alla destinazione, l'applicazione crea un oggetto Messaggio e richiama il metodo send () dell'oggetto MessageProducer .

Un'applicazione utilizza un oggetto MessageProducer per inviare messaggi. Un'applicazione normalmente crea un oggetto MessageProducer per una destinazione specifica, che può essere una coda o un argomento, in modo che tutti i messaggi inviati utilizzando il produttore del messaggio vengano inviati alla stessa destinazione. Di conseguenza, prima che un'applicazione possa creare un oggetto MessageProducer , deve prima creare un oggetto Coda o Argomento. Per informazioni su come creare un oggetto Coda o Argomento, consultare i seguenti argomenti:

- “Utilizzo di JNDI per richiamare gli oggetti gestiti in una applicazione JMS” a pagina 866
- “Utilizzo delle estensioni JMS IBM JMS” a pagina 867
- “Utilizzo delle estensioni WebSphere MQ JMS” a pagina 874
- “Creazione di destinazioni in una applicazione JMS” a pagina 879

Per creare un oggetto MessageProducer , un'applicazione utilizza il metodo createProducer() di un oggetto Session, come mostrato nel seguente esempio:

```
MessageProducer producer = session.createProducer(destination);
```

Il parametro `destination` è un oggetto Coda o Argomento creato in precedenza dall'applicazione.

Prima che un'applicazione possa inviare un messaggio, deve creare un oggetto Messaggio. Il corpo di un messaggio contiene i dati dell'applicazione e JMS definisce cinque tipi di contenuto del messaggio:

- Byte
- Associa
- Oggetto
- Flusso
- Testo

Ogni tipo di corpo del messaggio ha la propria interfaccia JMS, che è un'interfaccia secondaria dell'interfaccia del messaggio e un metodo nell'interfaccia Session per la creazione di un messaggio con quel tipo di contenuto. Ad esempio, l'interfaccia per un messaggio di testo è denominata TextMessagee un'applicazione utilizza il metodo createTextMessage () di un oggetto Session per creare un messaggio di testo, come mostrato nella seguente istruzione:

```
TextMessage outMessage = session.createTextMessage(outString);
```

Per ulteriori informazioni sui messaggi e sul corpo del messaggio, consultare “Messaggi JMS” a pagina 809.

Per inviare un messaggio, un'applicazione utilizza il metodo send () di un oggetto MessageProducer , come mostrato nel seguente esempio:

```
producer.send(outMessage);
```

Un'applicazione può utilizzare il metodo send () per inviare messaggi in uno dei domini di messaggistica. La natura della destinazione determina quale dominio di messaggistica viene utilizzato. Tuttavia, TopicPublisher, l'interfaccia secondaria di MessageProducer specifica del dominio di pubblicazione / sottoscrizione, dispone anche di un metodo publish (), che può essere utilizzato al posto del metodo send (). I due metodi sono funzionalmente gli stessi.

Un'applicazione può creare un oggetto MessageProducer senza una destinazione specificata. In questo caso, l'applicazione deve specificare la destinazione quando si richiama il metodo send ().

Se un'applicazione invia un messaggio all'interno di una transazione, il messaggio non viene consegnato alla sua destinazione fino a quando non viene eseguito il commit della transazione. Ciò significa che un'applicazione non può inviare un messaggio e ricevere una risposta al messaggio all'interno della stessa transazione.

È possibile configurare una destinazione in modo che quando un'applicazione invia messaggi ad essa, WebSphere MQ classes for JMS inoltra il messaggio e restituisce il controllo all'applicazione senza determinare se il gestore code ha ricevuto il messaggio in modo sicuro. A volte ci si riferisce a questo come *inserimento asincrono*. Per ulteriori informazioni, vedere “Inserimento asincrono dei messaggi nelle classi IBM WebSphere MQ per JMS” a pagina 922.

Ricezione di messaggi in un'applicazione JMS

Un'applicazione utilizza un consumatore di messaggi per ricevere messaggi. Un sottoscrittore di argomenti durevoli è un consumatore di messaggi che riceve tutti i messaggi inviati a una destinazione, inclusi quelli inviati mentre il consumatore è inattivo. Un'applicazione può selezionare i messaggi che desidera ricevere utilizzando un selettore di messaggi e può ricevere messaggi in modo asincrono utilizzando un listener di messaggi.

Un'applicazione utilizza un oggetto MessageConsumer per ricevere messaggi. Un'applicazione crea un oggetto MessageConsumer per una destinazione specifica, che può essere una coda o un argomento, in modo che tutti i messaggi ricevuti utilizzando il consumer dei messaggi vengano ricevuti dalla stessa destinazione. Di conseguenza, prima che un'applicazione possa creare un oggetto MessageConsumer , deve prima creare un oggetto Coda o Argomento. Per informazioni su come creare un oggetto Coda o Argomento, consultare i seguenti argomenti:

- “Utilizzo di JNDI per richiamare gli oggetti gestiti in una applicazione JMS” a pagina 866
- “Utilizzo delle estensioni JMS IBM JMS” a pagina 867
- “Utilizzo delle estensioni WebSphere MQ JMS” a pagina 874
- “Creazione di destinazioni in una applicazione JMS” a pagina 879

Per creare un oggetto MessageConsumer , un'applicazione utilizza il metodo createConsumer() di un oggetto Session, come mostrato nel seguente esempio:

```
MessageConsumer consumer = session.createConsumer(destination);
```

Il parametro `destination` è un oggetto Coda o Argomento creato in precedenza dall'applicazione.

L'applicazione utilizza quindi il metodo receive () dell'oggetto MessageConsumer per ricevere un messaggio dalla destinazione, come mostrato nel seguente esempio:

```
Message inMessage = consumer.receive(1000);
```

Il parametro sulla chiamata receive () specifica per quanto tempo, in millisecondi, il metodo attende l'arrivo di un messaggio appropriato se non è disponibile alcun messaggio immediatamente. Se si omette

questo parametro, la chiamata si blocca indefinitamente fino all'arrivo di un messaggio appropriato. Se non si desidera che l'applicazione attenda un messaggio, utilizzare invece il metodo receiveNoWait ().

Il metodo receive () restituisce un messaggio di un tipo specifico. Ad esempio, quando un'applicazione riceve un messaggio di testo, l'oggetto restituito dalla chiamata receive () è un oggetto TextMessage .

Tuttavia, il tipo dichiarato di oggetto restituito da una chiamata receive () è un oggetto Messaggio. Pertanto, per estrarre i dati dal corpo di un messaggio appena ricevuto, l'applicazione deve eseguire il cast dalla classe Message alla sottoclasse più specifica, ad esempio TextMessage. Se il tipo di messaggio non è noto, l'applicazione può utilizzare l'operatore `instanceof` per determinare il tipo. È sempre buona norma per un'applicazione determinare il tipo di messaggio prima di eseguire il casting, in modo che gli errori possano essere gestiti correttamente.

Il seguente codice utilizza l'operatore `instanceof` e mostra come estrarre i dati dal corpo di un messaggio di testo:

```
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");
}
```

Se un'applicazione invia un messaggio all'interno di una transazione, il messaggio non viene consegnato alla sua destinazione fino a quando non viene eseguito il commit della transazione. Ciò significa che un'applicazione non può inviare un messaggio e ricevere una risposta al messaggio all'interno della stessa transazione.

Se un utente di messaggi riceve messaggi da una destinazione configurata per la lettura anticipata, tutti i messaggi non persistenti che si trovano nel buffer di lettura anticipata al termine dell'applicazione vengono eliminati.

Nel dominio di pubblicazione / sottoscrizione, JMS identifica due tipi di utente di messaggi, sottoscrittore di argomenti non durevoli e sottoscrittore di argomenti durevoli, descritti nelle due seguenti sezioni.

Sottoscrittori di argomenti non durevoli

Un sottoscrittore di argomenti non durevoli riceve solo i messaggi pubblicati mentre il sottoscrittore è attivo. Una sottoscrizione non durevole inizia quando un'applicazione crea un sottoscrittore di argomenti non durevoli e termina quando l'applicazione chiude il sottoscrittore o quando il sottoscrittore non rientra nell'ambito. Come estensione in WebSphere MQ classes for JMS, un sottoscrittore di argomenti non durevoli riceve anche le pubblicazioni conservate, ma non quando si utilizza una connessione in tempo reale ad un broker.

Per creare un sottoscrittore di argomenti non durevoli, un'applicazione può utilizzare il metodo createConsumer() indipendente dal dominio, specificando un oggetto Argomento come destinazione. In alternativa, un'applicazione può utilizzare il metodo createSubscriber() specifico del dominio, come mostrato nel seguente esempio:

```
TopicSubscriber subscriber = session.createSubscriber(topic);
```

Il parametro `topic` è un oggetto Argomento creato precedentemente dall'applicazione.

Sottoscrittori di argomenti durevoli

Limitazione: Un'applicazione non è in grado di creare sottoscrittori di argomenti durevoli quando si utilizza una connessione in tempo reale a un broker.

Un sottoscrittore di argomenti durevoli riceve tutti i messaggi pubblicati durante la durata di una sottoscrizione durevole. Questi messaggi includono tutti quelli pubblicati mentre il sottoscrittore non è

attivo. Come estensione in WebSphere MQ classes per JMS, un sottoscrittore di argomenti durevoli riceve anche le pubblicazioni conservate.

Per creare un sottoscrittore di argomenti durevoli, un'applicazione utilizza il metodo createDurableSubscriber () di un oggetto Session, come mostrato nel seguente esempio:

```
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001");
```

Nella chiamata createDurableSubscriber (), il primo parametro è un oggetto Topic che l'applicazione ha creato in precedenza e il secondo parametro è un nome utilizzato per identificare la sottoscrizione durevole.

La sessione utilizzata per creare un sottoscrittore di argomenti durevoli deve avere associato un identificativo client. L'identificativo client associato a una sessione è uguale all'identificativo client per il collegamento utilizzato per creare la sessione. L'ID client può essere specificato impostando la proprietà CLIENTID dell'oggetto ConnectionFactory . In alternativa, un'applicazione può specificare l'identificativo del client richiamando il metodo setClientID () dell'oggetto Connection.

Il nome utilizzato per identificare una sottoscrizione duratura deve essere univoco solo all'interno dell'identificativo client e, pertanto, l'identificativo client fa parte dell'identificativo univoco completo di una sottoscrizione durevole. Per continuare a utilizzare una sottoscrizione durevole creata precedentemente, un'applicazione deve creare un sottoscrittore di argomenti durevoli utilizzando una sessione con lo stesso identificativo client di quello associato alla sottoscrizione durevole e utilizzando lo stesso nome sottoscrizione.

Una sottoscrizione durevole viene avviata quando un'applicazione crea un sottoscrittore di argomenti durevoli utilizzando un identificativo client e un nome sottoscrizione per cui attualmente non esiste alcuna sottoscrizione durevole. Tuttavia, una sottoscrizione durevole non termina quando l'applicazione chiude il sottoscrittore dell'argomento durevole. Per terminare una sottoscrizione durevole, un'applicazione deve chiamare il metodo unsubscribe () di un oggetto Session che ha lo stesso identificativo client di quello associato alla sottoscrizione durevole. Il parametro sulla chiamata unsubscribe () è il nome della sottoscrizione, come mostrato nel seguente esempio:

```
session.unsubscribe("D_SUB_000001");
```

L'ambito di una sottoscrizione durevole è un gestore code. Se esiste una sottoscrizione durevole su un gestore code e un'applicazione connessa a un altro gestore code crea una sottoscrizione durevole con lo stesso identificativo client e lo stesso nome sottoscrizione, le due sottoscrizioni durevoli sono completamente indipendenti.

Selettori di messaggi

Un'applicazione può specificare che solo i messaggi che soddisfano determinati criteri vengono restituiti da chiamate receive () successive. Quando si crea un oggetto MessageConsumer , l'applicazione può specificare un'espressione SQL (Structured Query Language) che determina quali messaggi vengono richiamati. Questa espressione SQL è denominata *selettore messaggi*. Il selettore dei messaggi può contenere i nomi dei campi di intestazione del messaggio JMS e le proprietà del messaggio. Per informazioni su come creare un selettore di messaggi, consultare “Selettori di messaggi in JMS” a pagina 809.

Il seguente esempio mostra come un'applicazione può selezionare i messaggi in base a una proprietà definita dall'utente denominata myProp:

```
MessageConsumer consumer;
.
consumer = session.createConsumer(destination, "myProp = 'blue'");
```

La specifica JMS non consente a un'applicazione di modificare il selettore di messaggi di un utente di messaggi. Dopo che un'applicazione crea un consumer di messaggi con un selettore di messaggi, il selettore di messaggi rimane per la vita di tale consumer. Se un'applicazione richiede più di un selettore di messaggi, l'applicazione deve creare un consumer di messaggi per ciascun selettore di messaggi.

Tenere presente che, quando un'applicazione è connessa a un gestore code versione 7, la proprietà MSGSELECTION della factory di connessione non ha alcun effetto. Per ottimizzare le prestazioni, tutta la selezione dei messaggi viene eseguita dal gestore code.

Soppressione delle pubblicazioni locali

Un'applicazione può creare un consumer di messaggi che ignora le pubblicazioni pubblicate sulla propria connessione del consumer. L'applicazione esegue questa operazione impostando il terzo parametro su una chiamata createConsumer() a `true`, come mostrato nel seguente esempio:

```
MessageConsumer consumer = session.createConsumer(topic, null, true);
```

Su una chiamata createDurableSubscriber (), l'applicazione esegue questa operazione impostando il quarto parametro su `true`, come mostrato nel seguente esempio

```
String selector = "company = 'IBM'";
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001",
                                                              selector, true);
```

Consegna asincrona dei messaggi

Un'applicazione può ricevere messaggi in modo asincrono registrando un listener di messaggi con un utente di messaggi. Il listener dei messaggi ha un metodo denominato onMessage, che viene richiamato in maniera asincrona quando è disponibile un messaggio adatto e il cui scopo è elaborare il messaggio. Il seguente codice illustra il meccanismo:

```
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
```

Un'applicazione può utilizzare una sessione per ricevere i messaggi in modo sincrono utilizzando le chiamate receive () o per ricevere i messaggi in modo asincrono utilizzando i listener di messaggi, ma non per entrambi. Se un'applicazione deve ricevere messaggi in modo sincrono e asincrono, deve creare sessioni separate.

Una volta impostata una sessione per ricevere i messaggi in modo asincrono, non è possibile richiamare i seguenti metodi su quella sessione o sugli oggetti creati da quella sessione:

- MessageConsumer.receive ()
- MessageConsumer.receive (lungo)
- MessageConsumer.receiveNoWait ()
- Session.acknowledge()

- MessageProducer.send (Destinazione, Messaggio)
- MessageProducer.send (Destinazione, Messaggio, int, int, long)
- MessageProducer.send (Messaggio)
- MessageProducer.send (Messaggio, int, int, long)
- Session.commit()
- Session.createBrowser(coda)
- Session.createBrowser(Coda, Stringa)
- Session.createBytesMessage()
- Session.createConsumer(Destinazione)
- Session.createConsumer(Destinazione, stringa, booleano)
- Session.createDurableSubscriber(Argomento, Stringa)
- Session.createDurableSubscriber(Argomento, stringa, stringa, booleano)
- Session.createMapMessage()
- Session.createMessage()
- Session.createObjectMessage()
- Session.createObjectMessage(Serializzabile)
- Session.createProducer(Destinazione)
- Session.createQueue(Stringa)
- Session.createStreamMessage()
- Session.createTemporaryQueue()
- Session.createTemporaryTopic()
- Session.createTextMessage()
- Session.createTextMessage(Stringa)
- Session.createTopic()
- Session.getAcknowledgeMode()
- Session.getMessageListener()
- Session.getTransacted()
- Session.rollback()
- Session.unsubscribe(Stringa)

Se viene richiamato uno di questi metodi, una JMSException contenente il messaggio:

```
JMSCC0033: Una chiamata di metodo sincrono non è consentita quando una sessione viene utilizzata
in modo asincrono: 'nome metodo'
```

viene generato.

Ricezione di messaggi non elaborabili

Un'applicazione può ricevere un messaggio che non è possibile elaborare. Ci possono essere diversi motivi per cui il messaggio non può essere elaborato, ad esempio il messaggio potrebbe avere un formato non corretto. Tali messaggi sono descritti come messaggi dannosi e richiedono una gestione speciale per evitare che il messaggio venga elaborato in modo ricorsivo.

Per i dettagli su come gestire i messaggi dannosi, consultare “Gestione dei messaggi non elaborabili in IBM WebSphere MQ classes for JMS” a pagina 891.

V 7.5.0.8 Richiamo dei dati utente di sottoscrizione

Se i messaggi che un'applicazione IBM WebSphere MQ classes for JMS sta utilizzando da una coda vengono inseriti da una sottoscrizione duratura definita amministrativamente, l'applicazione deve

accedere alle informazioni sui dati utente associate alla sottoscrizione. Queste informazioni vengono aggiunte al messaggio come proprietà.

Da Version 7.5.0, Fix Pack 8, quando un messaggio viene utilizzato da una coda che contiene un'intestazione RFH2 con la cartella MQPS, il valore associato alla chiave Sud, se presente, viene aggiunto come proprietà Stringa all'oggetto Messaggio JMS restituito all'applicazione IBM WebSphere MQ classes for JMS . Per abilitare il richiamo di questa proprietà dal messaggio, è possibile utilizzare la costante JMS_IBM_SUBSCRIPTION_USER_DATA nell'interfaccia JmsConstants con il metodo javax.jms.Message.getStringProperty(java.lang.String) per richiamare i dati utente della sottoscrizione.

Nel seguente esempio, una sottoscrizione duratura di gestione viene definita utilizzando il comando 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')
```

Le copie dei messaggi pubblicati nella stringa di argomenti PUBLIC vengono inserite nella coda, MY.SUBSCRIPTION.Q. I dati utente associati alla sottoscrizione durevole vengono quindi aggiunti come una proprietà al messaggio, memorizzato nella cartella MQPS dell'intestazione RFH2 con la chiave Sud.

L'applicazione IBM WebSphere MQ classes for JMS può richiamare:

```
javax.jms.Message.getStringProperty(JmsConstants.JMS_IBM_SUBSCRIPTION_USER_DATA);
```

Viene quindi restituita la seguente stringa:

```
Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q
```

.

Concetti correlati

“L'intestazione MQRFH2 e JMS” a pagina 813

Attività correlate

Definizione di una sottoscrizione di gestione

Riferimenti correlati

DEFINE SUB

Interfaccia JmsConstants

Chiusura di un'applicazione WebSphere MQ classes per JMS

È importante che un'applicazione WebSphere MQ classes for JMS chiuda esplicitamente alcuni oggetti JMS prima dell'arresto. I finalizer potrebbero non essere richiamati, quindi non fare affidamento su di essi per liberare risorse. Non consentire l'arresto di un'applicazione con la traccia compressa attiva.

La raccolta di dati inutilizzati da sola non può rilasciare tutte le risorse WebSphere MQ classes per JMS e WebSphere MQ in modo tempestivo, specialmente se un'applicazione crea molti oggetti JMS di breve durata al livello di sessione o inferiore. È quindi importante che un'applicazione chiuda un oggetto Connection, Session, MessageConsumero MessageProducer quando non è più richiesto.

Se un'applicazione termina senza chiudere una connessione, si verifica un rollback implicito per tutte le sessioni transazionali della connessione. Per assicurarsi che tutte le modifiche apportate dall'applicazione siano sottoposte a commit, chiudere esplicitamente la connessione prima di chiudere l'applicazione.

Non utilizzare i finalizer in un'applicazione per chiudere gli oggetti JMS. Poiché i finalizer potrebbero non essere richiamati, le risorse potrebbero non essere liberate. Quando una connessione viene chiusa, vengono chiuse tutte le sessioni da essa create. Allo stesso modo, i MessageConsumers e MessageProducers creati da una sessione vengono chiusi quando la sessione viene chiusa. Tuttavia, considerare la chiusura esplicita di Sessioni, MessageConsumerse MessageProducers per garantire che le risorse vengano liberate in modo tempestivo.

Se la compressione della traccia è attivata, è probabile che System.Halt() arresti e terminazioni JVM anomale e non controllate provochino un file di traccia danneggiato. Se possibile, disattivare la

funzionalità di traccia una volta raccolte le informazioni di traccia necessarie. Se si sta eseguendo la traccia di un'applicazione fino a una fine anomala, utilizzare l'output di traccia non compresso.

Gestione dei messaggi non elaborabili in IBM WebSphere MQ classes for JMS

Un messaggio non elaborabile è quello che non può essere elaborato da un'applicazione MDB di ricezione. Se viene rilevato un messaggio non elaborabile, gli oggetti JMS MessageConsumer e ConnectionConsumer possono riaccodarlo in base a due proprietà della coda, BOQNAME e BOTHRESH.

A volte, un messaggio formattato in modo errato arriva su una coda. In questo contesto, un formato non corretto indica che l'applicazione ricevente non può elaborare correttamente il messaggio. Un messaggio di questo tipo può causare il malfunzionamento dell'applicazione ricevente e il backout di questo messaggio formattato in modo errato. Il messaggio può essere inviato ripetutamente alla coda di input e ripetutamente ripristinato dall'applicazione. Questi messaggi sono noti come *messaggi non elaborabili*. L'oggetto JMS MessageConsumer rileva i messaggi non elaborabili e li reinvia verso una destinazione alternativa.

Il gestore code IBM WebSphere MQ conserva un record del numero di volte in cui è stato eseguito il backout di ciascun messaggio. Quando questo valore raggiunge un valore di soglia configurabile, l'utente del messaggio riaccoda il messaggio ad una coda di backout denominata. Se questa riaccodamento non riesce per qualsiasi motivo, il messaggio viene rimosso dalla coda di input e riaccodato alla coda di messaggi non recapitabili o eliminato. Per ulteriori dettagli, vedere “Rimozione dei messaggi dalla coda in ASF” a pagina 930.

Esiste una differenza tra il modo in cui i messaggi dannosi vengono riaccodati da MessageConsumers e ConnectionConsumers. ConnectionConsumers sono in grado di riaccodare i messaggi non elaborabili senza influire sulla consegna dei messaggi. Il processo di riaccodamento si svolge al di fuori di qualsiasi unità di lavoro associata alla consegna effettiva del messaggio al codice dell'applicazione. Ciò è possibile a causa della natura multi - thread dell'operazione ConnectionConsumer .

I MessageConsumers, tuttavia, sono a thread singolo al di sotto del livello di sessione e qualsiasi accodamento di messaggi dannosi avviene all'interno dell'unità di lavoro corrente. Ciò non influisce sul funzionamento dell'applicazione, tuttavia quando i messaggi non elaborabili vengono riaccodati in una sessione transagata o di riconoscimento client, l'azione di riaccodamento stessa non viene sottoposta a commit fino a quando non viene eseguito il commit dell'unità di lavoro corrente dal codice dell'applicazione o, se appropriato, dal codice del contenitore dell'applicazione.

Gli oggetti JMS ConnectionConsumer gestiscono i messaggi non elaborabili nello stesso modo e utilizzano le stesse proprietà della coda. Se più utenti della connessione stanno monitorando la stessa coda, è possibile che il messaggio non elaborabile possa essere consegnato a un'applicazione più volte del valore di soglia prima che si verifichi la riaccodamento. Questo comportamento è dovuto al modo in cui i singoli utenti della connessione monitorano le code e riaccodano i messaggi dannosi.

Il valore di soglia ed il nome della coda di backout sono attributi di una coda IBM WebSphere MQ . I nomi degli attributi sono BackoutThreshold e BackoutRequeueQName. La coda a cui si applicano è la seguente:

- Per la messaggistica point - to - point, questa è la coda locale sottostante. Ciò è importante quando gli utenti dei messaggi e dei collegamenti utilizzano gli alias della coda.
- Per la messaggistica di pubblicazione / sottoscrizione in modalità normale del fornitore di messaggistica IBM WebSphere MQ , la coda gestita dell'argomento viene creata dalla coda modello.
- Per la messaggistica di pubblicazione / sottoscrizione in modalità di migrazione del provider di messaggistica IBM WebSphere MQ , è la coda CCSUB definita nell'oggetto factory TopicConnectiono la coda CCDSUB definita nell'oggetto argomento.

IBM WebSphere MQ classes for JMS interroga il QName BackoutThreshold e BackoutRequeuedella coda. È quindi necessario concedere l'accesso inquire sulla coda all'utente che esegue l'applicazione.

V 7.5.0.9 Se la coda di destinazione è una coda cluster, le autorizzazioni necessarie dipendono dalla versione di IBM WebSphere MQ classes for JMS utilizzata:

- Quando si utilizza IBM WebSphere MQ classes for JMS per Version 7.5.0, Fix Pack 9 più una fix temporanea per APAR IT26482, è richiesto l'accesso alla richiesta.

- Per tutte le altre versioni, concedere inquire, sfogliare e ottenere l'accesso.

Per impostare gli attributi QName BackoutThreshold e BackoutRequeue, immettere il seguente comando MQSC:

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

Se l'attributo BackoutThreshold è impostato su un valore diverso da zero, per evitare un comportamento imprevisto impostare l'attributo QName BackoutRequeuesu un nome coda valido.

Per la messaggistica di pubblicazione / sottoscrizione, se il sistema crea una coda dinamica per ciascuna sottoscrizione, questi valori di attributo vengono ottenuti dalla coda modello IBM WebSphere MQ classes for JMS , SYSTEM.JMS.MODEL.QUEUE. Per modificare queste impostazioni, utilizzare:

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

Se il valore della soglia di backout è zero, la gestione dei messaggi non elaborabili è disabilitata e i messaggi non elaborabili rimangono sulla coda di input. Altrimenti, quando il conteggio di backout raggiunge il valore di soglia, il messaggio viene inviato alla coda di backout indicata. Se il conteggio di backout raggiunge il valore di soglia, ma il messaggio non può andare alla coda di backout, il messaggio viene inviato alla coda di messaggi non recapitabili o viene eliminato. Questa situazione si verifica se la coda di backout non è definita o se l'oggetto MessageConsumer non può inviare il messaggio alla coda di backout. Consultare “Rimozione dei messaggi dalla coda in ASF” a pagina 930 per ulteriori dettagli.

Quando un messaggio viene riaccodato alla coda di riaccodamento di backout, alcuni valori del campo in MQMD (Message Descriptor) del messaggio cambiano. Per i dettagli sul formato di MQMD, vedere MQMD - Descrittore messaggi .

I campi MQMD riportati di seguito cambiano valore quando il messaggio va nella coda di backout.

- PutDate viene aggiornato alla data in cui va nella coda di riaccodamento di backout.
- PutTime viene aggiornata con l'ora in cui passa alla coda di riaccodamento di backout.
- Il conteggio di backout viene reimpostato su zero.
- La scadenza del messaggio viene aggiornata per riflettere la scadenza rimanente al momento in cui il messaggio originale è stato ricevuto dall'applicazione JMS.

I valori nei seguenti campi rimangono gli stessi quando il messaggio va sulla coda di backout:

- StructId
- Versione
- Prospetto
- MessageType
- Feedback
- Codifica
- CodedCharSetId
- MsgId
- CorrelId
- ReplyToQ
- ReplyToQMgr
- Formato
- Persistenza
- Priorit...

Eccezioni in IBM WebSphere MQ classes for JMS

Un'applicazione IBM WebSphere MQ classes for JMS deve essere in grado di gestire le eccezioni generate da chiamate API JMS o consegnate a un gestore eccezioni.

IBM WebSphere MQ classes for JMS riporta problemi di runtime generando eccezioni. JMSException è la classe root per le eccezioni generate dai metodi JMS e la cattura delle eccezioni JMSException fornisce un metodo generico di gestione di tutte le eccezioni relative a JMS.

Ogni eccezione JMSException contiene le seguenti informazioni:

- Un messaggio di eccezione specifico del provider, che un'applicazione ottiene richiamando il metodo Throwable.getMessage().
- Un codice di errore specifico del fornitore, che un'applicazione ottiene richiamando il metodo JMSException.getErrorCode().
- Un'eccezione collegata. Un'eccezione generata da una chiamata API JMS è spesso il risultato di un problema di livello inferiore, riportato da un'altra eccezione collegata a questa eccezione. Un'applicazione ottiene un'eccezione collegata richiamando il metodo JMSException.getLinkedException() o il metodo Throwable.getCause().

La maggior parte delle eccezioni generate da IBM WebSphere MQ classes for JMS sono istanze di sottoclassi di JMSException. Tali sottoclassi implementano l'interfaccia com.ibm.msg.client.jms.JmsExceptionDetail , che fornisce ulteriori informazioni:

- Una descrizione del messaggio di eccezione, che un'applicazione ottiene richiamando il metodo JmsExceptionDetail.getExplanation().
- Una risposta utente consigliata all'eccezione, che un'applicazione ottiene richiamando il metodo JmsExceptionDetail.getUserAction().
- Le chiavi per il messaggio vengono inserite nel messaggio di eccezione. Un'applicazione ottiene un iteratore per tutte le chiavi richiamando il metodo JmsExceptionDetail.getKeys().
- Il messaggio viene inserito nel messaggio di eccezione. Ad esempio, un inserimento di un messaggio potrebbe essere il nome della coda che ha causato l'eccezione e potrebbe essere utile per un'applicazione per poter accedere a tale nome. Un'applicazione ottiene l'inserimento del messaggio corrispondente a una chiave specificata richiamando il metodo JmsExceptionDetail.getValue().

Tutti i metodi nell'interfaccia Dettagli JmsExceptionpotrebbero restituire un valore null se non sono disponibili dettagli.

Ad esempio, se un'applicazione tenta di creare un produttore di messaggi per una coda IBM WebSphere MQ che non esiste, viene generata un'eccezione con le seguenti informazioni:

```
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.
```

L'eccezione generata, com.ibm.msg.client.jms.DetailedInvalidDestinationException, è una sottoclasse di javax.jms.InvalidDestinationException e implementa l'interfaccia com.ibm.msg.client.jms.JmsExceptionDetail .

Eccezioni collegate

Un'eccezione collegata fornisce ulteriori informazioni su un problema di runtime. Pertanto, per ogni eccezione JMSException generata, un'applicazione deve controllare l'eccezione collegata. L'eccezione collegata stessa potrebbe avere un'altra eccezione collegata, e quindi le eccezioni collegate formano una catena che riporta al problema sottostante originale. Un'eccezione collegata viene implementata utilizzando il meccanismo di eccezione concatenato della classe java.lang.Throwable , e un'applicazione ottiene un'eccezione collegata richiamando il metodo Throwable.getCause(). Per un'eccezione JMSException, il metodo getLinkedException () in realtà delega al metodo Throwable.getCause().

Ad esempio, se un'applicazione specifica un numero di porta non corretto durante la connessione a un gestore code, le eccezioni formano la seguente catena:

```
com.ibm.msg.client.jms.DetailIllegalStateException
   |
   +--->com.ibm.mq.MQException
           |
           +--->com.ibm.mq.jmqi.JmqiException
                   |
                   +--->java.net.ConnectionException
```

Di solito, ogni eccezione in una catena viene generata da un livello differente nel codice. Ad esempio, le eccezioni nella catena precedente vengono generate dai seguenti livelli:

- La prima eccezione, un'istanza di una sottoclasse di JMSException, viene generata dal livello comune in IBM WebSphere MQ classes for JMS.
- L'eccezione successiva, un'istanza di com.ibm.mq.MQException, viene generata dal provider di messaggistica IBM WebSphere MQ .
- L'eccezione successiva, un'istanza di com.ibm.mq.jmqi.JmqiException, viene generata dall'interfaccia Java comune per MQI.
- L'eccezione finale, un'istanza di java.net.ConnectionException, viene generata dalla libreria di classi Java.

Per ulteriori informazioni sull'architettura a livelli di IBM WebSphere MQ classes for JMS, consultare “Classi IBM WebSphere MQ per l'architettura JMS” a pagina 799.

Utilizzando un codice simile al seguente, un'applicazione può eseguire l'iterazione attraverso questa catena per estrarre tutte le informazioni appropriate:

```
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();
    }
}
```

Si noti che un'applicazione deve sempre controllare il tipo di ogni eccezione in una catena poiché il tipo di eccezione può variare e le eccezioni di tipi diversi incapsulano informazioni diverse.

Come ottenere informazioni specifiche su un problema IBM WebSphere MQ

Le istanze di com.ibm.mq.MQException e com.ibm.mq.jmqi.JmqiException incapsulano IBM WebSphere MQ informazioni specifiche su un problema.

Un'eccezione MQException contiene le seguenti informazioni:

- Un codice di completamento che un'applicazione ottiene richiamando il metodo getCompCode ()
- Un codice motivo, che un'applicazione ottiene richiamando il metodo getReason()

Un'eccezione JmqiException include anche un codice di completamento e un codice motivo. Inoltre, tuttavia, un'eccezione JmqiException incapsula le informazioni in un messaggio AMQ*nnnn* o CSQ*nnnn* , se associato all'eccezione. Richiamando i metodi appropriati dell'eccezione, un'applicazione può ottenere i vari componenti di questo messaggio, come la severità, la spiegazione e la risposta dell'utente.

Per esempi su come utilizzare i metodi menzionati in questa sezione, consultare il codice di esempio in “Eccezioni collegate” a pagina 893.

Aggiornamento da versioni precedenti di IBM WebSphere MQ classes for JMS

Rispetto alle precedenti versioni di IBM WebSphere MQ classes for JMS, la maggior parte dei codici di errore e dei messaggi di eccezione sono stati modificati nella versione 7. Il motivo di queste modifiche è che IBM WebSphere MQ classes for JMS ora ha un'architettura a livelli e vengono generate eccezioni da diversi livelli nel codice.

Ad esempio, se un'applicazione tenta di connettersi a un gestore code che non esiste, una versione precedente di IBM WebSphere MQ classes for JMS ha generato un'eccezione JMSException con le seguenti informazioni:

```
MQJMS2005: Failed to create MQQueueManager for 'localhost:QM_test'.
```

Questa eccezione conteneva un'eccezione MQException collegata con le seguenti informazioni:

```
MQJE001: Completion Code 2, Reason 2058
```

Per confronto nelle stesse circostanze, la Versione 7 di IBM WebSphere MQ classes for JMS genera un'eccezione JMSException con le seguenti informazioni:

```
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.
```

Questa eccezione contiene un'eccezione MQException collegata con le seguenti informazioni:

```
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
```

Se l'applicazione analizza o verifica i messaggi di eccezione restituiti dal metodo Throwable.getMessage() o i codici di errore restituiti dal metodo JMSException.getErrorCode() e si sta eseguendo l'aggiornamento da una release precedente alla versione 7, è probabile che l'applicazione debba essere modificata per poter utilizzare la versione 7 di IBM WebSphere MQ classes for JMS.

Listener di eccezione

Un'applicazione può registrare un listener di eccezioni con un oggetto Connection. Successivamente, se si verifica un problema che rende la connessione inutilizzabile, IBM WebSphere MQ classes for JMS invia un'eccezione al listener delle eccezioni richiamando il metodo onException(). L'applicazione ha quindi l'occasione di ristabilire la connessione.

V 7.5.0.8 APAR IT14820, incluso da IBM WebSphere MQ Version 7.5.0, Fix Pack 8, ha corretto un difetto in cui un ExceptionListener JMS dell'applicazione non veniva richiamato per le eccezioni non interrotte di connessione (ad esempio MQRC_GET_INHIBITED) anche se la proprietà ASYNC_EXCEPTIONS sul factory di connessione JMS utilizzato dall'applicazione era impostata su ASYNC_EXCEPTIONS_ALL. Questo era il valore predefinito prima di Version 7.5.0, Fix Pack 8.

V 7.5.0.8 Per mantenere il comportamento per le applicazioni JMS correnti che configurano un JMS MessageListener e un JMS ExceptionListenere per garantire che il IBM WebSphere MQ classes for JMS sia congruente con la specifica JMS, il valore predefinito per la proprietà ConnectionFactory JMS ASYNC_EXCEPTIONS_CONNECTIONBROKEN per IBM WebSphere MQ classes for JMS. Di conseguenza, per impostazione predefinita solo le eccezioni che corrispondono ai codici di errore di connessione interrotta vengono recapitate a un ExceptionListener JMS.

V 7.5.0.8 Da Version 7.5.0, Fix Pack 8, i IBM WebSphere MQ classes for JMS sono stati aggiornati in modo che le JMSExceptions relative agli errori di non interruzione della connessione, che si verificano durante la consegna del messaggio ai consumatori di messaggi asincroni, vengano ancora consegnate a un ExceptionListener registrato quando il ConnectionFactory JMS utilizzato dall'applicazione ha la proprietà ASYNC_EXCEPTIONS impostata sul valore ASYNC_EXCEPTIONS_ALL.

V 7.5.0.8 Per ulteriori informazioni sulle modifiche apportate ai listener di eccezioni per Version 7.5.0, Fix Pack 8 e sul motivo per cui le modifiche sono state apportate da release precedenti, vedere JMS: Exception listener changes in Version 7.5.

Per qualsiasi altro tipo di problema, viene generata un'eccezione JMSException dalla chiamata API JMS corrente.

Se un'applicazione non registra un listener di eccezioni con un oggetto Connection, tutte le eccezioni che sarebbero state consegnate al listener di eccezioni vengono scritte nel log IBM WebSphere MQ classes for JMS .

Riferimenti correlati

ECCEZIONE asincrona

Registrazione degli errori in WebSphere MQ classes per JMS

Le informazioni sui problemi di runtime che potrebbero richiedere un'azione correttiva da parte dell'utente vengono scritte nel log delle classi WebSphere MQ per JMS.

Ad esempio, se un'applicazione tenta di impostare una proprietà di un factory di connessione, ma il nome della proprietà non viene riconosciuto, WebSphere MQ classes per JMS scrive le informazioni sul problema nel proprio log.

Per impostazione predefinita, il file contenente il log è denominato mqjms.log e si trova nella directory di lavoro corrente. Tuttavia, è possibile modificare il nome e l'ubicazione del file di log impostando la proprietà com.ibm.msg.client.commonservices.log.outputName nel file di configurazione WebSphere MQ classes per JMS. Per informazioni sul file di configurazione WebSphere MQ classes for JMS, consultare “Il file di configurazione IBM WebSphere MQ classes for JMS” a pagina 728e per ulteriori dettagli sui valori validi per la proprietà com.ibm.msg.client.commonservices.log.outputName , consultare “Registrazione e IBM WebSphere MQ classes for JMS” a pagina 797.

First failure support technology (FFST) in WebSphere MQ classes per JMS

Se si verifica un errore interno grave all'interno di WebSphere MQ classes per JMS, vengono generate le informazioni sulla tecnologia di supporto del primo errore (FFST).

Le informazioni FFST vengono scritte in un file denominato JMSC*nnnn*.FDC, dove *nnnn* è un numero di quattro cifre. Questo file si trova in una directory denominata FFDC, che è una sottodirectory della directory in cui viene scritto l'output di traccia. Per impostazione predefinita, l'output di traccia viene scritto nella directory di lavoro corrente, ma è possibile reindirizzarlo in una directory diversa impostando la proprietà **com.ibm.msg.client.commonservices.trace.outputName** nel file di configurazione WebSphere MQ classes per JMS. Per informazioni sul file di configurazione di WebSphere MQ classes per JMS, consultare "Il file di configurazione IBM WebSphere MQ classes for JMS" a pagina 728.

Se la traccia è abilitata quando vengono generate le informazioni FFST , anche le informazioni FFST vengono scritte nel file di traccia. Per ulteriori informazioni sulla traccia dei programmi JMS, consultare Traccia delle applicazioni IBM WebSphere MQ classes for JMS.

Per eliminare la produzione di file FFDC, impostare la proprietà **com.ibm.msg.client.commonservices.ffst.suppress**nel modo seguente:

0
Output di tutti i file FFDC (predefinito).

-1
Emettere solo i primi file FFDC di un determinato tipo.

intero
Eliminare tutti i file FFDC tranne quelli che sono un multiplo di questo numero.

Accesso alle funzioni WebSphere MQ da un'applicazione WebSphere MQ per JMS

WebSphere MQ classes for JMS fornisce funzioni per sfruttare una serie di funzioni di WebSphere MQ.

Attenzione: Queste funzioni sono al di fuori della specifica JMS o, in alcuni casi, violano la specifica JMS. Se li si utilizza, è improbabile che l'applicazione sia compatibile con altri fornitori JMS. Quelle funzioni che non sono conformi alla specifica JMS sono etichettate con un avviso di attenzione.

Lettura e scrittura del descrittore del messaggio da un'applicazione WebSphere MQ classes per JMS

Si controlla la possibilità di accedere al descrittore del messaggio (MQMD) impostando le proprietà su una destinazione e un messaggio.

Alcune applicazioni WebSphere MQ richiedono l'impostazione di valori specifici nell'MQMD dei messaggi inviati. WebSphere MQ classes for JMS fornisce attributi di messaggio che consentono alle applicazioni JMS di impostare i campi MQMD e quindi abilitare le applicazioni JMS a "guidare" WebSphere MQ .

È necessario impostare la proprietà dell'oggetto di destinazione WMQ_MQMD_WRITE_ENABLED su true affinché l'impostazione delle proprietà MQMD abbia effetto. È quindi possibile utilizzare i metodi di impostazione della proprietà del messaggio (ad esempio setStringProperty) per assegnare valori ai campi MQMD. Tutti i campi MQMD sono esposti tranne StrucId e Version; BackoutCount può essere letto ma non scritto.

Questo esempio determina l'inserimento di un messaggio in una coda o in un argomento con MQMD.UserIdentifier impostato suJoeBloggs".

```
// 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
// ...
```

È necessario impostare WMQ_MQMD_MESSAGE_CONTEXT prima di impostare JMS_IBM_MQMD_UserIdentifier. Per ulteriori informazioni sull'utilizzo di WMQ_MQMD_MESSAGE_CONTEXT, consultare “Proprietà oggetto messaggio JMS” a pagina 900.

Allo stesso modo, è possibile estrarre il contenuto dei campi MQMD impostando WMQ_MQMD_READ_ENABLED su true prima di ricevere un messaggio e quindi utilizzando i metodi get del messaggio, come la proprietà getString. Tutte le proprietà ricevute sono di sola lettura.

In questo esempio viene visualizzato il campo *valore* contenente il valore di MQMD.ApplIdentityData di un messaggio ricevuto da una coda o da un argomento.

```
// 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");
```

Proprietà oggetto di destinazione JMS
Due proprietà dell'oggetto Destination controllano l'accesso a MQMD da JMS e una terza controlla il contesto del messaggio.

Tabella 124. Nomi e descrizioni delle proprietà

Proprietà	**Forma breve**	**Descrizione**
WMQ_MQMD_WRITE_ENABLED	MDW	Se un'applicazione JMS può impostare i valori dei campi MQMD
WMQ_MQMD_READ_ENABLED	MDR	Se un'applicazione JMS può estrarre i valori dei campi MQMD
WMQ_MQMD_MESSAGE_ CONTESTO	MDCTX	Quale livello di contesto del messaggio deve essere impostato dall'applicazione JMS. L'applicazione deve essere in esecuzione con l'autorizzazione di contesto appropriata per rendere effettiva questa proprietà

Tabella 125. Nomi proprietà, valori e metodi set

Proprietà	Valori validi nello strumento di gestione (valori predefiniti in grassetto)	Valori validi nei programmi	Imposta metodo
WMQ_MQMD_WRITE_ENABLED	• **NO** Tutte le proprietà JMS_IBM_MQMD* vengono ignorate e i relativi valori non vengono copiati nella struttura MQMD sottostante. • Sì Le proprietà JMS_IBM_MQMD* vengono elaborate. I loro valori vengono copiati nella struttura MQMD sottostante.	• **Falso** • Sì	setMQMDWriteabilitato
WMQ_MQMD_READ_ABILITATO	• **NO** Quando si inviano messaggi, le proprietà JMS_IBM_MQMD* di un messaggio inviato non vengono aggiornate per riflettere i valori dei campi aggiornati in MQMD. Alla ricezione dei messaggi, nessuna delle proprietà JMS_IBM_MQMD* è disponibile sul messaggio ricevuto, anche se il mittente ne ha impostate alcune o tutte. • Sì Quando si inviano i messaggi, tutte le proprietà JMS_IBM_MQMD* su un messaggio inviato vengono aggiornate per riflettere i valori del campo aggiornati in MQMD, inclusi quelli che il mittente non ha impostato esplicitamente. Alla ricezione dei messaggi, tutte le proprietà JMS_IBM_MQMD* sono disponibili sul messaggio ricevuto, comprese quelle non impostate esplicitamente dal mittente.	• **Falso** • Sì	setMQMDReadAbilitata

Tabella 125. Nomi proprietà, valori e metodi set (Continua)			
Proprietà	**Valori validi nello strumento di gestione (valori predefiniti in grassetto)**	**Valori validi nei programmi**	**Imposta metodo**
WMQ_MQMD _CONTESTO_MESSAGGIO	• **Predefinito** La chiamata API MQOPEN e la struttura MQPMO non specificano opzioni di contesto del messaggio esplicite • SET_IDENTITY_CONTEXT La chiamata API MQOPEN specifica l'opzione di contesto del messaggio MQOO_SET_IDENTITY_CONTEXT e la struttura di MQPMO specifica MQPMO_SET_IDENTITY_CONTEXT • IMPOSTA_TUTTI_CONTESTO La chiamata API MQOPEN specifica l'opzione di contesto del messaggi MQOO_SET_ALL_CONTEXT e la struttura MQPM specifica MQPMO_SET_ALL_CONTEXT	• **WMQ_MDCTX_DEFAULT** • WMQ_MDCTX_SET_IDENTITY_CONTEXT • WMQ_MDCTX_SET_ALL_CONTEXT	Contesto setMQMDMessage

Proprietà oggetto messaggio JMS
Le proprietà dell'oggetto del messaggio con prefisso JMS_IBM_MQMD consentono di impostare o leggere il campo MQMD corrispondente.

invio di messaggi

Tutti i campi MQMD tranne StrucId e Version sono rappresentati. Queste proprietà fanno riferimento solo ai campi MQMD; dove una proprietà si verifica sia nell'intestazione MQMD che MQRFH2 , la versione in MQRFH2 non è impostata o estratta.

È possibile impostare una qualsiasi di queste proprietà, ad eccezione di JMS_IBM_MQMD_BackoutCount. Qualsiasi valore impostato per JMS_IBM_MQMD_BackoutCount viene ignorato.

Se una proprietà ha una lunghezza massima e si fornisce un valore troppo lungo, il valore viene troncato.

Per alcune proprietà, è necessario impostare anche la proprietà WMQ_MQMD_MESSAGE_CONTEXT sull'oggetto Destinazione. L'applicazione deve essere in esecuzione con l'autorizzazione di contesto appropriata perché questa proprietà diventi effettiva. Se non si imposta WMQ_MQMD_MESSAGE_CONTEXT su un valore appropriato, il valore della proprietà viene ignorato. Se si imposta WMQ_MQMD_MESSAGE_CONTEXT su un valore appropriato ma non si dispone di un'autorizzazione di contesto sufficiente per il gestore code, viene emessa una JMSException. Le proprietà che richiedono valori specifici di WMQ_MQMD_MESSAGE_CONTEXT sono le seguenti.

Le seguenti proprietà richiedono l'impostazione di WMQ_MQMD_MESSAGE_CONTEXT su WMQ_MDCTX_SET_IDENTITY_CONTEXT o WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- Dati JMS_IBM_MQMD_ApplIdentity

Le proprietà riportate di seguito richiedono che WMQ_MQMD_MESSAGE_CONTEXT sia impostato su WMQ_MDCTX_SET_ALL_CONTEXT:

- Tipo JMS_IBM_MQMD_PutAppl

- Nome JMS_IBM_MQMD_PutAppl
- JMS_IBM_MQMD_PutDate
- PutTime JMS_IBM_MQMD_
- JMS_IBM_MQMD_ApplOriginDati

ricezione di messaggi

Tutte queste proprietà sono disponibili su un messaggio ricevuto se la proprietà WMQ_MQMD_READ_ENABLED è impostata su true, indipendentemente dalle proprietà effettive che l'applicazione di produzione ha impostato. Un'applicazione non può modificare le proprietà di un messaggio ricevuto a meno che tutte le proprietà non vengano prima cancellate, in base alla specifica JMS. Il messaggio ricevuto può essere inoltrato senza modificare le proprietà.

Attenzione: Se la propria applicazione riceve un messaggio da una destinazione con la proprietà WMQ_MQMD_READ_ENABLED impostata su true e lo inoltra a una destinazione con WMQ_MQMD_WRITE_ENABLED impostata su true, tutti i valori del campo MQMD del messaggio ricevuto vengono copiati nel messaggio inoltrato.

Tabella delle proprietà

Questa tabella elenca le proprietà dell'oggetto Message che rappresentano i campi MQMD. Consultare i collegamenti per una descrizione completa dei campi e dei valori consentiti.

Tabella 126. Nomi, descrizioni e tipi di proprietà

Proprietà	**Descrizione**	**Tipo Java**	**Link alla descrizione completa**
Report MQMD_IBM_JMS	Opzioni per messaggi di report	Intero	Prospetto
JMS_IBM_MQMD_MsgType	Tipo messaggio	Intero	MsgType
JMS_IBM_MQMD_Expiry	Durata messaggio	Intero	Scadenza
Feedback MQMD_IBM_JMS	Feedback o codice motivo	Intero	Feedback
Codifica JMS_IBM_MQMD_	Codifica numerica dei dati di messaggio	Intero	Codifica
JMS_IBM_MQMD_CodedCharSetId	CSID (Character set identifier) dei dati del messaggio	Intero	CodedCharSetId
Formato_MQMD_IBM_JMS	Nome formato dei dati di messaggio	Stringa	Formato
JMS_IBM_MQMD_Priority [1]	Priorità messaggio	Intero	Priorit...
Persistenza JMS_IBM_MQMD_	Persistenza messaggio	Intero	Persistenza
JMS_IBM_MQMD_MsgId [2]	ID messaggio	Oggetto (byte []) [4]	MsgId
JMS_IBM_MQMD_CorrelId [3]	Identificativo di correlazione	Oggetto (byte []) [4]	CorrelId
JMS_IBM_MQMD_BackoutCount	Contatore backout	Intero	BackoutCount
JMS_IBM_MQMD_ReplyToQ	Nome della coda di risposta	Stringa	ReplyToQ

Tabella 126. Nomi, descrizioni e tipi di proprietà (Continua)

Proprietà	Descrizione	Tipo Java	Link alla descrizione completa
Gestore code JMS_IBM_MQMD_ReplyTo	Nome del gestore code di risposta	Stringa	ReplyToQMgr
JMS_IBM_MQMD_UserIdentifier	Identificativo utente	Stringa	UserIdentifier
JMS_IBM_MQMD_AccountingToken	Token account	Oggetto (byte []) [4]	AccountingToken
Dati JMS_IBM_MQMD_ApplIdentity	Dati dell'applicazione correlati all'identità	Stringa	ApplIdentityData
Tipo JMS_IBM_MQMD_PutAppl	Tipo di applicazione che inserisce il messaggio	Intero	PutApplType
Nome JMS_IBM_MQMD_PutAppl	Nome dell'applicazione che inserisce il messaggio	Stringa	PutApplName
JMS_IBM_MQMD_PutDate	Data in cui il messaggio è stato inserito	Stringa	PutDate
PutTime JMS_IBM_MQMD_	Ora in cui il messaggio è stato inserito	Stringa	PutTime
JMS_IBM_MQMD_ApplOriginDati	Dati dell'applicazione correlati all'origine	Stringa	ApplOriginData
JMS_IBM_MQMD_GroupId	ID gruppo	Oggetto (byte []) [4]	GroupId
Numero JMS_IBM_MQMD_MsgSeq	Numero di sequenza del messaggio logico all'interno del gruppo	Intero	MsgSeqNumber
Offset MQMD_IBM_JMS	Offset di dati nel messaggio fisico dall'inizio del messaggio logico	Intero	Offset
JMS_IBM_MQMD_MsgFlags	Indicatori di messaggio	Intero	MsgFlags
JMS_IBM_MQMD_OriginalLength	Lunghezza del messaggio originale	Intero	OriginalLength

1. **Attenzione:** Se si assegna un valore a JMS_IBM_MQMD_Priority non compreso nell'intervallo 0-9, ciò viola la specifica JMS.
2. **Attenzione:** La specifica JMS indica che l'ID messaggio deve essere impostato dal provider JMS e che deve essere univoco o null. Se si assegna un valore a JMS_IBM_MQMD_MsgId, questo valore viene copiato in JMSMessageID. Pertanto, non è impostato dal fornitore JMS e potrebbe non essere univoco: ciò viola la specifica JMS.
3. **Attenzione:** Se si assegna un valore a JMS_IBM_MQMD_CorrelId che inizia con la stringa 'ID:', ciò viola la specifica JMS.
4. **Attenzione:** L'utilizzo delle proprietà della matrice di byte su un messaggio viola le specifiche JMS.

Accesso ai dati del messaggio IBM WebSphere MQ da una applicazione utilizzando le classi WebSphere MQ per JMS

È possibile accedere ai dati completi del messaggio WebSphere MQ all'interno di un'applicazione utilizzando le classi IBM WebSphere MQ per JMS. Per accedere a tutti i dati, il messaggio deve essere un JMSBytesMessage. Il corpo di JMSBytesMessage include qualsiasi intestazione MQRFH2 , qualsiasi altra intestazione IBM WebSphere MQ e i seguenti dati del messaggio.

Impostare la proprietà WMQ_MESSAGE_BODY della destinazione su WMQ_MESSAGE_BODY_MQper ricevere tutti i dati del corpo del messaggio in JMSBytesMessage.

Se WMQ_MESSAGE_BODY viene impostato su WMQ_MESSAGE_BODY_JMS o WMQ_MESSAGE_BODY_UNSPECIFIED, il corpo del messaggio viene restituito senza l'intestazione JMS MQRFH2 e le proprietà di JMSBytesMessage riflettono le proprietà impostate in RFH2.

Alcune applicazioni non possono utilizzare le funzioni descritte in questo argomento. Se un'applicazione è connessa a un gestore code WebSphere MQ V6 , o se ha impostato PROVIDERVERSION su 6, le funzioni non saranno disponibili.

invio di un messaggio

Quando si inviano messaggi, la propriet ... di destinazione, WMQ_MESSAGE_BODY, ha la precedenza su WMQ_TARGET_CLIENT.

Se WMQ_MESSAGE_BODY è impostato su WMQ_MESSAGE_BODY_JMS, WebSphere MQ classes for JMS genera automaticamente un'intestazione MQRFH2 in base alle impostazioni delle proprietà e dei campi di intestazione JMSMessage .

Se WMQ_MESSAGE_BODY è impostato su WMQ_MESSAGE_BODY_MQ, non viene aggiunta alcuna intestazione aggiuntiva al corpo del messaggio

Se WMQ_MESSAGE_BODY è impostato su WMQ_MESSAGE_BODY_UNSPECIFIED, le classi WebSphere MQ per JMS inviano un'intestazione MQRFH2 , a meno che WMQ_TARGET_CLIENT non sia impostato su WMQ_TARGET_DEST_MQ. In fase di ricezione, l'impostazione di WMQ_TARGET_CLIENT su WMQ_TARGET_DEST_MQ determina la rimozione di qualsiasi MQRFH2 dal corpo del messaggio.

Nota: JMSBytesMessage e JMSTextMessage non richiedono MQRFH2, mentre JMSStreamMessage, JMSMapMessagee JMSObjectMessage lo richiedono.

WMQ_MESSAGE_BODY_UNSPECIFIED è l'impostazione predefinita per WMQ_MESSAGE_BODYe WMQ_TARGET_DEST_JMS è quella predefinita per WMQ_TARGET_CLIENT.

Se si invia un JMSBytesMessage, è possibile sovrascrivere le impostazioni predefinite per il corpo del messaggio JMS quando viene creato il messaggio WebSphere MQ . Utilizzare le seguenti proprietà:

- JMS_IBM_Format o JMS_IBM_MQMD_Format: questa proprietà specifica il formato dell'intestazione WebSphere MQ o del payload dell'applicazione che avvia il corpo del messaggio JMS se non è presente alcuna intestazione Websphere MQ precedente.
- JMS_IBM_Character_Set o JMS_IBM_MQMD_CodedCharSetId: questa proprietà specifica il CCSID dell'intestazione WebSphere MQ o del payload dell'applicazione che avvia il corpo del messaggio JMS se non è presente un'intestazione MQ Websphere precedente.
- JMS_IBM_Encoding o JMS_IBM MQMD_Encoding: questa proprietà specifica la codifica dell'intestazione WebSphere MQ o del payload dell'applicazione che avvia il corpo del messaggio JMS se non è presente alcuna intestazione Websphere MQ precedente.

Se vengono specificati entrambi i tipi di proprietà, le proprietà JMS_IBM_MQMD_* sovrascrivono le proprietà JMS_IBM_* corrispondenti, purché la proprietà di destinazione WMQ_MQMD_WRITE_ENABLED sia impostata su true.

Le differenze effettive tra l'impostazione delle proprietà del messaggio utilizzando JMS_IBM_MQMD_* e JMS_IBM_* sono significative:

1. Le proprietà JMS_IBM_MQMD_* sono specifiche del provider JMS IBM WebSphere MQ .

2. Le proprietà JMS_IBM_MQMD_* sono impostate solo in MQMD. Le proprietà JMS_IBM_* sono impostate in MQMD solo se il messaggio non ha un'intestazione JMS MQRFH2 . Altrimenti, vengono impostati nell'intestazione JMS RFH2 .
3. Le proprietà JMS_IBM_MQMD_* non hanno alcun effetto sulla codifica di testo e numeri scritti in un JMSMessage.

   È probabile che un'applicazione ricevente assuma che i valori di MQMD.Encoding e MQMD.CodedCharSetId corrispondano alla codifica e alla serie di caratteri dei numeri e del testo nel corpo del messaggio. Se vengono utilizzate le proprietà JMS_IBM_MQMD_* , è responsabilità dell'applicazione di invio renderla tale. La codifica e la serie di caratteri di numeri e testo nel corpo del messaggio sono impostate dalle proprietà JMS_IBM_* .

   Il frammento codificato in modo errato in Figura 163 a pagina 904 invia un messaggio codificato nella serie di caratteri 1208, con MQMD.CodedCharSetId impostato su 37.

   a. Invia messaggio codificato in modo errato

```
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. La ricezione del messaggio, in base al valore di JMS_IBM_CHARACTER_SET impostato dal valore MQMD.CodedCharSetId:

```
TextMessage tmi = (TextMessage) cons.receive();
System.out.println("Message is \"" + tmi.getText() + "\"");
```

   c. Output risultante:

```
Message is "ëÈÊ'>...??>?"
```

Figura 163. Dati di messaggi e MQMD codificati in modo incongruente

Uno dei frammenti di codice in Figura 164 a pagina 904 risulta in un messaggio inserito in una coda o in un argomento, con il suo corpo che contiene il payload dell'applicazione senza che venga aggiunta un'intestazione MQRFH2 generata automaticamente.

1. Impostazione WMQ_MESSAGE_BODY_MQ:

```
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

2. Impostazione WMQ_TARGET_DEST_MQ:

```
((MQDestination) destination).setMessageBodyStyle
                 (WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);
((MQDestination) destination).
                setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
```

Figura 164. Inviare un messaggio con un corpo del messaggio MQ .

ricezione di un messaggio

Se WMQ_MESSAGE_BODY è impostato su WMQ_MESSAGE_BODY_JMS, il tipo e il corpo del messaggio JMS in ingresso sono determinati dal contenuto del messaggio di WebSphere MQ ricevuto. Il tipo e il corpo del messaggio sono determinati dai campi nell'intestazione MQRFH2 o in MQMD, se non è presente alcun MQRFH2.

Se WMQ_MESSAGE_BODY è impostato su WMQ_MESSAGE_BODY_MQ, il tipo di messaggio JMS in ingresso è JMSBytesMessage. Il corpo del messaggio JMS è costituito dai dati del messaggio restituiti dalla chiamata API MQGET sottostante. La lunghezza del corpo del messaggio è la lunghezza restituita dalla chiamata MQGET . La serie di caratteri e la codifica dei dati nel corpo del messaggio sono determinati dai campi CodedCharSetId e Codifica di MQMD. Il formato dei dati nel corpo del messaggio è determinato dal campo Formato di MQMD

Se WMQ_MESSAGE_BODY è impostato su WMQ_MESSAGE_BODY_UNSPECIFIED, il valore predefinito, IBM WebSphere MQ classes per JMS lo imposta su WMQ_MESSAGE_BODY_JMS.

Quando si riceve un JMSBytesMessage, è possibile decodificarlo facendo riferimento alle seguenti proprietà:

- JMS_IBM_Format o JMS_IBM_MQMD_Format: questa proprietà specifica il formato dell'intestazione WebSphere MQ o del payload dell'applicazione che avvia il corpo del messaggio JMS se non è presente alcuna intestazione Websphere MQ precedente.
- JMS_IBM_Character_Set o JMS_IBM_MQMD_CodedCharSetId: questa proprietà specifica il CCSID dell'intestazione WebSphere MQ o del payload dell'applicazione che avvia il corpo del messaggio JMS se non è presente un'intestazione MQ Websphere precedente.
- JMS_IBM_Encoding o JMS_IBM MQMD_Encoding: questa proprietà specifica la codifica dell'intestazione WebSphere MQ o del payload dell'applicazione che avvia il corpo del messaggio JMS se non è presente alcuna intestazione Websphere MQ precedente.

Il seguente frammento di codice risulta in un messaggio ricevuto che è un JMSBytesMessage. Indipendentemente dal contenuto del messaggio ricevuto e dal campo del formato del MQMDricevuto, il messaggio è un JMSBytesMessage.

```
((MQDestination)destination).setMessageBodyStyle
                            (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

Proprietà di destinazione WMQ_MESSAGE_BODY
WMQ_MESSAGE_BODY determina se un'applicazione JMS elabora l' MQRFH2 di un messaggio WebSphere MQ come parte del payload del messaggio (ovvero, come parte del corpo del messaggio JMS).

Tabella 127. Nomi e descrizioni delle proprietà

Proprietà	**Forma breve**	**Descrizione**
CORPO_MESSAGGIO_WMQ_	MBODY	Se un'applicazione JMS elabora l' MQRFH2 di un messaggio WebSphere MQ come parte del payload del messaggio (ovvero, come parte del corpo del messaggio JMS).

Tabella 128. Nomi proprietà, valori e metodi set			
Proprietà	**Valori validi nello strumento di gestione (valori predefiniti in grassetto)**	**Valori validi nei programmi**	**Imposta metodo**
WMQ_MESSAGE_CORPO	• **NON SPECIFICATO** Durante l'invio, WebSphere MQ classes for JMS procede o non procede a generare e includere un'intestazione MQRFH2 , a seconda del valore di WMQ_TARGET_CLIENT. Durante la ricezione, agisce come valore JMS. • JMS Durante l'invio, WebSphere MQ classes for JMS genera automaticamente un'intestazione MQRFH2 e la include nel messaggio WebSphere MQ . Quando si riceve, le classi WebSphere MQ per JMS impostano le proprietà del messaggio JMS in base ai valori in MQRFH2 (se presente); non presenta MQRFH2 come parte del corpo del messaggio JMS. • MQ Durante l'invio, WebSphere MQ classes for JMS non genera un' MQRFH2. Quando si riceve, WebSphere MQ classes for JMS presenta MQRFH2 come parte del contenuto del messaggio JMS.	• **WMQ_MESSAGE_ BODY_UNSPECIFIED** • WMQ_MESSAGE_BODY_JMS • MESSAGE_BODY_MQ WMQ_	setMessageBodySty le

Messaggi persistenti JMS

Le classi WebSphere MQ per le applicazioni JMS possono utilizzare l'attributo della coda **NonPersistentMessageClass** per fornire migliori prestazioni per i messaggi persistenti JMS, a discapito di una certa affidabilità.

Una coda WebSphere MQ ha un attributo denominato **NonPersistentMessageClass**. Il valore di questo attributo determina se i messaggi non persistenti sulla coda vengono eliminati al riavvio del gestore code.

È possibile impostare l'attributo per una coda locale utilizzando WebSphere MQ Script (MQSC), DEFINE QLOCAL, con uno dei seguenti parametri:

NPMCLASS (NORMALE)
I messaggi non persistenti sulla coda vengono eliminati al riavvio del gestore code. Questo è il valore predefinito.

NPMCLASS (ALTO)
I messaggi non persistenti sulla coda non vengono eliminati quando il gestore code viene riavviato in seguito a un arresto inattivo o immediato. I messaggi non persistenti potrebbero essere eliminati, tuttavia, a seguito di un arresto preventivo o di un errore.

Questo argomento descrive come le classi WebSphere MQ per le applicazioni JMS possono utilizzare questo attributo della coda per fornire prestazioni migliori per i messaggi persistenti JMS.

La proprietà PERSISTENCE di un oggetto Coda o Argomento può avere il valore HIGH. È possibile utilizzare lo strumento di amministrazione JMS WebSphere MQ per impostare questo valore oppure un'applicazione può richiamare il metodo Destination.setPersistence() inoltrando il valore WMQConstants.WMQ_PER_NPHIGH come parametro.

Se un'applicazione invia un messaggio persistente JMS o un messaggio non persistente JMS a una destinazione in cui la proprietà PERSISTENCE ha il valore HIGH, e la coda sottostante WebSphere MQ è impostata su NPMCLASS (HIGH), il messaggio viene inserito nella coda come un messaggio non persistente WebSphere MQ . Se la proprietà PERSISTENCE della destinazione non ha il valore HIGH, o se la coda sottostante è impostata su NPMCLASS (NORMAL), un messaggio persistente JMS viene inserito nella coda come un messaggio persistente WebSphere MQ e un messaggio non persistente JMS viene inserito nella coda come un messaggio non persistente WebSphere MQ .

Se un messaggio persistente JMS viene inserito in una coda come un messaggio non persistente WebSphere MQ e si desidera assicurarsi che il messaggio non venga eliminato in seguito a un arresto inattivo o immediato di un gestore code, tutte le code attraverso cui il messaggio potrebbe essere instradato devono essere impostate su NPMCLASS (HIGH). Nel dominio di pubblicazione / sottoscrizione, queste code includono le code del sottoscrittore. Come supporto per l'applicazione di questa configurazione, WebSphere MQ classes for JMS genera un'eccezione InvalidDestinationse un'applicazione tenta di creare un consumatore di messaggi per una destinazione in cui la proprietà PERSISTENCE ha il valore HIGH e la coda sottostante WebSphere MQ è impostata su NPMCLASS (NORMAL).

L'impostazione della proprietà PERSISTENCE di una destinazione su HIGH non influisce sulla modalità di ricezione di un messaggio da tale destinazione. Un messaggio inviato come messaggio permanente JMS viene ricevuto come messaggio permanente JMS e un messaggio inviato come messaggio non permanente JMS viene ricevuto come messaggio permanente JMS.

Quando un'applicazione invia il primo messaggio ad una destinazione in cui la proprietà PERSISTENCE ha il valore HIGH, o quando un'applicazione crea il primo consumatore di messaggi per una destinazione in cui la proprietà PERSISTENCE ha il valore HIGH, WebSphere MQ classes for JMS emette una chiamata MQINQ per determinare se NPMCLASS (HIGH) è impostato sulla coda WebSphere MQ sottostante. L'applicazione deve quindi disporre dell'autorità per richiedere informazioni sulla coda. Inoltre, WebSphere MQ classes per JMS preserva il risultato della chiamata MQINQ finché la destinazione non viene eliminata e non emette ulteriori chiamate MQINQ. Pertanto, se si modifica l'impostazione NPMCLASS nella coda sottostante mentre l'applicazione sta ancora utilizzando la destinazione, WebSphere MQ classes per JMS non nota la nuova impostazione.

Consentendo ai messaggi persistenti JMS di essere inseriti nelle code WebSphere MQ come messaggi non persistenti WebSphere MQ , si stanno ottenendo prestazioni a scapito di una certa affidabilità. Se si richiede la massima affidabilità per i messaggi persistenti JMS, non inviare i messaggi ad una destinazione in cui la proprietà PERSISTENCE ha il valore HIGH.

Il livello JMS può utilizzare SYSTEM.JMS.TEMPQ.MODEL, invece di SYSTEM.DEFAULT.MODEL.QUEUE. SYSTEM.JMS.TEMPQ.MODEL crea code dinamiche permanenti che accettano messaggi persistenti, perché SYSTEM.DEFAULT.MODEL.QUEUE non può accettare messaggi persistenti. Se si desidera utilizzare le code temporanee per accettare i messaggi persistenti, è necessario utilizzare SYSTEM.JMS.TEMPQ.MODELo modificare la coda modello in una coda alternativa di propria scelta.

Utilizzo di SSL (Secure Sockets Layer) con le classi WebSphere MQ per JMS

WebSphere Le classi di MQ per le applicazioni JMS possono utilizzare la codifica SSL. Per fare ciò, è necessario un fornitore JSSE.

WebSphere MQ classes for JMS connections using TRANSPORT (CLIENT) supportano la codifica SSL (Secure Sockets Layer). SSL fornisce la codifica di comunicazione, l'autenticazione e l'integrità del messaggio. Generalmente viene utilizzato per proteggere le comunicazioni tra due peer su Internet o all'interno di una intranet.

WebSphere Le classi MQ per JMS utilizzano JSSE (Java Secure Socket Extension) per gestire la codifica SSL e pertanto richiedono un fornitore JSSE. Le JVM JSE v1.4 hanno un provider JSSE integrato. I dettagli su come gestire e memorizzare i certificati possono variare da fornitore a fornitore. Per informazioni, consultare la documentazione del provider JSSE.

Questa sezione presuppone che il provider JSSE sia installato e configurato correttamente e che i certificati appropriati siano stati installati e resi disponibili per il proprio provider JSSE. È ora possibile utilizzare JMSAdmin per impostare un certo numero di proprietà amministrative.

Se l'applicazione delle classi WebSphere MQ per JMS utilizza una CCDT (client channel definition table) per connettersi a un gestore code, consultare "Utilizzo di una tabella di definizione di canale client con IBM WebSphere MQ classes for JMS" a pagina 917.

Proprietà oggetto SSLCIPHERSUITE
Impostare SSLCIPHERSUITE per abilitare la codifica SSL su un oggetto ConnectionFactory .

Per abilitare la codifica SSL su un oggetto ConnectionFactory , utilizzare JMSAdmin per impostare la proprietà SSLCIPHERSUITE su una CipherSuite supportata dal provider JSSE. Deve corrispondere all'impostazione CipherSpec sul canale di destinazione. Tuttavia, CipherSuites sono distinte da CipherSpecs e, pertanto, hanno nomi differenti. "SSL CipherSpecs e CipherSuites in JMS" a pagina 911 contiene una tabella che associa i CipherSpecs supportati da WebSphere MQ ai relativi CipherSuites equivalenti noti a JSSE. Per ulteriori informazioni su CipherSpecs e CipherSuites con WebSphere MQ, consultare Sicurezza.

Ad esempio, per configurare un oggetto ConnectionFactory che può essere utilizzato per creare una connessione su un canale MQI abilitato SSL con CipherSpec di RC4_MD5_EXPORT, immettere il seguente comando per JMSAdmin:

```
ALTER CF(my.cf) SSLCIPHERSUITE(SSL_RSA_EXPORT_WITH_RC4_40_MD5)
```

Questo può essere impostato anche da un'applicazione, utilizzando il metodo setSSLCipherSuite () su un oggetto MQConnectionFactory .

Per comodità, se viene specificato un CipherSpec nella proprietà SSLCIPHERSUITE, JMSAdmin tenta di associare CipherSpec a una CipherSuite appropriata e invia un'avvertenza. Questo tentativo di associazione non viene effettuato se la proprietà è specificata da un'applicazione.

In alternativa, utilizzare CCDT (Client Channel Definition Table). Per ulteriori informazioni, vedere "Utilizzo di una tabella di definizione di canale client con IBM WebSphere MQ classes for JMS" a pagina 917.

proprietà oggetto SSLFIPSREQUIRED
Se si richiede una connessione per utilizzare una CipherSuite supportata dal provider IBM Java JSSE FIPS (IBM JSSEFIPS), impostare la proprietà SSLFIPSREQUIRED del factory di connessione su YES.

Il valore predefinito di questa proprietà è NO, che significa che una connessione può utilizzare qualsiasi CipherSuite supportato da WebSphere MQ.

Se un'applicazione utilizza più di una connessione, il valore di SSLFIPSREQUIRED utilizzato quando l'applicazione crea la prima connessione determina il valore utilizzato quando l'applicazione crea una connessione successiva. Ciò significa che il valore della proprietà SSLFIPSREQUIRED della factory di connessione utilizzata per creare una connessione successiva viene ignorato. È necessario riavviare l'applicazione se si desidera utilizzare un valore diverso di SSLFIPSREQUIRED.

Un'applicazione può impostare questa proprietà richiamando il metodo setSSLFipsRequired () di un oggetto ConnectionFactory . La proprietà viene ignorata se non è impostata alcuna CipherSuite .

Attività correlate
Specifica che solo i CipherSpecs certificati FIPS vengono utilizzati al runtime sul client MQI

Riferimenti correlati

Federal Information Processing Standards (FIPS) per UNIX, Linux e Windows

proprietà oggetto SSLPEERNAME

Utilizzare SSLPEERNAME per specificare un modello di DN (distinguished name), per assicurarsi che l'applicazione JMS si connetta al gestore code corretto.

Un'applicazione JMS può garantire la connessione al gestore code corretto specificando un modello DN (distinguished name). La connessione ha esito positivo solo se il gestore code presenta un DN che corrisponde al pattern. Per ulteriori dettagli sul formato di questo pattern, consultare gli argomenti correlati.

Il DN viene impostato utilizzando la proprietà SSLPEERNAME di un oggetto ConnectionFactory . Ad esempio, il seguente comando JMSAdmin imposta un oggetto ConnectionFactory in modo che il gestore code si identifichi con un nome comune che inizia con i caratteri `QMGR.` e con almeno due nomi di unità organizzative, il primo dei quali deve essere IBM e il secondo WEBSPHERE:

```
ALTER CF(my.cf) SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Il controllo non è sensibile al maiuscolo / minuscolo e i punti e virgola possono essere utilizzati al posto delle virgole. SSLPEERNAME può essere impostato anche da un'applicazione utilizzando il metodo setSSLPeerName () su un oggetto MQConnectionFactory . Se questa proprietà non è impostata, non viene eseguito alcun controllo sul DN (Distinguished Name) fornito dal gestore code. Questa proprietà viene ignorata se non è impostata alcuna CipherSuite .

Proprietà oggetto SSLCERTSTORES

Utilizzare SSLCERTSTORES per specificare un elenco di server LDAP da utilizzare per il controllo CRL (Certificate Revocation List).

È comune utilizzare un CRL (Certificate Revocation List) per identificare i certificati non più attendibili. I CRL sono generalmente ospitati su server LDAP. JMS consente a un server LDAP di essere specificato per il controllo CRL in Java 2 v1.4 o versioni successive. Il seguente esempio di JMSAdmin indica a JMS di utilizzare un CRL ospitato su un server LDAP denominato crl1.ibm.com:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com)
```

Nota: Per utilizzare correttamente un CertStore con un CRL ospitato su un server LDAP, assicurarsi che l'SDK (Software Development Kit) Java sia compatibile con il CRL. Alcuni SDK richiedono che il CRL sia conforme a RFC 2587, che definisce uno schema per LDAP v2. La maggior parte dei server LDAP v3 utilizza invece RFC 2256.

Se il server LDAP non è in esecuzione sulla porta predefinita 389, è possibile specificare la porta accodando i due punti (:) e il numero di porta al nome host. Se il certificato presentato dal gestore code è presente nel CRL ospitato su crl1.ibm.com, la connessione non viene completata. Per evitare un singolo punto di errore, JMS consente di fornire più server LDAP fornendo un elenco di server LDAP delimitati dal carattere spazio. Di seguito è riportato un esempio:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com ldap://crl2.ibm.com)
```

Quando vengono specificati più server LDAP, JMS tenta ciascuno di essi finché non trova un server con cui è possibile verificare correttamente il certificato del gestore code. Ciascun server deve contenere informazioni identiche.

Una stringa in questo formato può essere fornita da un'applicazione sul metodo MQConnectionFactory.setSSLCertStores (). In alternativa, l'applicazione può creare uno o più oggetti java.security.cert.CertStore , posizionarli in un oggetto Collection adatto e fornire questo oggetto Collection al metodo setSSLCertStores (). In questo modo, l'applicazione può personalizzare la verifica CRL. Consultare la documentazione JSSE per i dettagli sulla creazione e l'utilizzo degli oggetti CertStore .

Il certificato presentato dal gestore code durante l'impostazione di una connessione viene convalidato nel modo seguente:

1. Il primo oggetto CertStore nella raccolta identificata da sslCertStores viene utilizzato per identificare un server CRL.
2. È stato effettuato un tentativo di contattare il server CRL.
3. Se il tentativo ha esito positivo, il server viene ricercato per una corrispondenza per il certificato.
   a. Se viene rilevato che il certificato è stato revocato, il processo di ricerca è stato completato e la richiesta di connessione ha esito negativo con codice motivo MQRC_SSL_CERTIFICATE_REVOKED.
   b. Se il certificato non viene trovato, il processo di ricerca è stato completato e la connessione può continuare.
4. Se il tentativo di contattare il server non ha esito positivo, il successivo oggetto CertStore viene utilizzato per identificare un server CRL e il processo si ripete dal passo 2.

   Se questo era l'ultimo CertStore nella raccolta o se la raccolta non contiene oggetti CertStore , il processo di ricerca ha avuto esito negativo e la richiesta di connessione ha avuto esito negativo con codice motivo MQRC_SSL_CERT_STORE_ERROR.

L'oggetto Raccolta determina l'ordine in cui vengono utilizzati i CertStores .

Se l'applicazione utilizza setSSLCertStores () per impostare una raccolta di oggetti CertStore , non è più possibile eseguire il bind di MQConnectionFactory in uno spazio dei nomi JNDI. Il tentativo di eseguire questa operazione causa un'eccezione. Se la proprietà sslCertStores non è impostata, non viene eseguito alcun controllo di revoca sul certificato fornito dal gestore code. Questa proprietà viene ignorata se non è impostata alcuna CipherSuite .

proprietà oggetto SSLRESETCOUNT
Questa proprietà rappresenta il numero totale di byte inviati e ricevuti da una connessione prima che la chiave segreta utilizzata per la codifica venga rinegoziata.

Il numero di byte inviati è il numero prima della codifica e il numero di byte ricevuti è il numero dopo la decodifica. Il numero di byte include anche le informazioni di controllo inviate e ricevute dalle classi WebSphere MQ per JMS.

Ad esempio, per configurare un oggetto ConnectionFactory che può essere utilizzato per creare una connessione su un canale MQI abilitato SSL con una chiave segreta rinegoziata dopo il flusso di 4 MB di dati, immettere il seguente comando in JMSAdmin:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Un'applicazione può impostare questa proprietà richiamando il metodo setSSLResetCount () di un oggetto ConnectionFactory .

Se il valore di questa proprietà è zero, che è il valore predefinito, la chiave segreta non viene mai rinegoziata. La proprietà viene ignorata se non è impostata alcuna CipherSuite .

Proprietà oggetto SSLSocketFactory
Per personalizzare altri aspetti della connessione SSL per un'applicazione, creare un SSLSocketFactory e configurare JMS per utilizzarlo.

È possibile personalizzare altri aspetti della connessione SSL per un'applicazione. Ad esempio, è possibile che si desideri inizializzare l'hardware crittografico o modificare il keystore e il truststore in uso. A tale scopo, l'applicazione deve prima creare un oggetto javax.net.ssl.SSLSocketFactory personalizzato di conseguenza. Consultare la documentazione JSSE per informazioni su come eseguire questa operazione, poiché le funzioni personalizzabili variano da provider a provider. Una volta ottenuto un oggetto SSLSocketFactory adatto, utilizzare il metodo MQConnectionFactory.setSSLSocketFactory () per configurare JMS per utilizzare l'oggetto personalizzato SSLSocketFactory .

Se la propria applicazione utilizza il metodo setSSLSocketFactory () per impostare un oggetto SSLSocketFactory personalizzato, l'oggetto MQConnectionFactory non può più essere collegato in uno

spazio dei nomi JNDI. Il tentativo di eseguire questa operazione causa un'eccezione. Se questa proprietà non è impostata, viene utilizzato l'oggetto predefinito SSLSocketFactory . Consultare la propria documentazione JSSE per i dettagli sul comportamento dell'oggetto SSLSocketFactory predefinito. Questa proprietà viene ignorata se non è impostata alcuna CipherSuite .

Importante: Non presupporre che l'utilizzo delle proprietà SSL assicuri la sicurezza quando un oggetto ConnectionFactory viene richiamato da uno spazio nomi JNDI che non è esso stesso sicuro. In particolare, l'implementazione LDAP standard di JNDI non è sicura. Un aggressore può imitare il server LDAP, inducendo un'applicazione JMS a connettersi al server sbagliato senza accorgersene. Con adeguate disposizioni di sicurezza in atto, altre implementazioni di JNDI (come l'implementazione fscontext) sono sicure.

Esecuzione di modifiche al keystore o al truststore JSSE
Se si apportano modifiche al keystore o al truststore, è necessario eseguire determinate azioni per rendere effettive le modifiche.

Se si modifica il contenuto del keystore o del truststore JSSE o si modifica l'ubicazione del keystore o del file truststore, le classi WebSphere MQ per le applicazioni JMS in esecuzione al momento non acquisiscono automaticamente le modifiche. Per rendere effettive le modifiche, è necessario eseguire le seguenti operazioni:

- Le applicazioni devono chiudere tutte le relative connessioni ed eliminare tutte le connessioni inutilizzate nei pool di connessioni.
- Se il provider JSSE memorizza nella cache le informazioni dal keystore e dal truststore, tali informazioni devono essere aggiornate.

Una volta eseguite queste azioni, le applicazioni possono ricreare le connessioni.

A seconda della modalità di progettazione delle applicazioni e della funzione fornita dal provider JSSE, potrebbe essere possibile eseguire queste azioni senza arrestare e riavviare le applicazioni. Tuttavia, l'arresto e il riavvio delle applicazioni potrebbe essere la soluzione più semplice.

SSL CipherSpecs e CipherSuites in JMS
CipherSpecs supportati da WebSphere MQ e i relativi CipherSuitesequivalenti.

Tabella 129 a pagina 911 elenca i CipherSpecs supportati da WebSphere MQ e i relativi CipherSuitesequivalenti. Se la proprietà ConnectionFactory SSLFIPSREQUIRED è impostata su NO, un'applicazione WebSphere MQ classes per JMS può connettersi a un gestore code se viene specificato un CipherSpec supportato all'estremità server del canale MQI e l'equivalente CipherSuite viene specificato all'estremità client. Se SSLFIPSREQUIRED è impostato su YES, la combinazione di CipherSpec e di CipherSuite determina se l'applicazione può connettersi al gestore code.

All'estremità del server di un canale MQI, il nome di un CipherSpec può essere specificato come valore del parametro SSLCIPH in un comando DEFINE CHANNEL CHLTYPE (SVRCONN). All'estremità client di un canale MQI, il nome di CipherSuite può essere specificato nei modi seguenti:

- Un'applicazione può richiamare il metodo setSSLCipherSuite () di un oggetto ConnectionFactory .
- Utilizzando lo strumento di gestione JMS WebSphere MQ , è possibile impostare la proprietà SSLCIPHERSUITE di un oggetto ConnectionFactory .

Tabella 129. CipherSpecs supportati da WebSphere MQ e relativi CipherSuites equivalenti

CipherSpec	**CipherSuite equivalente**	**Connessione possibile se SFIPS [1] è impostato su YES?**
NULL_MD5	SSL_RSA_WITH_NULL_MD5	No

Tabella 129. CipherSpecs supportati da WebSphere MQ e relativi CipherSuites equivalenti (Continua)		
CipherSpec	**CipherSuite equivalente**	**Connessione possibile se SFIPS [1] è impostato su YES?**
NULL_SHA	SSL_RSA_WITH_NULL_SHA	No
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5	No
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	No
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA	No
RC2_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	No
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA	No
RC4_56_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA	No
DES_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA	No
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA	No
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	Nessun [7]
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	Sì [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	Sì [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	Sì [5] [7]
AES_SHA_IT[2]		
TLS_RSA_WITH_DES_CBC_SHA[8] [9]	SSL_RSA_WITH_DES_CBC_SHA	No[3]
TLS_RSA_WITH_3DES_EDE_CBC_SHA[8]	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Sì
FIPS_WITH_DES_CBC_SHA	SSL_RSA_FIPS_WITH_DES_CBC_SHA	Nessun[4]
FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA	Nessun[6]

Note:

1. Quando si utilizza lo strumento di gestione JMS WebSphere MQ , SFIPS è il nome breve della proprietà ConnectionFactory SSLFIPSREQUIRED.
2. Questo CipherSpec non ha un CipherSuiteequivalente.
3. Questa CipherSpec era certificata FIPS 140-2 prima del 19th maggio 2007.
4. Questa CipherSpec era certificata FIPS 140-2 prima del 19th maggio 2007. Il nome FIPS_WITH_DES_CBC_SHA è storico e riflette il fatto che questo CipherSpec era precedentemente (ma non più) conforme a FIPS. Questa CipherSpec è obsoleta e non se ne consiglia l'utilizzo.
5. Questi CipherSpecs (TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) non possono essere utilizzati per proteggere una connessione da WebSphere MQ Explorer a un gestore code a meno che non vengano applicati i file di politica non limitati appropriati al JRE utilizzato da Explorer.

   Consultare Informazioni sulla sicurezza per ulteriori informazioni sui file delle politiche.
6. Il nome FIPS_WITH_3DES_EDE_CBC_SHA è storico e riflette il fatto che questo CipherSpec era precedentemente (ma non è più) compatibile con FIPS. Questa CipherSpec è obsoleta e non se ne consiglia l'utilizzo.

7. Questi 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) richiedono IBM JREs 6.0 SR13 FP2 , 7.0 SR4 FP2 o versioni successive.
8. Questi CipherSpecs (TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_DES_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA256) possono utilizzare SSLv3 o TLS. Per impostazione predefinita, quando FIPS non è abilitato, si utilizza SSLv3 . Per utilizzare TLS, impostare la proprietà di sistema Java **com.ibm.mq.cfg.preferTLS** su `true`.
9. Questa CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA è obsoleta. Tuttavia, può essere ancora utilizzato per trasferire fino a 32 GB di dati prima che la connessione venga terminata con l'errore AMQ9288. Per evitare questo errore, è necessario evitare di utilizzare il triplo DES o abilitare la reimpostazione della chiave segreta quando si utilizza questo CipherSpec.

Informazioni correlate

Specifica che solo i CipherSpecs certificati FIPS vengono utilizzati al runtime sul client MQI

Federal Information Processing Standards (FIPS) per UNIX, Linux e Windows

Scrittura delle uscite del canale in Java per WebSphere MQ classes per JMS

Le uscite dei canali vengono create definendo classi Java che implementano le interfacce specificate.

Tre interfacce sono definite nel package com.ibm.mq.exits :

- WMQSendExit, per un'uscita di invio
- WMQReceiveExit, per un'uscita di ricezione
- WMQSecurityExit, per un'uscita di sicurezza

Il seguente codice di esempio definisce una classe che implementa tutte e tre le interfaccia:

```
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
  }
}
```

Ogni uscita riceve come parametri un oggetto MQCXP e un oggetto MQCD. Questi oggetti rappresentano le strutture MQCXP e MQCD definite nell'interfaccia procedurale.

Quando si chiama un'uscita di invio, il parametro agentBuffer contiene i dati che verranno inviati al gestore code del server. Un parametro di lunghezza non è richiesto perché l'espressione agentBuffer.limit () fornisce la lunghezza dei dati. L'uscita di invio restituisce come valore i dati da inviare al gestore code del server. Tuttavia, se l'uscita di invio non è l'ultima uscita di invio in una sequenza di uscite di invio, i

dati restituiti vengono passati invece alla successiva uscita di invio nella sequenza. Un'uscita di invio può restituire una versione modificata dei dati che riceve nel parametro agentBuffer oppure può restituire i dati invariati. Il corpo di uscita più semplice possibile è quindi:

```
{ return agentBuffer; }
```

Quando viene richiamata un'uscita di ricezione, il parametro agentBuffer contiene i dati ricevuti dal gestore code del server. L'uscita di ricezione restituisce come valore i dati da passare all'applicazione dalle classi WebSphere MQ per JMS. Tuttavia, se l'uscita di ricezione non è l'ultima uscita di ricezione in una sequenza di uscite di ricezione, i dati restituiti vengono trasmessi invece alla successiva uscita di ricezione nella sequenza.

Quando viene richiamata un'uscita di sicurezza, il parametro agentBuffer contiene i dati ricevuti in un flusso di sicurezza dall'uscita di sicurezza alla fine della connessione del server. L'uscita di sicurezza restituisce come valore i dati da inviare in un flusso di sicurezza all'uscita di sicurezza del server.

Le uscite del canale vengono richiamate con un buffer che ha un array di supporto. Per prestazioni ottimali, l'uscita deve restituire un buffer con un array di backup.

È possibile passare fino a 32 caratteri di dati utente a un'uscita canale quando viene richiamata. L'uscita accede ai dati utente richiamando il metodo getExitData () dell'oggetto MQCXP. Sebbene l'uscita possa modificare i dati utente richiamando il metodo setExitData (), i dati utente vengono aggiornati ogni volta che viene richiamata l'uscita. Tutte le modifiche apportate ai dati dell'utente vengono quindi perse. Tuttavia, l'uscita può trasmettere dati da una chiamata alla successiva utilizzando l'area utente di uscita dell'oggetto MQCXP. L'uscita accede all'area utente di uscita per riferimento richiamando il metodo getExitUserArea().

Ogni classe di uscita deve avere un costruttore. Il costruttore può essere il costruttore predefinito, come mostrato nell'esempio precedente, oppure un costruttore con un parametro stringa. Il costruttore viene richiamato per creare un'istanza della classe di uscita per ogni uscita definita nella classe. Pertanto, nell'esempio precedente, viene creata un'istanza della classe MyMQExits per l'uscita di invio, un'altra istanza per l'uscita di ricezione e una terza istanza per l'uscita di sicurezza. Quando viene richiamato un costruttore con un parametro stringa, il parametro contiene gli stessi dati utente passati all'uscita canale per cui viene creata l'istanza. Se una classe di uscita ha sia un costruttore predefinito che un singolo costruttore di parametri, il singolo costruttore di parametri ha la precedenza.

Non chiudere la connessione dall'interno di un'uscita canale.

Quando i dati vengono inviati all'estremità server di una connessione, la crittografia SSL viene eseguita *dopo* che vengono richiamate tutte le uscite del canale. Allo stesso modo, quando i dati vengono ricevuti dall'estremità del server di una connessione, la decodifica SSL viene eseguita *prima* che vengano richiamate le uscite del canale.

Nelle versioni di WebSphere MQ classes per JMS precedente alla versione 7.0, le uscite del canale sono state implementate utilizzando le interfacce MQSendExit, MQReceiveExite MQSecurityExit. È ancora possibile utilizzare queste interfacce, ma le nuove interfacce sono preferite per migliorare le funzioni e le prestazioni.

Configurazione di IBM WebSphere MQ classes for JMS per l'utilizzo delle uscite canale

Un'applicazione IBM WebSphere MQ classes for JMS può utilizzare la sicurezza del canale, inviare e ricevere uscite sul canale MQI che viene avviato quando l'applicazione si connette a un gestore code. L'applicazione può utilizzare le uscite scritte in Java, C o C + +. L'applicazione può anche utilizzare una sequenza di uscite di invio o ricezione eseguite in successione.

Le seguenti proprietà vengono utilizzate per specificare un'uscita di invio o una sequenza di uscite di invio, utilizzata da una connessione JMS :

- La proprietà **SENDEXIT** di un oggetto MQConnectionFactory .
- La proprietà **sendexit** su una specifica di attivazione utilizzata dall'adattatore di risorse IBM WebSphere MQ per la comunicazione in entrata,
- La proprietà **sendexit** in un oggetto ConnectionFactory utilizzato dall'adattatore di risorse IBM WebSphere MQ per la comunicazione di output.

Il valore della proprietà è una stringa che comprende uno o più elementi separati da virgole. Ogni voce identifica un'uscita di invio in uno dei seguenti modi:

- Il nome di una classe che implementa l'interfaccia `WMQSendExit` per un'uscita di invio scritta in Java.
- Una stringa nel formato *libraryName* (*entryPointName*) per un'uscita di invio scritta in C o C + +.

In modo simile, le seguenti proprietà specificano l'uscita di ricezione o la sequenza di uscite di ricezione utilizzata da una connessione:

- La proprietà **RECEXIT** di un oggetto MQConnectionFactory .
- La proprietà **receiveexit** su una specifica di attivazione utilizzata dall'adattatore di risorse IBM WebSphere MQ per la comunicazione in entrata,
- La proprietà **receiveexit** in un oggetto ConnectionFactory utilizzato dall'adattatore di risorse IBM WebSphere MQ per la comunicazione di output.

Le seguenti proprietà specificano l'uscita di sicurezza utilizzata da una connessione:

- La proprietà **SECEXIT** di un oggetto MQConnectionFactory .
- La proprietà **securityexit** su una specifica di attivazione utilizzata dall'adattatore di risorse IBM WebSphere MQ per la comunicazione in entrata,
- La proprietà **securityexit** in un oggetto ConnectionFactory utilizzato dall'adattatore di risorse IBM WebSphere MQ per la comunicazione di output.

Per MQConnectionFactories, è possibile impostare le proprietà **SENDEXIT**, **RECEXIT** e **SECEXIT** utilizzando lo strumento di amministrazione IBM WebSphere MQ JMS o IBM WebSphere MQ Explorer. In alternativa, un'applicazione può impostare le proprietà richiamando i metodi `setSendExit()`, `setReceiveExit()`e `setSecurityExit()` .

Le uscite canale vengono caricate dal proprio programma di caricamento classi. Per trovare un'uscita canale, il programma di caricamento classi ricerca le seguenti ubicazioni nell'ordine specificato.

1. Il percorso classe specificato dalla proprietà **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** o dall'attributo **JavaExitsClassPath** nella sezione Canali del file di configurazione del client IBM WebSphere MQ .
2. Il percorso classe specificato dalla proprietà di sistema Java **com.ibm.mq.exitClasspath**. Notare che questa proprietà è ora obsoleta.
3. La directory delle uscite IBM WebSphere MQ , come mostrato in Tabella 130 a pagina 915. Il programma di caricamento classi ricerca nella directory i file di classe che non sono compressi nei file JAR (Java archive). Se l'uscita del canale non viene trovata, il programma di caricamento classe ricerca i file JAR nella directory.

Tabella 130. La directory delle uscite IBM WebSphere MQ

Piattaforma	**Cartella**
UNIX and Linux	`/var/mqm/exits` (uscite canale a 32 bit) `/var/mqm/exits64` (uscite canale a 64 bit)
Windows	*dir_dati_installazione*\exits dove *install_data_dir* è la directory scelta per i file di dati IBM WebSphere MQ durante l'installazione. La directory predefinita è `C:\Program Files\IBM\WebSphere MQ`.

Nota: Se un'uscita del canale esiste in più di una ubicazione, IBM WebSphere MQ classes for JMS carica la prima istanza che trova.

Il parent del programma di caricamento classi è il programma di caricamento classi utilizzato per caricare IBM WebSphere MQ classes for JMS. È quindi possibile per il programma di caricamento classi parent caricare un'exit del canale se non è possibile trovarlo in una delle ubicazioni precedenti. Tuttavia, quando si sta utilizzando IBM WebSphere MQ classes for JMS in un ambiente come un server delle

applicazioni JEE , non è possibile influenzare la scelta del programma di caricamento classi parent e, quindi, il programma di caricamento classi deve essere configurato impostando la proprietà di sistema Java **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** sul server delle applicazioni.

Se l'applicazione è in esecuzione con Java Security Manager abilitato, il file di configurazione della politica utilizzato dall'ambiente di runtime Java in cui l'applicazione è in esecuzione deve disporre delle autorizzazioni per caricare una classe di uscita del canale. Per informazioni su come eseguire questa operazione, fare riferimento a Esecuzione delle classi IBM MQ per applicazioni JMS in Java Security Manager.

Le interfacce `MQSendExit`, `MQReceiveExite` `MQSecurityExit` fornite con versioni di IBM WebSphere MQ precedenti a Version 7.0 sono ancora supportate. Se si utilizzano uscite di canale che implementano queste interfacce, `com.ibm.mq.jar` deve essere presente nel percorso di classe.

Per informazioni su come scrivere uscite di canale in C, consultare “Programmi di uscita canale per canali di messaggistica” a pagina 399. È necessario memorizzare i programmi di uscita del canale scritti in C o C++ nella directory mostrata in Tabella 130 a pagina 915.

Se l'applicazione utilizza una tabella di definizione del canale client (CCDT) per connettersi a un gestore code, consultare “Utilizzo di una tabella di definizione di canale client con IBM WebSphere MQ classes for JMS” a pagina 917.

Specifica dei dati utente da trasmettere alle uscite canale quando si utilizzano le classi WebSphere MQ per JMS

È possibile passare fino a 32 caratteri di dati utente a un'uscita canale quando viene richiamata.

La proprietà SENDEXITINIT di un oggetto MQConnectionFactory specifica i dati utente passati a ciascuna uscita di invio quando viene richiamato. Il valore della proprietà è una stringa che comprende uno o più elementi di dati utente separati da virgole. La posizione di ogni elemento di dati utente all'interno della stringa determina a quale uscita di invio, in una sequenza di uscite di invio, vengono passati i dati utente. Ad esempio, il primo elemento dei dati utente nella stringa viene passato alla prima uscita di invio in una sequenza di uscite di invio.

È possibile impostare la proprietà SENDEXITINIT utilizzando lo strumento di amministrazione JMS di WebSphere MQ o WebSphere MQ Explorer. In alternativa, un'applicazione può impostare la proprietà richiamando il metodo setSendExitInit().

In modo simile, la proprietà RECEXITINIT di un oggetto ConnectionFactory specifica i dati utente passati a ciascuna uscita di ricezione e la proprietà SECEXITINIT specifica i dati dell'utente passati a una uscita di sicurezza. È possibile impostare queste proprietà utilizzando WebSphere MQ JMS administration tool o WebSphere MQ Explorer. In alternativa, un'applicazione può impostare proprietà richiamando i metodi setReceiveExitInit() e setSecurityExitInit().

Notare le seguenti regole quando si specificano i dati utente passati alle uscite del canale:

- Se il numero di elementi dei dati utente in una stringa è superiore al numero di uscite in una sequenza, gli elementi in eccesso dei dati utente vengono ignorati.
- Se il numero di elementi dei dati utente in una stringa è inferiore al numero di uscite in una sequenza, ogni elemento non specificato dei dati utente viene impostato su una stringa vuota. Due virgole in successione all'interno di una stringa, o una virgola all'inizio di una stringa, indicano anche un elemento non specificato dei dati utente.

Se un'applicazione utilizza una tabella di definizione del canale client (CCDT) per connettersi a un gestore code, i dati utente specificati in una definizione del canale di connessione client vengono trasmessi alle uscite del canale quando vengono richiamate. Per ulteriori informazioni sull'utilizzo di una tabella di definizione del canale client, consultare “Utilizzo di una tabella di definizione di canale client con IBM WebSphere MQ classes for JMS” a pagina 917.

Utilizzo di una tabella di definizione di canale client con IBM WebSphere MQ classes for JMS

Un'applicazione IBM WebSphere MQ classes per JMS può utilizzare le definizioni di canale di connessione client memorizzate in una CCDT (client channel definition table). Configurare un oggetto ConnectionFactory per utilizzare CCDT. Ci sono alcune limitazioni sul suo utilizzo.

Come alternativa alla creazione di una definizione di canale di connessione client impostando determinate proprietà di un oggetto ConnectionFactory , un'applicazione IBM WebSphere MQ classes for JMS può utilizzare le definizioni di canale di connessione client memorizzate in una tabella di definizioni di canale client. Queste definizioni vengono create dai comandi IBM WebSphere MQ Script (MQSC) o IBM WebSphere MQ Programmable Command Format (PCF). Quando l'applicazione crea un oggetto di connessione, IBM WebSphere MQ classes for JMS ricerca nella tabella di definizione di canale client una definizione di canale di connessione client appropriata e utilizza tale definizione per avviare un canale MQI. Per ulteriori informazioni sulle tabelle di definizione dei canali client e su come crearne una, consultare Tabella di definizione dei canali client.

Per utilizzare una tabella di definizione di canale client, la proprietà CCDTURL di un oggetto ConnectionFactory deve essere impostata su un oggetto URL. L'oggetto URL incapsula un URL (uniform resource locator) che identifica il nome e l'ubicazione del file contenente la tabella di definizione del canale client e specifica il modo in cui è possibile accedere al file. È possibile impostare la proprietà CCDTURL utilizzando lo strumento di amministrazione JMS IBM WebSphere MQ oppure un'applicazione può impostare la proprietà creando un oggetto URL e richiamando il metodo setCCDTURL() dell'oggetto ConnectionFactory .

Ad esempio, se il file ccdt1.tab contiene una tabella di definizione del canale client ed è memorizzato nello stesso sistema su cui è in esecuzione l'applicazione, l'applicazione può impostare la proprietà CCDTURL nel modo seguente:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
factory.setCCDTURL(chanTab1);
```

Come un altro esempio, si supponga che il file ccdt2.tab contenga una tabella di definizione di canale client e che sia memorizzato su un sistema diverso da quello su cui è in esecuzione l'applicazione. Se è possibile accedere al file utilizzando il protocollo FTP, l'applicazione può impostare la proprietà CCDTURL nel modo seguente:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
factory.setCCDTURL(chanTab2);
```

Oltre a impostare la proprietà CCDTURL dell'oggetto ConnectionFactory , la proprietà QMANAGER dello stesso oggetto deve essere impostata su uno dei seguenti valori:

- Il nome di un gestore code
- Un asterisco (*) seguito dal nome di un gruppo di gestori code
- Un asterisco (*)
- Una stringa vuota o una stringa contenente tutti i caratteri vuoti

Si tratta degli stessi valori che possono essere utilizzati per il parametro *QMgrName* in una chiamata MQCONN emessa da un'applicazione client che utilizza MQI (Message Queue Interface). Per ulteriori informazioni sul significato di questi valori, consultare MQCONN. È possibile impostare la proprietà QMANAGER utilizzando lo strumento di amministrazione JMS WebSphere MQ o IBM WebSphere MQ Explorer. In alternativa, un'applicazione può impostare la proprietà richiamando il metodo setQueueManager () dell'oggetto ConnectionFactory .

Se un'applicazione, quindi, crea un oggetto Connection dall'oggetto ConnectionFactory , IBM WebSphere MQ classes for JMS accede alla tabella di definizione del canale del client identificata dalla proprietà CCDTURL, utilizza la proprietà QMANAGER per ricercare nella tabella una definizione di canale di connessione client adatta e utilizza quindi la definizione di canale per avviare un canale MQI su un gestore code.

Notare che le proprietà CCDTURL e CHANNEL di un oggetto ConnectionFactory non possono essere entrambe impostate quando l'applicazione richiama il metodo createConnection(). Se entrambe le proprietà sono impostate, il metodo genera un'eccezione. La proprietà CCDTURL o CHANNEL viene considerata impostata se il suo valore è diverso da null, una stringa vuota o una stringa contenente tutti i caratteri vuoti.

Quando IBM WebSphere MQ classes for JMS trova una definizione di canale di connessione client adatta nella tabella di definizione di canale client, utilizza soltanto le informazioni estratte dalla tabella per avviare un canale MQI. Tutte le proprietà relative al canale dell'oggetto ConnectionFactory vengono ignorate.

In particolare, notare i seguenti punti se si utilizza SSL (Secure Sockets Layer):

- Un canale MQI utilizza SSL solo se la definizione del canale estratta dalla tabella di definizione del canale del client specifica il nome di una CipherSpec supportata da IBM WebSphere MQ classes for JMS.
- Una tabella di definizione del canale client contiene anche informazioni sull'ubicazione dei server LDAP (Lightweight Directory Access Protocol) che contengono i CRL (Certificate Revocation List). IBM WebSphere MQ classes for JMS utilizza solo tali informazioni per accedere ai server LDAP che contengono i CRL.
- Una tabella di definizione di canale client può contenere anche l'ubicazione di un responder OCSP (Online Certificate Status Protocol). IBM WebSphere MQ classes for JMS non può utilizzare le informazioni OCSP in un file di tabella di definizione del canale client. Tuttavia, è possibile configurare OCSP come descritto nella sezione Utilizzo di Online Certificate Protocol.

Per ulteriori informazioni sull'utilizzo di SSL con una tabella di definizione del canale client, consultare Utilizzo del client transazionale esteso con canali SSL.

Notare anche i seguenti punti se si utilizzano le uscite del canale:

- Un canale MQI utilizza solo le uscite del canale e i dati utente associati specificati dalla definizione del canale estratta dalla tabella di definizioni del canale client.
- Una definizione di canale estratta da una tabella di definizione di canale client può specificare uscite di canale scritte in Java. Ciò significa, ad esempio, che il parametro SCYEXIT nel comando DEFINE CHANNEL per creare una definizione di canale di connessione del client può specificare il nome di una classe che implementa l'interfaccia WMQSecurityExit . Allo stesso modo, il parametro SENDEXIT può specificare il nome di una classe che implementa l'interfaccia di WMQSendExit e il parametro RCVEXIT può specificare il nome di una classe che implementa l'interfaccia di WMQReceiveExit . Per ulteriori informazioni su come scrivere un'uscita di canale in Java, consultare “Scrittura delle uscite del canale in Java per WebSphere MQ classes per JMS” a pagina 913.

  È supportato anche l'uso di uscite di canale scritte in un linguaggio diverso da Java. Per informazioni su come specificare i parametri SCYEXIT, SENDEXIT e RCVEXIT nel comando DEFINE CHANNEL per le uscite canale scritte in un'altra lingua, vedere DEFINE CHANNEL.

Riconnessione client JMS automatica

Configurare il client JMS per la riconnessione automatica in seguito a un errore di rete, gestore code o server.

Utilizzare le proprietà `CONNECTIONNAMELIST` e `CLIENTRECONNECTOPTIONS` della classe `MQConnectionFactory` per configurare una connessione client per riconnettersi automaticamente in seguito a un errore di connessione o una richiesta di gestione per riconnettere le applicazioni client dopo l'arresto di un gestore code.

L'elenco completo dei nomi di connessione in un elenco connectionNameè accessibile solo ai metodi set /getconnectionNameList che possono gestire un elenco di nomi di connessione. Metodi come get / setHostname che non gestiscono elenchi di nomi, accedono al nome nell'elenco.

Le connessioni client ricollegabili automaticamente diventano ricollegabili solo una volta stabilita la connessione.

Se un'applicazione continua a funzionare correttamente dopo essere stata riconnessa automaticamente dipende dalla sua progettazione. Leggere gli argomenti correlati per comprendere come progettare client ricollegabili. Alcuni client esistenti potrebbero funzionare correttamente senza modifiche in seguito alla riconnessione automatica.

La riconnessione automatica del client non è supportata dalle classi WebSphere MQ per Java.

Per impedire che tutti i client collegati a un gestore code non riuscito si riconnettano simultaneamente, i tentativi di riconnessione vengono ritardati da intervalli parzialmente fissi e in parte casuali.

Per impostazione predefinita, i tentativi di riconnessione si verificano ai seguenti intervalli:

1. Il primo tentativo viene effettuato dopo un ritardo iniziale di un secondo, più un elemento casuale fino a 250 millisecondi.
2. Il secondo tentativo viene effettuato per due secondi, più un intervallo casuale fino a un massimo di 500 millisecondi, dopo che il primo tentativo non è riuscito.
3. Il terzo tentativo viene effettuato quattro secondi, più un intervallo casuale fino a un secondo, dopo che il secondo tentativo non è riuscito.
4. Il quarto tentativo viene effettuato otto secondi, più un intervallo casuale di un massimo di due secondi, dopo che il terzo tentativo non è riuscito.
5. Il quinto tentativo viene effettuato in 16 secondi, più un intervallo casuale di un massimo di quattro secondi, dopo che il quarto tentativo non è riuscito.
6. Il sesto tentativo, e tutti i tentativi successivi vengono effettuati 25 secondi, più un intervallo casuale di un massimo di sei secondi e 250 millisecondi, dopo che il tentativo precedente ha avuto esito negativo.

Questo processo di riconnessione continua, fino a quando il client non viene riconnesso correttamente al gestore code o fino a quando non trascorre l'intervallo massimo di riconnessione.

Se è necessario aumentare i valori predefiniti, per riflettere in modo più accurato la quantità di tempo richiesta per il ripristino del gestore code o per l'attivazione del gestore code in standby, modificare i valori di ritardo in MQCLIENT.INI utilizzando l'attributo **ReconDelay** .

Concetti correlati

Riconnessione client automatizzata

Attività correlate

Configurazione di un client utilizzando un file di configurazione

Condivisione di una connessione TCP/IP in IBM WebSphere MQ classes for JMS

È possibile creare più istanze di un canale MQI per condividere una connessione TCP/IP singola.

Le applicazioni in esecuzione all'interno dello stesso JRE (Java runtime environment) e che utilizzano le classi IBM WebSphere MQ classes for JMS o l'adattatore di risorse IBM WebSphere MQ per connettersi a un gestore code utilizzando il trasporto CLIENT, possono condividere la stessa istanza del canale.

Esiste una relazione uno - a - uno tra le istanze del canale e le connessioni TCP/IP. Viene creata una connessione TCP/IP per ogni istanza di canale.

Se un canale è stato definito con il parametro **SHARECNV** impostato su un valore superiore a 1, tale numero di conversazioni può condividere un'istanza del canale. Per abilitare una factory di connessione o una specifica di attivazione all'utilizzo di questa funzione, impostare la proprietà **SHARECONVALLOWED** su YES.

Ogni sessione JMS e connessione JMS creata da un'applicazione JMS crea la propria conversazione con il gestore code.

Quando viene avviata una specifica di attivazione, le classi IBM WebSphere MQ per l'adattatore di risorse JMS avviano una conversazione con il gestore code per la specifica di attivazione da utilizzare. Ogni sessione del server nel pool di sessioni del server associato alla specifica di attivazione avvia anche una conversazione con il gestore code.

L'attributo SHARECNV è un approccio ottimale alla condivisione delle connessioni. Pertanto, quando un valore SHARECNV maggiore di 0 viene utilizzato con le IBM WebSphere MQ classes for JMS, non è garantito che una nuova richiesta di connessione condivida sempre una connessione già stabilita.

Calcolo del numero di istanze del canale

Utilizzare le formule riportate di seguito per determinare il numero massimo di istanze di canale create da un'applicazione:

Specifiche di attivazione

Numero di istanze del canale = (<**maxPoolDepth**> + 1) / <**SHARECNV**>

Dove <**maxPoolDepth**> è il valore della proprietà **maxPoolDepth** e <**SHARECNV**> è il valore della proprietà **SHARECNV** sul canale utilizzato dalla specifica di attivazione.

Altre applicazioni JMS

Numero di istanze del canale = (< connessioni JMS> + sessioni JMS>) / <**SHARECNV**>

Dove < Connessioni JMS> è il numero di connessioni create dall'applicazione, dove < Sessioni JMS> è il numero di sessioni JMS create dall'applicazione e <**SHARECNV**> è il valore della proprietà **SHARECNV** sul canale utilizzato dalla specifica di attivazione.

Esempi

I seguenti esempi mostrano come utilizzare le formule per calcolare il numero di istanze del canale create su un gestore code dalle applicazioni utilizzando l'adattatore di risorse IBM WebSphere MQ classes for JMS o IBM WebSphere MQ classes for JMS .

Esempio di applicazione JMS
Una connessione dell'applicazione JMS si connette a un gestore code utilizzando il trasporto CLIENT e crea una connessione JMS e tre sessioni JMS. Il canale che l'applicazione sta utilizzando per connettersi al gestore code ha la proprietà **SHARECNV** impostata sul valore 10. Quando l'applicazione è in esecuzione, ci sono quattro conversazioni tra l'applicazione e il gestore code e un'istanza di canale. Le quattro conversazioni condividono tutte l'istanza del canale.

Esempio di specifica di attivazione
Una specifica di attivazione si connette a un gestore code utilizzando il trasporto CLIENT. La specifica di attivazione è configurata con la proprietà **maxPoolDepth** impostata su 10. Il canale che la specifica di attivazione è configurata per utilizzare ha la proprietà **SHARECNV** impostata su 10. Quando la specifica di attivazione è in esecuzione ed elabora 10 messaggi contemporaneamente, il numero di conversazioni tra la specifica di attivazione e il gestore code è 11 (10 conversazioni per la sessione server e una per la specifica di attivazione). Il numero di istanze del canale utilizzate dalla specifica di attivazione è 2.

Esempio di specifica di attivazione
Una specifica di attivazione si connette a un gestore code utilizzando il trasporto CLIENT. La specifica di attivazione è configurata con la proprietà **maxPoolDepth** impostata su 5. Il canale che la specifica di attivazione è configurata per utilizzare ha la proprietà **SHARECNV** impostata su 0. Quando la specifica di attivazione è in esecuzione ed elabora 5 messaggi contemporaneamente, il numero di conversazioni tra la specifica di attivazione e il gestore code è 6 (cinque conversazioni per le sessioni del server e una per la specifica di attivazione). Il numero di istanze del canale utilizzate dalla specifica di attivazione è 6, poiché la proprietà **SHARECNV** sul canale è impostata su 0, ogni conversazione utilizza la propria istanza del canale.

Specifica di un intervallo di porte per connessioni client nelle classi WebSphere MQ per JMS

Utilizzare la proprietà LOCALADDRESS per specificare un intervallo di porte a cui l'applicazione può collegarsi.

Quando un'applicazione WebSphere MQ cerca di connettersi a un gestore code WebSphere MQ in modalità client, un firewall potrebbe consentire solo le connessioni che hanno origine da porte specificate

o da un intervallo di porte. In questa situazione, è possibile utilizzare la proprietà LOCALADDRESS di un oggetto factory ConnectionFactory, QueueConnectiono TopicConnectionper specificare una porta o un intervallo di porte a cui l'applicazione può collegarsi.

È possibile impostare la proprietà LOCALADDRESS utilizzando lo strumento di amministrazione JMS WebSphere MQ o richiamando il metodo setLocalAddress () in un'applicazione JMS. Di seguito viene riportato un esempio di impostazione della proprietà dall'interno di un'applicazione:

```
mqConnectionFactory.setLocalAddress("192.0.2.0(2000,3000)");
```

Quando l'applicazione si connette successivamente a un gestore code, l'applicazione si collega a un indirizzo IP locale e a un numero di porta compresi nell'intervallo tra 192.0.2.0(2000) e 192.0.2.0(3000).

In un sistema con più di un'interfaccia di rete, è anche possibile utilizzare la proprietà LOCALADDRESS per specificare quale interfaccia di rete deve essere utilizzata per una connessione.

Per una connessione in tempo reale a un broker, la proprietà LOCALADDRESS è rilevante solo quando viene utilizzato il multicast. In questo caso, è possibile utilizzare la proprietà per specificare quale interfaccia di rete locale deve essere utilizzata per una connessione, ma il valore della proprietà non deve contenere un numero di porta o un intervallo di numeri di porta.

Gli errori di connessione potrebbero verificarsi se si limita l'intervallo di porte. Se si verifica un errore, viene generata una JMSException con una MQException incorporata che contiene il codice motivo WebSphere MQ MQRC_Q_MGR_NOT_AVAILABLE e il seguente messaggio:

```
Tentativo di connessione socket rifiutato a causa delle limitazioni LOCAL_ADDRESS_PROPERTY
```

Si potrebbe verificare un errore se tutte le porte nell'intervallo specificato sono in uso o se l'indirizzo IP, il nome host o il numero di porta specificati non sono validi (ad esempio, un numero di porta negativo).

Poiché le classi WebSphere MQ per JMS potrebbero creare connessioni diverse da quelle richieste da un'applicazione, specificare sempre un intervallo di porte. In generale, ogni sessione creata da un'applicazione richiede una porta e WebSphere MQ classes per JMS potrebbe richiedere tre o quattro porte aggiuntive. Se si verifica un errore di connessione, aumentare l'intervallo di porte.

Il pool di connessioni, utilizzato per impostazione predefinita nelle classi WebSphere MQ per JMS, potrebbe avere un effetto sulla velocità con cui le porte possono essere riutilizzate. Di conseguenza, potrebbe verificarsi un errore di connessione mentre le porte vengono liberate.

Compressione dei canali nelle classi WebSphere MQ per JMS

Un'applicazione WebSphere MQ classes può utilizzare le funzioni WebSphere MQ per comprimere un'intestazione o i dati di un messaggio.

La compressione dei dati che fluiscono su un canale WebSphere MQ può migliorare le prestazioni del canale e ridurre il traffico di rete. Utilizzando la funzione fornita con WebSphere MQ, è possibile comprimere i dati che fluiscono sui canali di messaggi e sui canali MQI. In entrambi i tipi di canale, è possibile comprimere i dati di intestazione e di messaggio indipendentemente l'uno dall'altro. Per impostazione predefinita, non viene compresso alcun dato su un canale.

Un'applicazione WebSphere MQ classes per JMS application specifica le tecniche che possono essere utilizzate per comprimere i dati di intestazione o di messaggio su una connessione creando un oggetto java.util.Collection . Ogni tecnica di compressione è un oggetto intero nella raccolta e l'ordine in cui l'applicazione aggiunge le tecniche di compressione alla raccolta è l'ordine in cui le tecniche di compressione vengono negoziate con il gestore code quando l'applicazione crea la connessione. L'applicazione può quindi passare la raccolta a un oggetto ConnectionFactory richiamando il metodo setHdrCompList(), per i dati di intestazione o il metodo setMsgCompList(), per i dati del messaggio. Quando l'applicazione è pronta, può creare la connessione.

I seguenti frammenti di codice illustreranno l'approccio descritto. Il primo frammento di codice mostra come implementare la compressione dei dati di intestazione:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(WMQConstants.WMQ_COMPHDR_SYSTEM));
.
.
```

```
.
((MQConnectionFactory) cf).setHdrCompList(headerComp);
.
.
.
connection = cf.createConnection();
```

Il secondo frammento di codice mostra come implementare la compressione dei dati del messaggio:

```
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();
```

Nel secondo esempio, le tecniche di compressione vengono negoziate nell'ordine RLE, quindi ZLIBHIGH, quando viene creato il collegamento. La tecnica di compressione selezionata non può essere modificata durante la durata dell'oggetto Connection. Per utilizzare la compressione su una connessione, è necessario richiamare i metodi setHdrCompList() e setMsgCompList() prima di creare l'oggetto Connection.

Inserimento asincrono dei messaggi nelle classi IBM WebSphere MQ per JMS

Normalmente, quando un'applicazione invia messaggi a una destinazione, l'applicazione deve attendere che il gestore code confermi di aver elaborato la richiesta. È possibile migliorare le prestazioni della messaggistica in alcune circostanze scegliendo invece di inserire i messaggi in modo asincrono. Quando un'applicazione inserisce un messaggio in maniera asincrona, il gestore code non restituisce l'esito positivo o negativo di ciascuna chiamata, ma è possibile controllare periodicamente la presenza di errori.

Se una destinazione restituisce il controllo all'applicazione, senza determinare se il gestore code ha ricevuto il messaggio in modo sicuro, dipende dalle seguenti proprietà:

- La proprietà destinazione JMS PUTSYNCALLOWED (nome breve - PAALD).

  PUTASYNCALLOWED controlla se le applicazioni JMS possono inserire i messaggi in modo asincrono, se la coda o l'argomento sottostante rappresentato dalla destinazione JMS, consente questa opzione.
- La proprietà della coda o dell'argomento IBM WebSphere MQ DEFPRESP (Tipo di risposta di inserimento predefinito).

  DEFPRESP specifica se le applicazioni che inserono i messaggi nella coda o che pubblicano i messaggi nell'argomento possono utilizzare la funzionalità di inserimento asincrono.

La seguente tabella mostra i possibili valori per le proprietà PUTASYNCALLOWED e DEFPRESP e quali valori sono richiesti per la funzionalità di inserimento asincrono da abilitare:

Tabella 131. Proprietà PUTASYNCALLOWED e DEFPRESP che determinano se i messaggi vengono inseriti in modo asincrono.

proprietà della coda WebSphere MQ	**PUTASYNCALLOWED = NO**	**PUTASYNCALLOWED = SI**	**PUTASYNCALLOWED = AS_DEST o AS_Q_DEF o AS_T_DEF**
DEFPRESP=SYNC	Funzionalità di inserimento asincrono non abilitata	Funzionalità put asincrono abilitata	Funzionalità di inserimento asincrono non abilitata
DEFPRESP=ASYNC	Funzionalità di inserimento asincrono non abilitata	Funzionalità put asincrono abilitata	Funzionalità put asincrono abilitata

Per i messaggi inviati in una sessione sottoposta a transazione, l'applicazione determina se il gestore code ha ricevuto i messaggi in modo sicuro quando richiama `commit()`.

Se un'applicazione invia messaggi persistenti all'interno di una sessione sottoposta a transazione e uno o più messaggi non vengono ricevuti in modo sicuro, la transazione non riesce a eseguire il commit e genera un'eccezione. Tuttavia, se un'applicazione invia messaggi non persistenti all'interno di una sessione

sottoposta a transazione e uno o più messaggi non vengono ricevuti in modo sicuro, la transazione viene sottoposta a commit correttamente. L'applicazione non riceve alcun feedback sul fatto che i messaggi non persistenti non sono arrivati in modo sicuro.

Per i messaggi non persistenti inviati in una sessione non sottoposta a transazione, la proprietà SENDCHECKCOUNT dell'oggetto *ConnectionFactory* specifica il numero di messaggi da inviare prima che IBM WebSphere MQ classes for JMS verifichi che il gestore code abbia ricevuto i messaggi in modo sicuro.

Se un controllo rileva che uno o più messaggi non sono stati ricevuti in modo sicuro e l'applicazione ha registrato un listener di eccezioni con la connessione, IBM WebSphere MQ classes for JMS richiama il metodo `onException()` del listener di eccezioni per inoltrare un'eccezione JMS all'applicazione.

L'eccezione JMS ha un codice di errore JMSWMQ0028 e questo codice visualizza il seguente messaggio:

```
At least one asynchronous put message failed or gave a warning.
```

L'eccezione JMS ha anche un'eccezione collegata che fornisce ulteriori dettagli. Il valore predefinito della proprietà SENDCHECKCOUNT è zero, il che significa che non vengono eseguiti tali controlli.

Questa ottimizzazione è molto utile per un'applicazione che si connette a un gestore code in modalità client e deve inviare una sequenza di messaggi in rapida successione, ma non richiede un feedback immediato dal gestore code per ogni messaggio inviato. Tuttavia, un'applicazione può ancora utilizzare questa ottimizzazione anche se si connette a un gestore code in modalità di bind, ma il vantaggio delle prestazioni previsto non è lo stesso.

Utilizzo della lettura anticipata con le classi WebSphere MQ per JMS

La funzionalità di lettura anticipata fornita da WebSphere MQ consente l'invio di messaggi non persistenti ricevuti all'esterno di una transazione a IBM WebSphere MQ classes for JMS prima che un'applicazione li richieda. Il IBM WebSphere MQ classes for JMS memorizza i messaggi in un buffer interno e li trasmette all'applicazione quando l'applicazione li richiede.

Le applicazioni IBM WebSphere MQ classes for JMS che utilizzano `MessageConsumers` o `MessageListeners` per ricevere i messaggi da una destinazione esterna a una transazione possono utilizzare la funzionalità di lettura anticipata. L'utilizzo della lettura anticipata consente alle applicazioni che utilizzano questi oggetti di beneficiare di prestazioni migliori quando ricevono messaggi.

Se un'applicazione che utilizza `MessageConsumers` o `MessageListeners` può utilizzare la lettura anticipata dipende dalle seguenti proprietà:

- La proprietà di destinazione JMS READAHEADALLOWED (nome breve - RAALD) READAHEADALLOWED controlla se le applicazioni JMS possono utilizzare la lettura anticipata durante il richiamo o l'esplorazione di messaggi non persistenti al di fuori di una transazione, se la coda sottostante o l'argomento rappresentato dalla destinazione JMS consente questa opzione.
- La coda IBM WebSphere MQ o la proprietà dell'argomento DEFREADA (lettura anticipata predefinita). DEFREADA specifica se le applicazioni che stanno ricevendo o sfogliando messaggi non persistenti al di fuori di una transazione possono utilizzare la lettura anticipata.

La seguente tabella mostra i possibili valori per le proprietà READAHEADALLOWED e DEFREADA e quali valori sono richiesti per abilitare la funzione di lettura anticipata:

Tabella 132. Proprietà READAHEADALLOWED e DEFREADA che determinano se la lettura anticipata viene utilizzata durante la ricezione o l'esplorazione di messaggi non persistenti all'esterno di una transazione.

proprietà di destinazione WebSphere MQ	**READAHEADALLOWED = SÌ**	**READAHEADALLOWED = NO**	**AS_DEST o AS_Q_DEF o AS_T_DEF**
proprietà della coda WebSphere MQ			
DEFREADA = NO	Funzionalità di lettura anticipata abilitata	Funzionalità di lettura anticipata non abilitata	Funzionalità di lettura anticipata non abilitata

Tabella 132. Proprietà READAHEADALLOWED e DEFREADA che determinano se la lettura anticipata viene utilizzata durante la ricezione o l'esplorazione di messaggi non persistenti all'esterno di una transazione. (Continua)

proprietà di destinazione WebSphere MQ	**READAHEADALLOWED = SÌ**	**READAHEADALLOWED = NO**	**AS_DEST o AS_Q_DEF o AS_T_DEF**
DEFREADA = SI	Funzionalità di lettura anticipata abilitata	Funzionalità di lettura anticipata non abilitata	Funzionalità di lettura anticipata abilitata
DEFREADA = DISABILITATO	Funzionalità di lettura anticipata non abilitata	Funzionalità di lettura anticipata non abilitata	Funzionalità di lettura anticipata non abilitata

Se la funzionalità di lettura anticipata è abilitata, quando `MessageConsumer` o `MessageListener` viene creato da un'applicazione, IBM WebSphere MQ classes for JMS crea un buffer interno per la destinazione monitorata da `MessageConsumer` o `MessageListener` . Esiste un buffer interno per ogni `MessageConsumer` o `MessageListener`. Il gestore code avvia l'invio di messaggi non persistenti a IBM WebSphere MQ classes for JMS quando l'applicazione chiama uno dei seguenti metodi:

- `MessageConsumer.receive()`
- `MessageConsumer.receive(long timeout)`
- `MessageConsumer.receiveNoWait()`
- `Session.setMessageListener(MessageListener listener)`

Il IBM WebSphere MQ classes for JMS restituisce automaticamente il primo messaggio all'applicazione, mediante la chiamata al metodo effettuata dall'applicazione. Gli altri messaggi non persistenti vengono memorizzati da IBM WebSphere MQ classes for JMS nel buffer interno creato per la destinazione. Quando l'applicazione richiede l'elaborazione del messaggio successivo, IBM WebSphere MQ classes for JMS restituirà il successivo messaggio nel buffer interno.

Il IBM WebSphere MQ classes for JMS richiede più messaggi non persistenti dal gestore code quando il buffer interno è vuoto.

Il buffer interno utilizzato da IBM WebSphere MQ classes for JMS viene eliminato quando un'applicazione chiude un `MessageConsumer`o la sessione JMS a cui è associato un `MessageListener` .

Per `MessageConsumers`, i messaggi non elaborati nel buffer interno vengono persi.

Quando si utilizza `MessageListeners`, ciò che accade ai messaggi nel buffer interno dipende dalla proprietà di destinazione JMS READAHEADCLOSEPOLICY (nome breve - RACP). Il valore predefinito della proprietà è DELIVER_ALL, che significa che la sessione JMS utilizzata per creare `MessageListener` non viene chiusa fino a quando tutti i messaggi nel buffer interno non vengono consegnati all'applicazione. Se la proprietà è impostata su DELIVER_CURRENT, la sessione JMS verrà chiusa dopo che il messaggio corrente è stato elaborato dall'applicazione e tutti i restanti messaggi nel buffer interno vengono eliminati.

Pubblicazioni conservate in WebSphere MQ classes per JMS

È possibile configurare un client WebSphere MQ per JMS per utilizzare le pubblicazioni conservate.

Un autore (publisher) può specificare che una copia di una pubblicazione deve essere conservata in modo che possa essere inviata ai futuri sottoscrittori (subscriber) che registrano un interesse per l'argomento. Questa operazione viene eseguita nelle classi WebSphere MQ per JMS impostando la proprietà intera JMS_IBM_RETAIN sul valore 1. Le costanti sono state definite per questi valori nell'interfaccia com.ibm.msg.client.jms.JmsConstants . Ad esempio, se è stato creato un messaggio *msg*, per impostarlo come pubblicazione conservata utilizzare il seguente codice:

```
// set as a retained publication
msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION);
```

È ora possibile inviare il messaggio come normale. JMS_IBM_RETAIN può anche essere interrogato in un messaggio ricevuto. È quindi possibile verificare se un messaggio ricevuto è una pubblicazione conservata.

Supporto XA nelle classi WebSphere MQ per JMS

JMS supporta le transazioni conformi a XA nei bind e nelle modalità client con un gestore transazioni supportato.

Se si richiede la funzione XA in un ambiente del server delle applicazioni, è necessario configurare l'applicazione in modo appropriato. Fare riferimento alla documentazione del proprio server delle applicazioni per informazioni su come configurare le applicazioni per utilizzare le transazioni distribuite.

Utilizzo di una connessione in tempo reale a un broker di WebSphere Event Broker o WebSphere Message Broker

Le classi WebSphere MQ per l'applicazione JMS possono utilizzare una connessione in tempo reale a un broker di WebSphere Event Broker o WebSphere Message Broker per la messaggistica di pubblicazione / sottoscrizione. Sia il broker che le classi WebSphere MQ per JMS devono essere configurati per abilitare una connessione in tempo reale.

Quando un'applicazione utilizza una connessione in tempo reale ad un broker di WebSphere Event Broker o WebSphere Message Broker, l'applicazione e il broker si scambiano i messaggi utilizzando WebSphere MQ Real-Time Transport. A seconda della configurazione, i messaggi possono anche essere consegnati all'applicazione utilizzando WebSphere MQ Multicast Transport.

Per informazioni su come un'applicazione può collegarsi a un gestore code WebSphere MQ e utilizzare WebSphere MQ Enterprise Transport per scambiare messaggi con un broker di WebSphere Event Broker o WebSphere Message Broker, Consultare la documentazione per le release precedenti di WebSphere MQ classes per JMS. Per poter utilizzare WebSphere MQ Enterprise Transport, un'applicazione deve connettersi a un gestore code utilizzando una factory di connessione in esecuzione in modalità di migrazione del provider di messaggistica WebSphere MQ .

Configurazione di un Broker di WebSphere Event Broker o WebSphere Message Broker per una connessione in tempo reale

Per fare in modo che un'applicazione WebSphere MQ classes for JMS utilizzi una connessione in tempo reale ad un broker di WebSphere Event Broker o WebSphere Message Broker, è necessario configurare il broker creando e distribuendo un flusso di messaggi per leggere i messaggi dalla porta TCP/IP su cui il broker è in ascolto e pubblicare i messaggi. In base ai propri requisiti, potrebbe essere necessario configurare il broker in modi aggiuntivi.

Per configurare il Broker, è necessario creare e distribuire uno dei seguenti flussi di messaggi:

- Un flusso di messaggi che contiene un nodo di elaborazione dei messaggi del flusso Real -timeOptimized
- Un flusso di messaggi che contiene un nodo di elaborazione messaggi Real -timeInput e un nodo di elaborazione messaggi Publication

È necessario configurare il nodo Real -timeOptimizedFlow o Real -timeInput per l'ascolto sulla porta TCP/IP utilizzata per le connessioni in tempo reale. Per impostazione predefinita, il numero di porta per le connessioni in tempo reale è 1506.

È inoltre necessario configurare il broker se si dispone di uno dei seguenti requisiti:

- Se si desidera che l'applicazione si connetta al broker utilizzando l'autenticazione SSL (Secure Sockets Layer)
- Se si desidera che l'applicazione si connetta al broker utilizzando il tunnelling HTTP
- Se si desidera che i messaggi vengano consegnati ad un consumatore di messaggi utilizzando multicast

Per informazioni su come configurare un Broker, consultare la documentazione del prodotto *WebSphere Event Broker* o *WebSphere Message Broker product documentation*.

Configurazione delle classi WebSphere MQ per JMS per una connessione in tempo reale a un broker di WebSphere Event Broker o WebSphere Message Broker

Per un'applicazione WebSphere MQ classes for JMS, che utilizza una connessione in tempo reale a un Broker di WebSphere Event Broker o WebSphere Message Broker, WebSphere MQ classes for JMS deve essere configurato impostando determinate proprietà della factory di connessione. In base ai requisiti, potrebbe essere necessario configurare le classi WebSphere MQ per JMS in modi aggiuntivi.

Per configurare WebSphere MQ classes per JMS, è necessario impostare le seguenti proprietà della factory di connessione:

- La proprietà TRANSPORT deve essere impostata su DIRECT.

  Tuttavia, perché un'applicazione possa connettersi utilizzando il tunnelling HTTP, la proprietà TRANSPORT deve essere impostata su DIRECTHTTP. Vedere “Utilizzo del tunnelling HTTP” a pagina 927.
- La proprietà HOSTNAME deve essere impostata sul nome host o sull'indirizzo IP del sistema su cui è in esecuzione il broker.
- La proprietà PORT deve essere impostata sul numero della porta su cui il broker è in ascolto per le connessioni in tempo reale.

Un'applicazione può impostare queste proprietà dinamicamente in fase di runtime utilizzando le estensioni JMS IBM o le estensioni JMS WebSphere MQ . In alternativa, se il factory di connessione è un oggetto amministrato, un amministratore può impostare queste proprietà utilizzando lo strumento di amministrazione JMS WebSphere MQ o WebSphere MQ Explorer.

Per informazioni sulle proprietà e sui metodi utilizzati dalle applicazioni per impostare i valori, consultare Proprietà degli oggetti IBM WebSphere MQ classes for JMS. Per informazioni su come utilizzare lo strumento di gestione JMS WebSphere MQ , consultare “Utilizzo dello strumento di amministrazione JMS WebSphere MQ” a pagina 936. Per informazioni su come utilizzare WebSphere MQ Explorer, consultare la guida fornita con WebSphere MQ Explorer.

Se si dispone di uno dei seguenti requisiti, WebSphere MQ classes per JMS richiede ulteriore configurazione:

- Se si desidera che un'applicazione si connetta al broker utilizzando l'autenticazione SSL (Secure Sockets Layer)
- Se si desidera che un'applicazione si connetta al broker utilizzando il tunnelling HTTP
- Se si desidera che un'applicazione si connetta al broker tramite un server proxy
- Se si desidera che i messaggi vengano consegnati ad un consumatore di messaggi utilizzando multicast

Le seguenti sezioni descrivono come configurare le classi WebSphere MQ per JMS per ognuno di questi requisiti.

Utilizzo dell'autenticazione SSL (Secure Sockets Layer)

L'autenticazione SSL può essere utilizzata su una connessione in tempo reale a un broker. Per questo tipo di connessione è supportata solo l'autenticazione. Non è possibile utilizzare SSL per codificare e decodificare i dati del messaggio che fluiscono tra l'applicazione e il broker o per rilevare la manomissione dei dati.

Notare la differenza tra questa situazione e quella in cui un'applicazione si connette a un gestore code in modalità client. In quest' ultimo caso, è possibile utilizzare il supporto SSL di WebSphere MQ per codificare e decodificare i dati del messaggio che passano tra l'applicazione e il gestore code e per rilevare la manomissione dei dati, oltre che per fornire l'autenticazione.

Se si desidera proteggere i dati del messaggio su una connessione in tempo reale ad un broker, è possibile utilizzare la funzione fornita dal broker. È possibile assegnare un valore di qualità di protezione (QoP) a ciascun argomento con i messaggi che si desidera proteggere. È quindi possibile selezionare un livello diverso di protezione del messaggio per ciascun argomento. Per ulteriori informazioni sulla protezione dei

messaggi fornita da un broker, consultare la *documentazione del prodotto WebSphere Event Broker* o la *WebSphere Message Broker*.

Per utilizzare l'autenticazione SSL su una connessione in tempo reale a un broker, la proprietà DIRECTAUTH della factory di connessione deve essere impostata su CERTIFICATE.

Se si desidera utilizzare SSL per l'autenticazione reciproca, la proprietà Tipo protocollo di autenticazione del broker deve specificare l'opzione R per SSL simmetrico. Se si desidera utilizzare SSL solo per l'autenticazione del broker, la proprietà Tipo protocollo di autenticazione del broker deve specificare l'opzione S per SSL asimmetrico. Ma, in questo caso, l'applicazione deve connettersi al broker richiamando createConnection() con un ID utente e una password come parametri, come nel seguente esempio:

```
factory.createConnection("user1", "user1pw");
```

Il broker utilizza quindi l'ID utente e la password, invece di SSL, per autenticare l'applicazione. Per ulteriori informazioni su come configurare il broker per l'autenticazione SSL, consultare la *WebSphere Event Broker* o la *WebSphere Message Broker*.

Note:

1. Il valore della proprietà DIRECTAUTH determina se l'autenticazione SSL viene utilizzata su una connessione in tempo reale a un broker, non il valore della proprietà SSLCIPHERSUITE.
2. Quando l'autenticazione SSL viene utilizzata su una connessione in tempo reale a un Broker, le proprietà SSLPEERNAME e SSLCRL sono utilizzate per eseguire gli stessi controlli eseguiti quando un'applicazione si connette a un gestore code in modalità client.
3. Le classi WebSphere MQ per JMS possono utilizzare la stessa configurazione di keystore e truststore JSSE (Java Secure Socket Extension) per fornire il supporto SSL in una delle seguenti situazioni:
   - Quando un'applicazione utilizza una connessione in tempo reale a un broker
   - Quando un'applicazione si collega a un gestore code in modalità client

Utilizzo del tunnelling HTTP

Un'applicazione WebSphere MQ classes for JMS può connettersi a un broker utilizzando il tunnelling HTTP, il che significa che l'applicazione si connette al broker utilizzando il protocollo HTTP come se si connettesse a un sito web.

Per utilizzare il tunnelling HTTP su una connessione in tempo reale a un broker, la proprietà TRANSPORT della produzione connessioni deve essere impostata su DIRECTHTTP.

Il tunnelling HTTP non può essere utilizzato insieme all'autenticazione SSL, alla connessione tramite un server proxy o alla consegna di messaggi mediante multicast. La versione supportata del protocollo HTTP Š 1.0. La versione HTTP 1.1 non è supportata.

Connessione tramite un server proxy

Le classi WebSphere MQ per l'applicazione JMS possono utilizzare una connessione in tempo reale a un broker collegandosi tramite un server proxy. WebSphere MQ classes for JMS si collega direttamente al server proxy e utilizza il protocollo Internet definito in RFC 2817 per chiedere al server proxy di inoltrare al broker la richiesta di connessione.

Per connettersi a un broker tramite un server proxy, è necessario impostare le seguenti proprietà della factory di connessione:

- La proprietà PROXYHOSTNAME deve essere impostata sul nome host o sull'indirizzo IP del sistema su cui è in esecuzione il server proxy.
- La proprietà PROXYPORT deve essere impostata sul numero della porta su cui il server proxy è in ascolto.

Se la proprietà PROXYHOSTNAME non è impostata o è impostata sulla stringa vuota, WebSphere MQ classes for JMS tenta di connettersi direttamente al broker utilizzando solo le proprietà HOSTNAME e PORT e non tenta di connettersi tramite un server proxy.

Consegna dei messaggi mediante multicast

Utilizzando una connessione in tempo reale a un broker, i messaggi possono essere consegnati a un consumatore di messaggi utilizzando multicast.

Per abilitare il multicast, la proprietà MULTICAST dell'oggetto Argomento deve essere impostata sull'opzione multicast richiesta. In alternativa, se la proprietà MULTICAST dell'oggetto Argomento è impostata su ASCF, la proprietà MULTICAST della factory di connessione deve essere impostata sull'opzione multicast richiesta.

WebSphere MQ classes for JMS supporta i protocolli multicast PTL (Packet Transfer Layer) e PGM (Pragmatiche General Multicast), e include il supporto per entrambe le implementazioni del protocollo PGM, PGM/IP e PGM UDP incapsulato. Tuttavia, il supporto PGM/IP è disponibile solo sulle seguenti piattaforme:

- AIX (solo 32 bit)
- Linux (piattaformax86)
- Linux (piattaforma zSeries , solo 32 bit)
- Solaris SPARC (solo 32 bit)
- Windows (solo 32 bit)
- z/OS

WebSphere MQ classes per JMS Application Server Facilities

Questo argomento descrive il modo in cui WebSphere MQ classes per JMS implementa la classe ConnectionConsumer e la funzionalità avanzata nella classe Session. Riepiloga anche la funzione di un pool di sessioni server.

WebSphere MQ classes for JMS supporta le ASF (Application Server Facilities) specificate in *Java Message Service Specification, Versione 1.1* (consultare il sito Web Java di Sun all'indirizzo https://java.sun.com). Questa specifica identifica tre ruoli all'interno di questo modello di programmazione:

- **Il provider JMS** fornisce ConnectionConsumer e la funzionalità di sessione avanzata.
- **Il server delle applicazioni** fornisce la funzionalità ServerSessionPool e ServerSession .
- **L'applicazione client** utilizza la funzionalità fornita dal provider JMS e dal server delle applicazioni.

Le informazioni contenute in questo argomento non si applicano se un'applicazione utilizza una connessione in tempo reale ad un broker.

ConnectionConsumer JMS

L'interfaccia ConnectionConsumer fornisce un metodo ad alte prestazioni per consegnare i messaggi contemporaneamente a un pool di thread.

La specifica JMS consente a un server delle applicazioni di integrarsi strettamente con un'implementazione JMS utilizzando l'interfaccia `ConnectionConsumer` . Questa funzione fornisce l'elaborazione simultanea dei messaggi. Generalmente, un server delle applicazioni crea un lotto di thread e l'implementazione JMS rende i messaggi disponibili per tali thread. Un server delle applicazioni JMS (come WebSphere Application Server) può utilizzare questa funzione per fornire una funzionalità di messaggistica di alto livello, come i bean basati sui messaggi.

Le applicazioni normali non utilizzano ConnectionConsumer, ma i client JMS esperti potrebbero utilizzarlo. Per tali client, ConnectionConsumer fornisce un metodo ad alte prestazioni per consegnare i messaggi contemporaneamente a un lotto di thread. Quando un messaggio arriva su una coda o su un argomento, JMS seleziona un thread dal pool e gli consegna un batch di messaggi. A tale scopo, JMS esegue un metodo MessageListenerassociato `onMessage()` .

È possibile ottenere lo stesso risultato creando più oggetti Session e MessageConsumer , ciascuno con un MessageListenerregistrato. Tuttavia, ConnectionConsumer fornisce prestazioni migliori, un minore utilizzo delle risorse e una maggiore flessibilità. In particolare, è richiesto un minor numero di oggetti Session.

Pianificazione di un'applicazione con ASF

In questa sezione viene descritto come pianificare un'applicazione che include:

- “Principi generali per la messaggistica point - to - point utilizzando ASF” a pagina 929
- “Principi generali per la messaggistica di pubblicazione / sottoscrizione utilizzando ASF” a pagina 930
- “Rimozione dei messaggi dalla coda in ASF” a pagina 930
- Gestione dei messaggi dannosi in ASF. Consultare “Gestione dei messaggi non elaborabili in IBM WebSphere MQ classes for JMS” a pagina 891.

Principi generali per la messaggistica point - to - point utilizzando ASF

Utilizzare questo argomento per informazioni generali sulla messaggistica point - to - point utilizzando ASF.

Quando un'applicazione crea un ConnectionConsumer da un oggetto QueueConnection , specifica un oggetto coda JMS e una stringa selettore. ConnectionConsumer inizia quindi a fornire messaggi alle sessioni nel pool ServerSessionassociato. I messaggi arrivano sulla coda e, se corrispondono al selettore, vengono consegnati alle sessioni nel pool ServerSessionassociato.

In termini WebSphere MQ , l'oggetto coda fa riferimento a un QLOCAL o a un QALIAS sul gestore code locale. Se si tratta di un QALIAS, tale QALIAS deve fare riferimento a un QLOCAL. Il QLOCAL WebSphere MQ completamente risolto è noto come *QLOCAL sottostante*. Un ConnectionConsumer viene definito *attivo* se non è chiuso e il relativo QueueConnection principale è avviato.

È possibile eseguire più ConnectionConsumers, ognuno con selettori differenti, rispetto allo stesso QLOCAL sottostante. Per mantenere le prestazioni, i messaggi indesiderati non devono essere accumulati sulla coda. I messaggi indesiderati sono quelli per cui nessun ConnectionConsumer attivo dispone di un selettore corrispondente. È possibile impostare il factory QueueConnectionin modo che questi messaggi indesiderati vengano rimossi dalla coda (per i dettagli, consultare “Rimozione dei messaggi dalla coda in ASF” a pagina 930). È possibile impostare questo comportamento in due modi:

- Utilizzare lo strumento di amministrazione JMS per impostare il factory QueueConnectionsu MRET (NO).
- Nel programma, utilizzare:

```
MQQueueConnectionFactory.setMessageRetention(WMQConstants.WMQ_MRET_NO)
```

Se non si modifica questa impostazione, il valore predefinito è di conservare tali messaggi indesiderati nella coda.

Quando si configura il gestore code WebSphere MQ , considerare i seguenti punti:

- Il QLOCAL sottostante deve essere abilitato per l'input condiviso. Per eseguire questa operazione, utilizzare il seguente comando MQSC:

```
ALTER QLOCAL(your.qlocal.name) SHARE GET(ENABLED)
```

- Il gestore code deve avere una coda di messaggi non recapitabili abilitata. Se un ConnectionConsumer riscontra un problema quando inserisce un messaggio nella coda di messaggi non recapitabili, la consegna del messaggio dal QLOCAL sottostante si arresta. Per definire una coda di messaggi non recapitabili, utilizzare:

```
ALTER QMGR DEADQ(your.dead.letter.queue.name)
```

- L'utente che esegue ConnectionConsumer deve disporre dell'autorità per eseguire MQOPEN con MQOO_SAVE_ALL_CONTEXT e MQOO_PASS_ALL_CONTEXT. Per i dettagli, consultare la documentazione di WebSphere MQ per la piattaforma specifica.
- Se i messaggi indesiderati vengono lasciati nella coda, le prestazioni del sistema vengono ridotte. Pertanto, pianificare i selettori dei messaggi in modo che tra di essi, ConnectionConsumers rimuoverà tutti i messaggi dalla coda.

Per i dettagli sui comandi MQSC, consultare Guida di riferimento a MQSC.

Principi generali per la messaggistica di pubblicazione / sottoscrizione utilizzando ASF

ConnectionConsumers riceve messaggi per un argomento specificato. Un ConnectionConsumer può essere durevole o non durevole. È necessario specificare la coda o le code utilizzate da ConnectionConsumer .

Quando un'applicazione crea un ConnectionConsumer da un oggetto TopicConnection , specifica un oggetto Topic e una stringa selettore. Il ConnectionConsumer inizia quindi a ricevere i messaggi che corrispondono al selettore su tale argomento, incluse le pubblicazioni conservate per l'argomento sottoscritto.

In alternativa, un'applicazione può creare un ConnectionConsumer durevole associato a un nome specifico. Questo ConnectionConsumer riceve i messaggi che sono stati pubblicati sull'argomento dall'ultima volta che ConnectionConsumer è stato attivo. Riceve tutti questi messaggi che corrispondono al selettore sull'argomento. Tuttavia, se ConnectionConsumer sta utilizzando la lettura anticipata, può perdere i messaggi non persistenti che si trovano nel buffer del client quando viene chiuso.

Se WebSphere MQ classes per JMS si trova in modalità di migrazione del provider di messaggistica WebSphere MQ , viene utilizzata una coda separata per le sottoscrizioni ConnectionConsumer non durevoli.L'opzione configurabile CCSUB sul factory TopicConnectionspecifica la coda da utilizzare. Normalmente, CCSUB specifica una singola coda per l'utilizzo da parte di tutti i ConnectionConsumers che utilizzano lo stesso factory TopicConnection. Tuttavia, è possibile fare in modo che ogni ConnectionConsumer generi una coda temporanea specificando un prefisso del nome della coda seguito da un asterisco (*).

Se WebSphere MQ classes per JMS si trova in modalità di migrazione del provider di messaggistica WebSphere MQ , la proprietà CCDSUB dell'argomento specifica la coda da utilizzare per le sottoscrizioni durevoli. Di nuovo, questa può essere una coda già esistente o un prefisso del nome della coda seguito da un asterisco (*). Se si specifica una coda già esistente, tutti i ConnectionConsumers durevoli che sottoscrivono l'argomento utilizzano questa coda. Se si specifica un prefisso del nome della coda seguito da un asterisco (*), viene generata una coda la prima volta che un ConnectionConsumer durevole viene creato con un nome particolare. Questa coda viene riutilizzata successivamente quando viene creato un ConnectionConsumer durevole con lo stesso nome.

Quando si configura il gestore code WebSphere MQ , considerare i seguenti punti:

- Il gestore code deve avere una coda di messaggi non recapitabili abilitata. Se un ConnectionConsumer riscontra un problema quando inserisce un messaggio nella coda di messaggi non recapitabili, la consegna del messaggio dal QLOCAL sottostante si arresta. Per definire una coda di messaggi non recapitabili, utilizzare:

  ```
  ALTER QMGR DEADQ(your.dead.letter.queue.name)
  ```

- L'utente che esegue ConnectionConsumer deve disporre dell'autorità per eseguire MQOPEN con MQOO_SAVE_ALL_CONTEXT e MQOO_PASS_ALL_CONTEXT. Per i dettagli, consultare la documentazione WebSphere MQ per la propria piattaforma.
- È possibile ottimizzare le prestazioni per un singolo ConnectionConsumer creando una coda dedicata separata. Questo è al costo di un ulteriore utilizzo delle risorse.

Rimozione dei messaggi dalla coda in ASF

Quando un'applicazione utilizza ConnectionConsumers, è possibile che JMS debba rimuovere i messaggi dalla coda in diverse situazioni.

Queste situazioni sono le seguenti:

Messaggio formattato in modo non corretto
Potrebbe arrivare un messaggio che JMS non può analizzare.

Messaggio dannoso
Un messaggio potrebbe raggiungere la soglia di backout, ma ConnectionConsumer non riesce a riaccodare il messaggio sulla coda di backout.

Nessun ConnectionConsumer interessato
Per la messaggistica point - to - point, quando QueueConnectionFactory è impostato in modo da non conservare i messaggi indesiderati, arriva un messaggio indesiderato da parte di ConnectionConsumers.

In queste situazioni, ConnectionConsumer tenta di eliminare il messaggio dalla coda. Le opzioni di disposizione nel campo report di MQMD del messaggio impostano il comportamento esatto. Queste opzioni sono:

MQRO_DEAD_LETTER_Q
Il messaggio viene riaccodato alla coda di messaggi non instradabili del gestore code. Questa è l'opzione predefinita.

MQRO_DISCARD_MSG
Il messaggio viene eliminato.

Anche ConnectionConsumer genera un messaggio di report, che dipende anche dal campo del report MQMD del messaggio. Questo messaggio viene inviato alla coda ReplyTodel messaggio sul gestore code ReplyTo. Se si verifica un errore durante l'invio del messaggio di report, il messaggio viene inviato alla coda di messaggi non recapitabili. Le opzioni del report di eccezione nel campo del report dei dettagli della serie MQMD del messaggio del report. Queste opzioni sono:

MQRO_ECCEZIONE
Viene generato un messaggio di report che contiene l'MQMD del messaggio originale. Non contiene dati del corpo del messaggio.

MQRO_EXCEPTION_WITH_DATA
Viene generato un messaggio di report che contiene MQMD, tutte le intestazioni MQ e 100 byte di dati del corpo.

MQRO_EXCEPTION_WITH_FULL_DATA
Viene generato un messaggio di report che contiene tutti i dati del messaggio originale.

predefinito
Non viene generato alcun messaggio di prospetto.

Quando vengono generati messaggi di report, vengono rispettate le opzioni riportate di seguito:

- ID_MSG_NEW_MQRO
- MQRO_PASS_MSG_ID
- ID_COPY_MQRO_MSG_TO_CORREL_ID
- ID CORREL_PASS_MQRO_

Se un messaggio non elaborabile non può essere riaccodato, forse perché la coda di messaggi non recapitabili è piena o l'autorizzazione è specificata in modo non corretto, ciò che accade dipende dalla persistenza del messaggio. Se il messaggio non è persistente, viene eliminato e non viene generato alcun messaggio di report. Se il messaggio è persistente, la consegna dei messaggi a tutti i consumer di connessione in ascolto su tale destinazione si arresta. Tali utenti della connessione devono essere chiusi e il problema risolto prima che possano essere ricreati e la consegna del messaggio riavviata.

È importante definire una coda di messaggi non recapitabili e controllarla regolarmente per assicurarsi che non si verifichino problemi. In particolare, verificare che la coda di messaggi non recapitabili non raggiunga la profondità massima e che la dimensione massima del messaggio sia sufficiente per tutti i messaggi.

Quando un messaggio viene accodato alla coda di messaggi non recapitabili, è preceduto da un'intestazione di messaggi non recapitabili (MQDLH) WebSphere MQ . Consultare MQDLH - Dead

- letter header per i dettagli sul formato di MQDLH. È possibile identificare i messaggi che un ConnectionConsumer ha inserito nella coda di messaggi non recapitabili o segnalare i messaggi generati da un ConnectionConsumer dai seguenti campi:

- PutApplIl tipo è `MQAT_JAVA (0x1C)`
- PutApplIl nome è "`MQ JMS ConnectionConsumer`"

Questi campi si trovano nell'MQDLH dei messaggi nella coda di messaggi non instradabili e nell'MQMD dei messaggi di report. Il campo feedback di MQMD e il campo Motivo di MQDLH contengono un codice che descrive l'errore. Per dettagli su questi codici, consultare “Codici di errore e feedback in ASF” a pagina 933. Altri campi sono quelli descritti in MQDLH - Dead - letter header.

Gestione dei messaggi non elaborabili in ASF

All'interno delle funzioni del server delle applicazioni, la gestione dei messaggi non elaborabili viene gestita in modo leggermente diverso rispetto alle altre classi WebSphere MQ per JMS.

Per informazioni sulla gestione dei messaggi non elaborabili in WebSphere MQ classes for JMS, consultare “Gestione dei messaggi non elaborabili in IBM WebSphere MQ classes for JMS” a pagina 891.

Quando si utilizzano le ASF (Application Server Facilities), ConnectionConsumer, invece di MessageConsumer, elabora messaggi non elaborabili. Il ConnectionConsumer riaccoda i messaggi in base alle proprietà QName BackoutThreshold e BackoutRequeuedella coda.

Quando un'applicazione utilizza ConnectionConsumers, le circostanze in cui viene eseguito il backout di un messaggio dipendono dalla sessione fornita dal server delle applicazioni:

- Quando la sessione non è sottoposta a transazione, con AUTO_RICONOSCIUTO_AUTOMATICO o DUPS_OK_RICONOSCIMENTO, viene eseguito il backout di un messaggio solo dopo un errore di sistema o se l'applicazione termina in modo imprevisto.
- Quando la sessione non è sottoposta a transazioni con CLIENT_ACKNOWLEDGED, i messaggi non riconosciuti possono essere sottoposti a backout dal server delle applicazioni che richiama Session.recover().

  Di solito, l'implementazione client di MessageListener o il server delle applicazioni richiama Message.acknowledge(). Message.acknowledge() riconosce tutti i messaggi consegnati nella sessione fino a quel momento.
- Quando la sessione viene eseguita, i messaggi non riconosciuti possono essere ripristinati dal server delle applicazioni che richiama Session.rollback().
- Se il server delle applicazioni fornisce una XASession, i messaggi vengono sottoposti a commit o a backout in base a una transazione distribuita. Il server delle applicazioni si assume la responsabilità del completamento della transazione.

Il provider JMS incorporato in WebSphere Application Server, Versione 5.0 e Versione 5.1 gestisce i messaggi non elaborabili in un modo diverso da quello appena descritto per WebSphere MQ classes per JMS. Per informazioni sul modo in cui il provider JMS integrato gestisce i messaggi non elaborabili, consultare la relativa documentazione del prodotto WebSphere Application Server.

Gestione degli errori

Questa sezione descrive vari aspetti della gestione degli errori, inclusi “Ripristino da condizioni di errore in ASF” a pagina 932 e “Codici di errore e feedback in ASF” a pagina 933.

Ripristino da condizioni di errore in ASF

Se un ConnectionConsumer rileva un errore grave, la consegna del messaggio a tutti i ConnectionConsumers con un interesse nello stesso QLOCAL si arresta. Quando ciò si verifica, viene notificato qualsiasi ExceptionListener registrato con la connessione interessata. Esistono due modi in cui un'applicazione può eseguire il ripristino da queste condizioni di errore.

Di solito, un errore grave di questa natura si verifica se ConnectionConsumer non può riaccodare un messaggio alla coda di messaggi non recapitabili o se si verifica un errore durante la lettura dei messaggi da QLOCAL.

Poiché ogni ExceptionListener registrato con la connessione interessata viene notificato, è possibile utilizzarli per identificare la causa del problema. In alcuni casi, l'amministratore di sistema deve intervenire per risolvere il problema.

Utilizzare una delle tecniche riportate di seguito per eseguire il ripristino da queste condizioni di errore:

- Chiamare `close()` su tutti gli ConnectionConsumersinteressati. L'applicazione può creare nuovi ConnectionConsumers solo dopo che tutti i ConnectionConsumers interessati sono stati chiusi e gli eventuali problemi di sistema sono stati risolti.
- Richiamare `stop()` su tutte le connessioni interessate. Dopo che tutte le connessioni sono state arrestate e gli eventuali problemi di sistema sono stati risolti, l'applicazione può `start()` eseguire correttamente le connessioni.

Codici di errore e feedback in ASF

Utilizzare i codici di errore e di feedback per determinare la causa di un errore. Di seguito vengono forniti i codici di errore comuni generati da ConnectionConsumer .

Per determinare la causa di un errore, utilizzare le seguenti informazioni:

- Il codice di feedback nei messaggi di report
- Il codice di errore in MQDLH di tutti i messaggi nella coda di messaggi non instradabili

ConnectionConsumers genera i seguenti codici di errore.

MQRC_BACKOUT_THRESHOLD_REACHED (0x93A; 2362)

Causa
Il messaggio ha raggiunto la soglia di backout definita su QLOCAL, ma non viene definita alcuna coda di backout.

Sulle piattaforme in cui non è possibile definire la coda di backout, il messaggio ha raggiunto la soglia di backout definita da JMS di 20.

Azione
Se non si desidera, definire la coda di backout per il QLOCAL pertinente. Cercare anche la causa dei più backout.

MQRC_MSG_NOT_MATCHED (0x93B; 2363)

Causa
Nella messaggistica point - to - point, è presente un messaggio che non corrisponde ad alcuno dei selettori per ConnectionConsumers che controllano la coda. Per mantenere le prestazioni, il messaggio viene riaccodato alla coda di messaggi non recapitabili.

Azione
Per evitare questa situazione, assicurarsi che ConnectionConsumers che utilizza la coda fornisca una serie di selettori che trattano tutti i messaggi oppure impostare il factory QueueConnectionper conservare i messaggi.

In alternativa, esaminare l'origine del messaggio.

MQRC_JMS_FORMAT_ERROR (0x93C; 2364)

Causa
JMS non può interpretare il messaggio sulla coda.

Azione
Esaminare l'origine del messaggio. JMS normalmente consegna i messaggi di un formato non previsto come BytesMessage o TextMessage. Di tanto in tanto, questa operazione non riesce se il messaggio è formattato in modo non corretto.

Altri codici visualizzati in questi campi sono causati da un tentativo non riuscito di riaccodare il messaggio a una coda di backout. In questa situazione, il codice descrive il motivo per cui la riaccodamento non è riuscito. Per diagnosticare la causa di questi errori, fare riferimento a Codici di errore API.

Se il messaggio di report non può essere inserito nella coda ReplyTo, viene inserito nella coda di messaggi non recapitabili. In questa situazione, il campo di feedback di MQMD viene completato come descritto in questo argomento. Il campo motivo in MQDLH spiega perché non è stato possibile inserire il messaggio di report nella coda ReplyTo.

La funzione di un pool di sessioni server in AFS

Questo argomento riepiloga la funzione di un pool di sessioni server.

Figura 165 a pagina 935 riepiloga i principi della funzionalità ServerSessionPool e ServerSession .

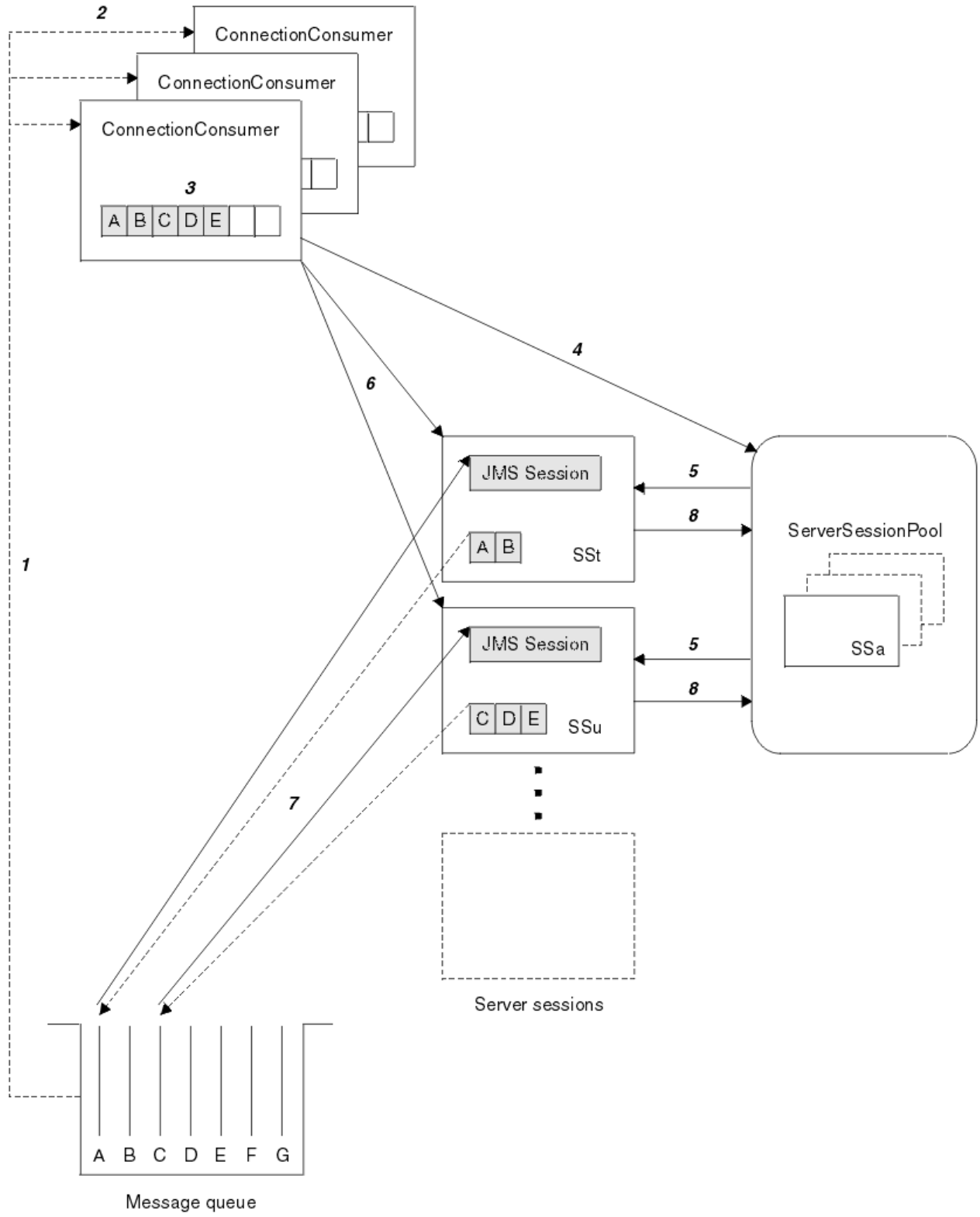


Figura 165. Funzionalità ServerSessionPool e ServerSession

1. I ConnectionConsumers ricevono i riferimenti ai messaggi dalla coda.
2. Ogni ConnectionConsumer seleziona riferimenti di messaggi specifici.
3. Il buffer ConnectionConsumer contiene i riferimenti ai messaggi selezionati.
4. ConnectionConsumer richiede una o più ServerSessions dal pool ServerSession.

5. Le ServerSessions vengono assegnate dal pool ServerSession.
6. ConnectionConsumer assegna i riferimenti dei messaggi a ServerSessions e avvia l'esecuzione dei thread ServerSession .
7. Ogni ServerSession richiama i relativi messaggi di riferimento dalla coda. Li passa al metodo onMessage dal MessageListener associato alla Sessione JMS.
8. Una volta completata l'elaborazione, ServerSession viene restituito al pool.

Un server delle applicazioni normalmente fornisce la funzione ServerSessionPool e ServerSession .

Utilizzo dello strumento di amministrazione JMS WebSphere MQ

Utilizzare lo strumento di amministrazione per definire le proprietà di otto tipi di oggetti WebSphere MQ classes per JMS e memorizzarli all'interno di uno spazio dei nomi JNDI. Le applicazioni possono quindi utilizzare JNDI per richiamare questi oggetti gestiti dallo spazio dei nomi.

Gli oggetti JMS delle classi WebSphere MQ che è possibile gestire utilizzando lo strumento sono:

- MQConnectionFactory
- Factory di MQQueueConnection
- Factory MQTopicConnection
- MQQUEUE
- MQArgomento
- MQXAConnectionFactory
- Factory MQXAQueueConnection
- Factory MQXATopicConnection

Per i dettagli su questi oggetti, vedere “Amministrazione di oggetti JMS” a pagina 941 .

I tipi di proprietà e i valori necessari per utilizzare questo strumento sono elencati in Proprietà degli oggetti IBM WebSphere MQ classes for JMS.

Lo strumento consente inoltre agli amministratori di modificare i contesti secondari dello spazio dei nomi della directory all'interno di JNDI. Consultare “Manipolazione dei contesti secondari con lo strumento di gestione JMS WebSphere MQ” a pagina 940.

È inoltre possibile creare e configurare oggetti amministrati JMS con WebSphere MQ Explorer.

Richiamo dello strumento di amministrazione IBM WebSphere MQ classes for JMS

Lo strumento di gestione ha un'interfaccia della riga comandi. È possibile utilizzarlo in modo interattivo o utilizzarlo per avviare un processo batch.

La modalità interattiva fornisce un prompt dei comandi in cui è possibile immettere i comandi di gestione. In modalità batch, il comando per avviare lo strumento include il nome di un file contenente uno script di comandi di amministrazione.

Modalità interattiva

Per avviare lo strumento in modalità interattiva, immettere il comando:

```
JMSAdmin [-t] [-v] [-cfg config_filename]
```

dove:

-t
Abilita traccia (il valore predefinito è traccia disattivata)

Il file di traccia viene generato in "%MQ_JAVA_DATA_PATH%"\errors (Windows) o /var/mqm/trace (UNIX). Il nome del file di traccia è nel formato:

mqjms_*PID*.trc

dove *PID* è l'ID processo della JVM.

-v
Produce output dettagliato (il valore predefinito è output conciso)

-cfg nomefile_config
Denomina un file di configurazione alternativo. Se questo parametro viene omesso, viene utilizzato il file di configurazione predefinito, JMSAdmin.config. (Consultare "Configurazione dello strumento di amministrazione JMS" a pagina 937)

Viene visualizzato un prompt dei comandi, che indica che lo strumento è pronto ad accettare i comandi di gestione. Questa richiesta inizialmente viene visualizzata come:

```
InitCtx>
```

che indica che il contesto corrente (ovvero, il contesto JNDI a cui fanno attualmente riferimento tutte le operazioni di denominazione e di directory) è il contesto iniziale definito nel parametro di configurazione PROVIDER_URL (consultare "Configurazione dello strumento di amministrazione JMS" a pagina 937).

Mentre si attraversa lo spazio dei nomi della directory, il prompt cambia per riflettere questo, in modo che il prompt visualizzi sempre il contesto corrente.

Modalità batch

Per avviare lo strumento in modalità batch, immettere il comando:

```
JMSAdmin <test.scp
```

dove *test.scp* Š un file script che contiene i comandi di gestione (consultare "Comandi di amministrazione nello strumento di gestione JMS di WebSphere MQ" a pagina 939). L'ultimo comando nel file deve essere il comando END .

Configurazione dello strumento di amministrazione JMS

Lo strumento di amministrazione JMS di WebSphere MQ utilizza un file di configurazione per impostare i valori di determinate proprietà. Viene fornito un file di esempio, che è possibile adattare al proprio sistema.

Il file di configurazione è un file di testo semplice costituito da una serie di coppie chiave - valore, separate dal segno uguale (=). Questo è mostrato nel seguente esempio:

```
#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
```

(Un # nella prima colonna della riga indica un commento o una riga non utilizzata.)

Un file di configurazione di esempio viene fornito con WebSphere MQ. Il file è denominato JMSAdmin.confige si trova nella directory <MQ_JAVA_INSTALL_PATH>/bin . Modificare questo file per adattarlo alla configurazione del proprio sistema.

Configurare lo strumento di amministrazione con i valori per le seguenti proprietà:

INITIAL_CONTEXT_FACTORY
Il service provider utilizzato dallo strumento. I valori supportati per questa proprietà sono i seguenti:

- com.sun.jndi.ldap.LdapCtxFactory (per LDAP)
- com.sun.jndi.fscontext.RefFSContextFactory (per il contesto del file system)

È anche possibile utilizzare un factory InitialContextnon presente nell'elenco precedente. Per ulteriori dettagli, vedere “Utilizzo di un factory InitialContextnon elencato con lo strumento di amministrazione JMS WebSphere MQ” a pagina 938.

PROVIDER_URL
L'URL del contesto iniziale della sessione; la root di tutte le operazioni JNDI eseguite dallo strumento. Sono supportate due forme di questa proprietà:

- ldap://nomehost/nomecontesto
- file: [unità:] /nomepercorso

Il formato dell'URL LDAP può variare, a seconda del provider LDAP. Per ulteriori informazioni, consultare la documentazione LDAP.

AUTENTICAZIONE_SICUREZZA
Indica se JNDI passa le credenziali di protezione al provider del servizio. Questa proprietà viene utilizzata solo quando viene utilizzato un fornitore di servizi LDAP. Questa proprietà può assumere uno dei tre seguenti valori:

- nessuno (autenticazione anonima)
- semplice (autenticazione semplice)
- CRAM-MD5 (meccanismo di autenticazioneCRAM-MD5)

Se non viene fornito un valore valido, la proprietà assume il valore predefinito none. Consultare “Configurazione della sicurezza per lo strumento di gestione JMS” a pagina 939 per ulteriori dettagli sulla sicurezza con lo strumento di gestione.

Queste proprietà sono impostate in un file di configurazione. Quando si richiama lo strumento, è possibile specificare questa configurazione utilizzando il parametro della riga comandi `-cfg` , come descritto in “Richiamo dello strumento di amministrazione IBM WebSphere MQ classes for JMS” a pagina 936. Se non si specifica un nome per il file di configurazione, lo strumento tenta di caricare il file di configurazione predefinito (`JMSAdmin.config`). Ricerca questo file prima nella directory corrente e poi nella directory `<MQ_JAVA_INSTALL_PATH>/bin` , dove `<MQ_JAVA_INSTALL_PATH>` è il percorso dell'installazione di WebSphere MQ per JMS.

Utilizzo di un factory InitialContextnon elencato con lo strumento di amministrazione JMS WebSphere MQ

Sono supportati due valori factory InitialContext. È possibile utilizzare altri contesti JNDI impostando i parametri nel file di configurazione di gestione JMS.

È possibile utilizzare lo strumento di gestione per connettersi a contesti JNDI diversi da quelli elencati in “Configurazione dello strumento di amministrazione JMS” a pagina 937 utilizzando tre parametri definiti nel file di configurazione JMSAdmin.

Per utilizzare un factory InitialContextdifferente:

1. Impostare la proprietà INITIAL_CONTEXT_FACTORY sul nome classe richiesto.
2. Definire il funzionamento del factory InitialContextutilizzando le proprietà USE_INITIAL_DIR_CONTEXT, NAME_PREFIX e NAME_READABILITY_MARKER.

Le impostazioni per queste proprietà sono descritte nei commenti del file di configurazione di esempio.

Non è necessario definire le tre proprietà qui elencate se si utilizza uno dei valori INITIAL_CONTEXT_FACTORY supportati. Tuttavia, è possibile fornire loro valori per sovrascrivere i valori predefiniti del sistema. Se si omettono una o più delle tre proprietà InitialContextFactory, lo strumento di amministrazione fornisce valori predefiniti appropriati in base ai valori delle altre proprietà.

Configurazione della sicurezza per lo strumento di gestione JMS

Utilizzare la proprietà SECURITY_AUTHENTICATION per determinare se le credenziali di sicurezza vengono inoltrate al fornitore del servizio.

La proprietà SECURITY_AUTHENTICATION è descritta in "Configurazione dello strumento di amministrazione JMS" a pagina 937. Il suo effetto è il seguente:

- Se si imposta questo parametro su none, JNDI non passa alcuna credenziale di sicurezza al provider di servizi e viene eseguita l' *autenticazione anonima* .
- Se si imposta il parametro su `semplice` o `CRAM-MD5`, le credenziali di sicurezza vengono trasmesse tramite JNDI al provider di servizi sottostante. Queste credenziali di sicurezza sono nel formato di un DN (distinguished name) utente e parola d'ordine.

Se le credenziali di sicurezza sono richieste, vengono richieste quando lo strumento viene inizializzato. Evitare ciò impostando le proprietà PROVIDER_USERDN e PROVIDER_PASSWORD nel file di configurazione JMSAdmin.

Nota: Se non si utilizzano queste proprietà, il testo immesso, *inclusa la password*, viene ripetuto sullo schermo. Ciò potrebbe avere implicazioni sulla sicurezza.

Lo strumento non esegue l'autenticazione; l'attività è delegata al server LDAP. L'amministratore del server LDAP deve impostare e gestire i privilegi di accesso a parti differenti della directory. Per ulteriori informazioni, consultare la documentazione LDAP. Se l'autenticazione ha esito negativo, lo strumento visualizza un messaggio di errore appropriato e termina.

Informazioni più dettagliate sulla sicurezza e JNDI sono disponibili nella documentazione sul sito Web Java di Sun (https://java.sun.com).

Comandi di amministrazione nello strumento di gestione JMS di WebSphere MQ

Lo strumento di amministrazione accetta comandi costituiti da un verbo di amministrazione e dai relativi parametri appropriati.

Quando viene visualizzato il prompt dei comandi, lo strumento è pronto ad accettare i comandi. I comandi di amministrazione sono generalmente del seguente formato:

```
verb [param]*
```

dove **verb** è uno dei verbi di amministrazione elencati in Tabella 133 a pagina 939. Tutti i comandi validi contengono un verbo, che viene visualizzato all'inizio del comando in formato standard o breve.

I parametri che un verbo può assumere dipendono dal verbo. Ad esempio, il comando END non può prendere alcun parametro, ma il comando DEFINE può prendere qualsiasi numero di parametri. I dettagli dei verbi che utilizzano almeno un parametro vengono discussi negli argomenti correlati.

Tabella 133. Verbi di amministrazione

Verbo	Forma breve	Descrizione
MODIFICA	Alt	Modificare almeno una delle proprietà di un oggetto gestito
Definisci	DEF	Creare e memorizzare un oggetto gestito o creare un contesto secondario
VISUALIZZA	DIS	Visualizza le proprietà di uno o più oggetti gestiti memorizzati o il contenuto del contesto corrente
ELIMINA	DEL	Rimuovere uno o più oggetti gestiti dallo spazio dei nomi o rimuovere un contesto secondario vuoto

Tabella 133. Verbi di amministrazione (Continua)		
Verbo	**Forma breve**	**Descrizione**
CHANGE	CHG	Modificare il contesto corrente, consentendo all'utente di attraversare lo spazio dei nomi della directory in un punto qualsiasi al di sotto del contesto iniziale (autorizzazione di sicurezza in sospeso)
Copia	CP	Creare una copia di un oggetto gestito memorizzato, memorizzandolo con un nome alternativo
SPOSTA	mV	Modificare il nome con cui è memorizzato un oggetto gestito
FINE		Chiudere lo Strumento di amministrazione

I nomi dei verbi non sono sensibili al maiuscolo / minuscolo.

Di solito, per terminare i comandi, premere il tasto di ritorno a capo. Tuttavia, è possibile sovrascriverlo immettendo il segno più (+) direttamente prima del ritorno a capo. Ciò consente di immettere comandi a più righe, come mostrato nel seguente esempio:

```
DEFINE Q(BookingsInputQueue) +
       QMGR(QM.POLARIS.TEST) +
       QUEUE(BOOKINGS.INPUT.QUEUE) +
       PORT(1415) +
       CCSID(437)
```

Le righe che iniziano con uno dei seguenti caratteri vengono considerate come commenti e vengono ignorate: * # /.

Manipolazione dei contesti secondari con lo strumento di gestione JMS WebSphere MQ

Utilizzare i comandi **CHANGE**, **DEFINE**, **DISPLAY** e **DELETE** per manipolare i contesti secondari dello spazio dei nomi della directory.

L'utilizzo di questi verbi è descritto in Tabella 134 a pagina 940.

Tabella 134. Sintassi e descrizione dei comandi utilizzati per manipolare i contesti secondari	
Sintassi dei comandi	**Descrizione**
DEFINE CTX (ctxName)	Tenta di creare un contesto secondario del contesto corrente, con il nome ctxName. Non riesce se si verifica una violazione della sicurezza, se il contesto secondario esiste già o se il nome fornito non è valido.
VISUALIZZA CTX	Visualizza il contenuto del contesto corrente. Gli oggetti gestiti sono annotati con a, i contesti secondari con `[D]`. Viene visualizzato anche il tipo Java di ciascun oggetto.
DELETE CTX (ctxName)	Tenta di eliminare il contesto child del contesto corrente con il nome ctxName. Non riesce se il contesto non viene trovato, non è vuoto o se si verifica una violazione della sicurezza.

Tabella 134. Sintassi e descrizione dei comandi utilizzati per manipolare i contesti secondari (Continua)	
Sintassi dei comandi	**Descrizione**
MODIFICA CTX (ctxName)	Modifica il contesto corrente, in modo che ora faccia riferimento al contesto child con il nome ctxName. È possibile fornire uno dei due valori speciali di ctxName : **= ATTIVO** passa al parent del contesto corrente **= INIT** passa direttamente al contesto iniziale Ha esito negativo se il contesto specificato non esiste o se si verifica una violazione della sicurezza.

Amministrazione di oggetti JMS

Questa sezione descrive gli otto tipi di oggetto che lo strumento di gestione può gestire. Include dettagli su ciascuna delle loro proprietà configurabili e sui verbi che possono manipolarle.

È inoltre possibile creare e configurare oggetti amministrati JMS con WebSphere MQ Explorer.

Tipi di oggetto JMS

La tabella mostra gli otto tipi di oggetti gestiti.

La colonna Parola chiave mostra le stringhe che è possibile sostituire per *TYPE* nei comandi mostrati in Tabella 136 a pagina 942.

Tabella 135. I tipi di oggetto JMS gestiti dallo strumento di gestione		
Tipo di oggetto	**Parola chiave**	**Descrizione**
MQConnectionFactory	CF	L'implementazione WebSphere MQ dell'interfaccia ConnectionFactory JMS. Questo rappresenta un oggetto factory per la creazione di connessioni in entrambi i domini point-to-point e di pubblicazione / sottoscrizione.
Factory di MQQueueConnection	QCF	L'implementazione WebSphere MQ dell'interfaccia JMS QueueConnectionFactory. Rappresenta un oggetto factory per la creazione di connessioni nel dominio point - to - point.
Factory MQTopicConnection	TCF	L'implementazione WebSphere MQ dell'interfaccia JMS TopicConnectionFactory. Rappresenta un oggetto factory per la creazione di connessioni nel dominio di pubblicazione / sottoscrizione.
MQQUEUE	Q	L'implementazione WebSphere MQ dell'interfaccia della coda JMS. Rappresenta una destinazione per i messaggi nel dominio point - to - point.

Tabella 135. I tipi di oggetto JMS gestiti dallo strumento di gestione (Continua)		
Tipo di oggetto	**Parola chiave**	**Descrizione**
MQArgomento	T	L'implementazione WebSphere MQ dell'interfaccia Argomento JMS. Rappresenta una destinazione per i messaggi nel dominio di pubblicazione / sottoscrizione.
MQXAConnectionFactory"1" a pagina 942	XACF	L'implementazione WebSphere MQ dell'interfaccia XAConnectionFactory JMS. Rappresenta un oggetto di produzione per la creazione di connessioni in entrambi i domini point-to-point e di pubblicazione / sottoscrizione e dove le connessioni utilizzano le versioni XA delle classi JMS.
MQXAQueueConnectionFactory"1" a pagina 942	XAQCF	L'implementazione WebSphere MQ dell'interfaccia JMS XAQueueConnectionFactory. Rappresenta un oggetto factory per la creazione di connessioni nel dominio point-to-point che utilizzano le versioni XA delle classi JMS.
MQXATopicConnectionFactory"1" a pagina 942	XATCF	L'implementazione WebSphere MQ dell'interfaccia JMS XATopicConnectionFactory. Rappresenta un oggetto factory per la creazione di connessioni nel dominio di pubblicazione / sottoscrizione che utilizzano le versioni XA delle classi JMS.
Nota: 1. Queste classi vengono fornite per essere utilizzate dai fornitori dei server delle applicazioni. È improbabile che siano direttamente utili per i programmatori di applicazioni.		

Verbi utilizzati con oggetti JMS

È possibile utilizzare i comandi `ALTER`, `DEFINE`, `DISPLAY`, `DELETE`, `COPY`e `MOVE` per manipolare gli oggetti amministrati nello spazio dei nomi della directory.

Tabella 136 a pagina 942 riepiloga l'utilizzo di questi verbi. Sostituire *TYPE* con la parola chiave che rappresenta l'oggetto amministrato richiesto, come elencato in Tabella 135 a pagina 941.

Tabella 136. Sintassi e descrizione dei comandi utilizzati per manipolare gli oggetti gestiti	
Sintassi dei comandi	**Descrizione**
ALTER *TYPE*(nome) [proprietà] *	Tenta di aggiornare le proprietà dell'oggetto gestito con quelle fornite. Ha esito negativo se si verifica una violazione della sicurezza, se non è possibile trovare l'oggetto specificato o se le nuove proprietà fornite non sono valide.
DEFINE *TYPE*(nome) [proprietà] *	Tenta di creare un oggetto gestito di tipo *TYPE* con le proprietà fornite e lo memorizza con il nome `name` nel contesto corrente. Non riesce se si verifica una violazione della sicurezza, se il nome fornito non è valido o se esiste un oggetto con tale nome o se le proprietà fornite non sono valide.

Tabella 136. Sintassi e descrizione dei comandi utilizzati per manipolare gli oggetti gestiti (Continua)	
Sintassi dei comandi	**Descrizione**
DISPLAY *TYPE*(nome)	Visualizza le proprietà dell'oggetto amministrato di tipo *TYPE*, associato al nome name nel contesto corrente. Non riesce se l'oggetto non esiste o se si verifica una violazione della sicurezza.
DELETE *TYPE*(nome)	Tenta di rimuovere l'oggetto gestito di tipo *TYPE*, con nome name, dal contesto corrente. Non riesce se l'oggetto non esiste o se si verifica una violazione della sicurezza.
COPY *TYPE*(nameA) *TYPE*(nameB)	Crea una copia dell'oggetto gestito di tipo *TYPE*, con il nome nameA, denominando la copia nameB. Tutto ciò si verifica nell'ambito del contesto corrente. Non riesce se l'oggetto da copiare non esiste, se esiste un oggetto denominato nameB o se si verifica una violazione della sicurezza.
MOVE *TYPE*(nameA) *TYPE*(nameB)	Sposta (ridenomina) l'oggetto gestito di tipo *TYPE*, con il nome nameA, in nameB. Tutto ciò si verifica nell'ambito del contesto corrente. Non riesce se l'oggetto da spostare non esiste, se esiste un oggetto con nome nameB o se si verifica una violazione della sicurezza.

Creazione di oggetti con lo strumento di gestione JMS WebSphere MQ

Creare gli oggetti e memorizzarli in uno spazio nomi JNDI utilizzando il comando DEFINE,

Utilizzare la sintassi del seguente comando:

```
DEFINE TYPE(name) [property]*
```

Vale a dire, il comando DEFINE , seguito da un *TYPE* (name) riferimento oggetto amministrato, seguito da zero o più *proprietà* (consultare Proprietà degli oggetti IBM WebSphere MQ classes for JMS).

Considerazioni sulla denominazione LDAP per oggetti JMS

Per memorizzare gli oggetti in un ambiente LDAP, è necessario fornire nomi conformi a determinate convenzioni. Lo strumento di amministrazione può aiutare a rispettare le convenzioni di denominazione aggiungendo un prefisso predefinito.

Una convenzione di denominazione è che i nomi degli oggetti e dei contesti secondari devono includere un prefisso, come cn= (nome comune) o ou= (unità organizzativa).

Lo strumento di amministrazione semplifica l'utilizzo dei provider di servizi LDAP consentendo di fare riferimento a nomi di oggetti e di contesto senza un prefisso. Se non si fornisce un prefisso, lo strumento aggiunge automaticamente un prefisso predefinito al nome fornito. Per LDAP, è cn=.

È possibile cambiare il prefisso predefinito impostando la proprietà NAME_PREFIX nel file di configurazione JMSAdmin, come descritto in “Utilizzo di un factory InitialContextnon elencato con lo strumento di amministrazione JMS WebSphere MQ” a pagina 938.

Questo è mostrato nel seguente esempio.

```
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
```

Sebbene il nome oggetto fornito (`testQueue`) non abbia un prefisso, lo strumento ne aggiunge automaticamente uno per garantire la conformità con la convenzione di denominazione LDAP. Allo stesso modo, l'inoltro del comando `DISPLAY Q(testQueue)` comporta anche l'aggiunta di questo prefisso.

Potrebbe essere necessario configurare il server LDAP per memorizzare gli oggetti Java. Per informazioni relative a questa configurazione, consultare la documentazione per il server LDAP.

Condizioni di errore di esempio durante la creazione di un oggetto JMS

Quando si crea un oggetto, possono verificarsi diverse condizioni di errore comuni.

I seguenti sono esempi di queste condizioni di errore:

CipherSpec associato a 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
```

Proprietà non valida per l'oggetto

```
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
```

Tipo non valido per il valore della proprietà

```
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
```

Conflitto di proprietà - il 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
```

Conflitto di proprietà - Inizializzazione uscita

```
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
```

Il valore della proprietà non è compreso nell'intervallo valido

```
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
```

Proprietà sconosciuta

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) PIZZA(ham and mushroom)
  Unable to create a valid object, please check the parameters supplied
  Unknown property: PIZZA
```

Di seguito sono riportati esempi di condizioni di errore che potrebbero verificarsi in Windows quando si ricercano oggetti gestiti da JNDI da un'applicazione JMS.

1. Se si utilizza il provider JNDI WebSphere , com.ibm.websphere.naming.WsnInitialContextFactory, è necessario utilizzare una barra (/) per accedere agli oggetti gestiti definiti nei contesti secondari; ad esempio, jms /MyQueueName. Se si utilizza una barra rovesciata (\), viene generata un'eccezione InvalidName.
2. Se si utilizza il provider JNDI Sun, com.sun.jndi.fscontext.RefFSContextFactory, è necessario utilizzare una barra retroversa (\) per accedere agli oggetti gestiti definiti nei contesti secondari; ad esempio, ctx1\\fred. Se si utilizza una barra (/), viene generata unaFoundException NameNot.

Utilizzo di WebSphere MQ Explorer per la configurazione JMS

Utilizzare la GUI (graphical user interface) di IBM WebSphere MQ Explorer per creare oggetti JMS da oggetti WebSphere MQ e oggetti WebSphere MQ da oggetti JMS, nonché per gestire e monitorare altri oggetti WebSphere MQ .

Prima di iniziare

Prima di creare e configurare gli oggetti amministrati JMS con Esplora risorse di WebSphere MQ , aggiungere un contesto iniziale per definire la root dello spazio nomi JNDI in cui gli oggetti JMS sono memorizzati nel servizio di denominazione e di directory. Per ulteriori informazioni, fare riferimento all'assistenza utente di IBM WebSphere MQ Explorer per gli oggetti amministrati JMS.

Informazioni su questa attività

È possibile eseguire le seguenti attività con Esplora risorse di IBM WebSphere MQ , contestualmente da un oggetto esistente in Esplora risorse di IBM WebSphere MQ o dall'interno di una procedura guidata di creazione di un nuovo oggetto. Fare riferimento alla guida di WebSphere MQ Explorer per esempi di assistenza utente WebSphere MQ Explorer per alcune attività tipiche.

Procedura

- Creare un factory di connessione JMS da uno dei seguenti oggetti WebSphere MQ :
  a) Un gestore code WebSphere MQ , sia sul computer locale che su un sistema remoto.
  b) Un canale WebSphere MQ
  c) Un listener WebSphere MQ
- Aggiungere un gestore code WebSphere MQ a WebSphere MQ Explorer utilizzando un factory di connessione JMS
- Creare una coda JMS da una coda WebSphere MQ
- Creare una coda WebSphere MQ da una coda JMS
- Creare un argomento JMS da un argomento WebSphere MQ , che può essere un oggetto WebSphere MQ o un argomento dinamico
- Creare un argomento WebSphere MQ da un argomento JMS

Utilizzo del pacchetto WebSphere MQ Headers

Il pacchetto di intestazioni WebSphere MQ fornisce una serie di interfacce e classi helper che è possibile utilizzare per manipolare le intestazioni WebSphere MQ di un messaggio. Generalmente, si utilizza il pacchetto WebSphere MQ Headers perché si desidera eseguire i servizi di gestione utilizzando il server dei comandi (utilizzando i messaggi PCF (Programmable Command Format)).

Informazioni su questa attività

Il pacchetto WebSphere MQ Headers si trova nei pacchetti `com.ibm.mq.headers` e `com.ibm.mq.pcf` . È possibile utilizzare questa funzionalità per entrambe le API alternative che WebSphere MQ fornisce per l'utilizzo nelle applicazioni Java:

- WebSphere MQ classes per Java (noto anche come WebSphere MQ Headers Base Java).
- WebSphere MQ classes for Java Message Service (classi WebSphere MQ per JMS, denominate anche WebSphere MQ JMS).

WebSphere MQ Le applicazioni Java di base generalmente manipolano oggetti MQMessage e le classi di supporto delle intestazioni possono interagire direttamente con questi oggetti, poiché comprendono nativamente le interfacce Java di base WebSphere MQ .

In WebSphere MQ JMS, il payload per un messaggio è in genere una stringa o un oggetto array di byte, che può essere manipolato con i flussi DataInput e DataOutput . Il pacchetto di intestazioni WebSphere

MQ può essere utilizzato per interagire con questi flussi di dati ed è adatto per la manipolazione di tutti i messaggi MQ inviati e ricevuti dalle applicazioni JMS WebSphere MQ .

Pertanto, sebbene il pacchetto WebSphere MQ Headers contenga riferimenti al package WebSphere MQ Base Java, è anche progettato per essere utilizzato all'interno delle applicazioni WebSphere MQ JMS ed è adatto per l'utilizzo in ambienti Java Platform, Enterprise Edition (Java EE).

Un modo tipico in cui è possibile utilizzare il package WebSphere MQ Headers consiste nel modificare i messaggi di gestione in PCF (Programmable Command Format), ad esempio per uno dei seguenti motivi:

- Per accedere ai dettagli relativi a una risorsa WebSphere MQ .
- Per monitorare la profondità di una coda.
- Per impedire l'accesso a una coda.

Utilizzando i messaggi PCF con l'API JMS WebSphere MQ , questo tipo di gestione delle risorse incentrate sull'applicazione può essere eseguita dall'interno delle applicazioni Java EE senza dover ricorrere all'API Java di base WebSphere MQ .

Procedura

- Per utilizzare il pacchetto WebSphere MQ Headers per manipolare le intestazioni dei messaggi per le classi WebSphere MQ per Java, consultare “Utilizzo con classi WebSphere MQ per Java” a pagina 946.
- Per utilizzare il pacchetto WebSphere MQ Headers per manipolare le intestazioni dei messaggi per JMS, consultare “Utilizzo con le classi WebSphere MQ per JMS” a pagina 947.

Utilizzo con classi WebSphere MQ per Java

Le classi WebSphere MQ per le applicazioni Java in genere manipolano oggetti MQMessage e le classi di supporto delle intestazioni possono interagire direttamente con tali oggetti, poiché comprendono nativamente le classi WebSphere MQ per le interfacce Java.

Informazioni su questa attività

WebSphere MQ fornisce alcune applicazioni di esempio che dimostrano come utilizzare il package WebSphere MQ Headers con l'API Java di base WebSphere MQ (WebSphere MQ classes per Java).

Gli esempi mostrano due cose:

- Come creare un messaggio PCF per eseguire un'azione di gestione ed analizzare il messaggio di risposta.
- Come inviare questo messaggio PCF utilizzando le classi WebSphere MQ per Java.

In base alla piattaforma utilizzata, tali esempi vengono installati nella directory `pcf` nella directory `samples` o `tools` dell'installazione di WebSphere MQ (consultare “Directory di installazione per classi WebSphere MQ per Java” a pagina 658).

Procedura

1. Creare un messaggio PCF per eseguire un'azione di gestione e analizzare il messaggio di risposta.
2. Inviare questo messaggio PCF utilizzando le classi WebSphere MQ per Java.

Concetti correlati

“Gestione delle intestazioni dei messaggi WebSphere MQ con classi WebSphere MQ per Java” a pagina 679
Vengono fornite classi Java che rappresentano diversi tipi di intestazione del messaggio. Sono fornite anche due classi helper.

“Gestione dei messaggi PCF con classi WebSphere MQ per Java” a pagina 685
Le classi Java sono fornite per creare e analizzare messaggi strutturati PCF e per facilitare l'invio di richieste PCF e la raccolta di risposte PCF.

Utilizzo con le classi WebSphere MQ per JMS

Per utilizzare le intestazioni WebSphere MQ con le classi WebSphere MQ per JMS, è necessario eseguire le stesse operazioni essenziali delle classi WebSphere MQ per Java. È possibile creare il messaggio PCF e analizzare la risposta esattamente nello stesso modo utilizzando il pacchetto WebSphere MQ Headers e lo stesso codice di esempio delle classi WebSphere MQ per Java.

Informazioni su questa attività

Per inviare un messaggio PCF utilizzando l'API WebSphere MQ , il payload del messaggio deve essere scritto in un messaggio di byte JMS e inviato utilizzando le API JMS standard. L'unica considerazione è che il messaggio non deve contenere un JMS RFH2 o qualsiasi altra intestazione con valori specifici in MQMD.

Per inviare un messaggio PCF, completare le seguenti operazioni. Il modo in cui viene creato il messaggio PCF e le informazioni vengono estratte dal messaggio di risposta è lo stesso utilizzato per le classi WebSphere MQ per Java (consultare “Utilizzo con classi WebSphere MQ per Java” a pagina 946).

Procedura

1. Creare una destinazione coda JMS che rappresenti il SISTEMA SYSTEM.ADMIN.COMMAND.QUEUE.

   WebSphere MQ Le applicazioni JMS inviano i messaggi PCF a SYSTEM.ADMIN.COMMAND.QUEUEe devono accedere a un oggetto di destinazione JMS che rappresenta questa coda. La destinazione deve avere le seguenti proprietà impostati:

   ```
   WMQ_MQMD_WRITE_ENABLED = YES
   WMQ_MESSAGE_BODY = MQ
   ```

   Se si utilizza WebSphere Application Server, è necessario definire queste proprietà come proprietà personalizzate sulla destinazione.

   Per creare la destinazione in modo programmatico dall'interno di un'applicazione, utilizzare il codice seguente:

   ```
   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. Convertire un messaggio PCF in un messaggio di byte JMS contenente i valori MQMD corretti.

   È necessaria la creazione di un messaggio di byte JMS e la scrittura del messaggio PCF. È necessario creare una coda di risposta, ma questa non deve avere impostazioni specifiche.

   Il frammento di codice di esempio riportato di seguito mostra come creare un messaggio di byte JMS e scrivere un oggetto com.ibm.mq.headers,pcf.PCFMessage . L'oggetto PCFMessage (pcfCmd) è stato precedentemente creato utilizzando il package di intestazioni WebSphere MQ . (Si noti che il pacchetto per caricare PCFMessage è 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. Inviare il messaggio e ricevere la risposta utilizzando le API JMS standard.
4. Convertire il messaggio di risposta in un messaggio PCF per l'elaborazione.

   Per richiamare il messaggio di risposta ed elaborarlo come messaggio PCF, utilizzare il codice seguente:

```
 // 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);
```

Concetti correlati

“Messaggi JMS” a pagina 809
I messaggi JMS sono composti da un'intestazione, proprietà e un corpo. JMS definisce cinque tipi di corpo del messaggio.

Utilizzo dei servizi Web in WebSphere MQ

È possibile sviluppare applicazioni IBM WebSphere MQ per servizi Web utilizzando il trasporto IBM WebSphere MQ per SOAP o il bridge IBM WebSphere MQ per HTTP.

Il trasporto IBM WebSphere MQ per SOAP fornisce un trasporto JMS per SOAP. Il trasporto IBM WebSphere MQ per SOAP è inoltre integrato in altri ambienti come Microsoft Windows Communication Foundation, WebSphere Application Server e CICS Transaction Server.

Per ulteriori informazioni sul trasporto IBM WebSphere MQ per SOAP, consultare “Trasporto WebSphere MQ per SOAP” a pagina 949.

Con il bridge IBM WebSphere MQ per HTTP, le applicazioni client possono scambiare messaggi con IBM WebSphere MQ senza dover installare un client WebSphere MQ MQI. È possibile richiamare WebSphere MQ da qualsiasi piattaforma o lingua con funzionalità HTTP.

Per ulteriori informazioni sul bridge IBM WebSphere MQ per HTTP, consultare “WebSphere MQ Bridge per HTTP” a pagina 1025.

Concetti correlati

“Concetti dello sviluppo di applicazioni” a pagina 8
È possibile utilizzare una scelta di linguaggi procedurali o orientati agli oggetti per scrivere applicazioni IBM WebSphere MQ . Utilizzare i link in questo argomento per informazioni sui concetti di IBM WebSphere MQ che sono utili per sviluppatori di applicazioni.

“Scelta del linguaggio di programmazione da utilizzare” a pagina 79
Utilizza queste informazioni per scoprire i linguaggi di programmazione e i framework supportati da IBM WebSphere MQe alcune considerazioni per utilizzarli.

“Progettazione di applicazioni IBM WebSphere MQ” a pagina 90
Una volta stabilito il modo in cui le applicazioni possono sfruttare le piattaforme e gli ambienti disponibili, è necessario decidere come utilizzare le funzioni offerte da WebSphere MQ.

“Programmi di WebSphere MQ di esempio” a pagina 97

Utilizzare questa raccolta di argomenti per informazioni sui programmi WebSphere MQ di esempio su piattaforme differenti.

“Scrittura di un'applicazione di accodamento” a pagina 195
Utilizzare queste informazioni per informazioni sulla scrittura delle applicazioni di accodamento, la connessione e la disconnessione da un gestore code, la pubblicazione / sottoscrizione e l'apertura e la chiusura di oggetti.

“Scrittura delle applicazioni client” a pagina 354
Informazioni necessarie per scrivere le applicazioni client su WebSphere MQ.

“Scrittura di applicazioni di pubblicazione / sottoscrizione” a pagina 279
Iniziare a scrivere applicazioni WebSphere MQ di pubblicazione / sottoscrizione.

“Creazione di un'applicazione IBM WebSphere MQ” a pagina 431
Utilizzare queste informazioni per informazioni sulla generazione di un'applicazione IBM WebSphere MQ su piattaforme differenti.

“Gestione degli errori del programma” a pagina 550
Queste informazioni spiegano gli errori associati alle chiamate MQI delle applicazioni quando effettua una chiamata o quando il suo messaggio viene consegnato alla destinazione finale.

Trasporto WebSphere MQ per SOAP

Il trasporto WebSphere MQ per SOAP fornisce un trasporto JMS per SOAP. Il trasporto WebSphere MQ per SOAP è integrato anche in altri ambienti come Microsoft Windows Communication Foundation, WebSphere Application Server e CICS Transaction Server.

Introduzione al trasporto IBM WebSphere MQ per SOAP

Il trasporto IBM WebSphere MQ per SOAP fornisce un trasporto JMS per SOAP. Il listener e il mittente SOAP WebSphere MQ forniscono un mezzo per richiamare i servizi Web.

Il listener WebSphere MQ SOAP supporta i servizi ospitati da .NET Framework 1, .NET Framework 2 e Axis 1.4. Il mittente SOAP di WebSphere MQ supporta i client dei servizi Web in esecuzione su .NET Framework 1, .NET Framework 2, Axis 1.4 e Axis2. I client possono essere un server WebSphere MQ o un'applicazione client. Il trasporto IBM WebSphere MQ per SOAP è integrato anche in altri ambienti quali Microsoft Windows Communication Foundation, WebSphere Application Server e CICS Transaction Server.

L'integrazione in Microsoft Windows Communication Foundation fa parte del supporto IBM WebSphere MQ per .NET Framework 3.

Il trasporto IBM WebSphere MQ per SOAP è una serie di protocolli e strumenti per trasportare i messaggi SOAP utilizzando JMS su IBM WebSphere MQ. Viene fornito in diversi modi per ambienti di applicazioni differenti, come mostrato in Tabella 137 a pagina 949.

Tabella 137. Trasporto IBM WebSphere MQ per ambienti di applicazioni SOAP

	Integrato con componenti aggiuntivi di WebSphere MQ	**Integrato in un contesto**
Fornito come parte dell'installazione di WebSphere MQ	.NET Framework 1 .NET Framework 2 asse 1.4	Windows Communication Foundation (.NET Framework 3) Axis2 (solo client)
Fornito in un altro pacchetto software		WebSphere Application Server CICS Transaction Server 4.1 WebSphere ESB WebSphere Process Server for Multiplatforms

L'integrazione del trasporto IBM WebSphere MQ per SOAP in un framework dell'applicazione semplifica lo sviluppo e la distribuzione dei servizi Web in IBM WebSphere MQ.

Con ulteriori componenti IBM WebSphere MQ SOAP, è possibile interagire direttamente con i componenti SOAP WebSphere MQ per sviluppare e distribuire i servizi. Utilizzare gli strumenti IBM WebSphere MQ SOAP per configurare e distribuire i servizi Web e i client dei servizi Web in IBM WebSphere MQ.

Negli ambienti integrati, lo sviluppo e la distribuzione sono più semplici. Per lo sviluppo e la distribuzione si utilizzano gli stessi strumenti che si utilizzano per sviluppare e distribuire un servizio Web HTTP SOAP. È comunque necessario configurare le code, i canali e i gestori code IBM WebSphere MQ richiesti utilizzando gli strumenti WebSphere MQ .

È possibile combinare e mettere in corrispondenza i client e i server IBM WebSphere MQ SOAP da uno qualsiasi di questi ambienti.

Vantaggi

Il trasporto WebSphere MQ per SOAP offre agli utenti IBM WebSphere MQ esistenti i seguenti vantaggi principali:

Utilizzo della rete IBM WebSphere MQ per collegare i servizi Web esistenti.

I servizi potrebbero essere quelli scritti dall'utente o i servizi forniti come interfacce per altre applicazioni software in pacchetto distribuite dall'utente.

Il vantaggio deriva dall'utilizzo della rete WebSphere MQ esistente per collegare i servizi Web. Il trasporto IBM WebSphere MQ ha il vantaggio di essere un servizio di messaggistica in coda gestito e affidabile.

Scrittura di nuove applicazioni o conversione di applicazioni esistenti per utilizzare le interfacce SOAP piuttosto che IBM WebSphere MQ .

In genere, le applicazioni richiedono lo sviluppo di un adattatore WebSphere MQ specifico da integrare con un'altra applicazione. Gli adattatori hanno due parti: la parte del connettore, che inserisce e riceve i messaggi da e verso il trasporto, e la parte dell'adattatore che converte i dati in e da formati specifici dell'applicazione. L'integrazione di ogni coppia di applicazioni è una nuova sfida.

Il vantaggio di SOAP deriva dalla standardizzazione su SOAP per la definizione delle interfacce dell'applicazione e quindi dalla scelta di trasporti. Non è necessario scrivere adattatori specifici dell'applicazione ed è possibile scegliere se utilizzare IBM WebSphere MQ o HTTP come connettore. Il trasporto scelto dipende dalla qualità del servizio e dalla connettività richiesti.

Per gli utenti SOAP su HTTP esistenti, il vantaggio del trasporto WebSphere MQ per SOAP deriva dall'uso di un trasporto asincrono gestito e affidabile. I vantaggi sono due:

Un modello di programmazione veramente asincrono per disponibilità e prestazioni.
Utilizzando un'interfaccia client asincrona, le applicazioni client e di servizio non devono essere disponibili contemporaneamente. Le richieste inviate dal client verranno memorizzate fino a quando il servizio non sarà disponibile per elaborarle.

Una rete gestita già pronta, progettata per essere affidabile e disponibile.

Scegliendo IBM WebSphere MQ come trasporto, si ottiene il vantaggio di utilizzare una rete gestita che fornisce una messaggistica affidabile.

Al contrario, i trasporti come HTTP e FTP su TCP/IP non sono gestiti. Una rete non gestita è ideale per connessioni imprevedibili: ci sono meno attività di gestione.

Riepilogo

Il trasporto IBM WebSphere MQ per SOAP fornisce i seguenti componenti:

- Il bind di trasporto SOAP/JMS viene utilizzato nei documenti WSDL per collegare un servizio SOAP a un trasporto JMS. L'implementazione WebSphere MQ del bind SOAP/JMS utilizza un URI che assume una delle due forme:

Trasporto WebSphere MQ per SOAP

```
jms:/queue?&Name=Value&Name=Value...
```

Raccomandazione del candidato WebSphere MQ wire format per W3C

```
jms:queue:qName?connectionFactory=connectQueueManager(qMgrName)&Name=Value&Name=Value...
```

- L'associazione di un messaggio SOAP su un messaggio WebSphere MQ .
- Due listener IBM WebSphere MQ SOAP per ricevere richieste SOAP, uno per Java e uno per .NET Framework 1 o .NET Framework 2. I listener utilizzano .NET o Axis 1.4 per elaborare la richiesta SOAP.
- Due IBM WebSphere MQ mittenti SOAP per creare richieste SOAP IBM WebSphere MQ . I clienti dei servizi Web si registrano con un mittente per elaborare richieste `jms:` SOAP.
- Integrazione con WCF (Windows Communication Foundation), a volte noto come .NET 3, per inviare e ricevere WebSphere MQ Transport for SOAP messages.
- Integrazione del client con Axis2, a volte noto come JAX - WS, per inviare messaggi SOAP JMS WebSphere MQ Transport for SOAP o W3C .
- Il comando **amqwdeployWMQService**, che crea componenti di sviluppo e runtime e script per distribuire un servizio Web utilizzando il trasporto IBM WebSphere MQ per SOAP.
- Codice di servizio e client Java e .NET di esempio.
- Uno script per impostare il percorso della classe e altri script del programma di utilità.

Negli ambienti integrati il mittente e il listener sono integrati in ciascun ambiente, così come le estensioni agli strumenti di distribuzione e sviluppo.

Integrazione di SOAP e WebSphere MQ

Il trasporto WebSphere MQ per SOAP estende SOAP e gli strumenti dei servizi Web e il runtime, con WebSphere MQ come trasporto alternativo a HTTP per SOAP. Non è necessario modificare i servizi Web esistenti per utilizzare il trasporto WebSphere MQ per SOAP come trasporto. Il trasporto utilizza un formato URI personalizzato per SOAP/JMS. Il formato URI W3C per SOAP/JMS è supportato in modo limitato dai client Axis2 .

È necessario aggiungere una riga di codice aggiuntiva ai client negli ambienti .NET Framework 1, .NET Framework 2 e Axis 1.4 . Non è richiesto alcun codice aggiuntivo nei client Axis 2 e Windows Communication Foundation (WCF). Il listener WebSphere MQ SOAP esegue i servizi negli ambienti .NET Framework 1, .NET Framework 2 e Axis 1.4 . Il trasporto WebSphere MQ per SOAP è integrato in altri ambienti di server delle applicazioni, inclusi WCF, CICS e WebSphere Application Server.

Descrizione di SOAP

SOAP[9]descrive il formato standardizzato dei messaggi e dei protocolli di interazione che le applicazioni utilizzano per scambiare richieste, risposte e datagrammi. SOAP è indipendente dal trasporto utilizzato per trasferire i messaggi e dall'ambiente dell'applicazione che invia e riceve i messaggi. W3C definisce SOAP Versione 1.2 in modo succinto:

SOAP Versione 1.2 fornisce la definizione delle informazioni basate su XML che possono essere utilizzate per lo scambio di informazioni strutturate e immesse tra peer in un ambiente decentralizzato e distribuito.[10].

Per utilizzare SOAP deve essere collegato a un trasporto, ad esempio HTTP, e-mail o WebSphere MQ.

Un framework di bind del protocollo SOAP è la serie di regole per il trasporto di un messaggio SOAP su un altro protocollo, ad esempio HTTP. Versione SOAP 1.2 Parte 2: Adjuncts (Seconda edizione) descrive il bind HTTP SOAP.

[9] Storicamente, l'acronimo era Simple Object Access Protocol.
[10] W3C: SOAP Version 1.2 Parte 0

Il suggerimento candidato W3C , 4 giugno 2009, SOAP su Java Message Service 1.0, descrive il suggerimento per il bind JMS SOAP. Poiché JMS è una specifica API e non un protocollo di trasporto, il suggerimento SOAP JMS non descrive il formato di collegamento dei messaggi JMS SOAP. Descrive i protocolli di interazione SOAP e il bind API JMS. Di conseguenza, quando si utilizza la raccomandazione SOAP JMS, è necessario utilizzare ancora la stessa implementazione JMS per il client SOAP e il server SOAP. Abilita l'esecuzione di un'applicazione JMS SOAP su qualsiasi implementazione di JMS. Un'implementazione JMS può essere inserita in un server di applicazioni J2EE , se sia il server che l'implementazione JMS sono conformi alla specifica JCA. WebSphere MQ JMS è conforme alla specifica JCA e può essere collegato a un server delle applicazioni compatibile.

Il trasporto WebSphere MQ per il bind SOAP è come lo standard W3C proposto, ma non è lo stesso. Il suo uso è descritto nell'argomento MQRFH2 SOAP. Diversamente dalla raccomandazione del candidato W3C , il bind SOAP non viene specificato formalmente. Di fatto, si tratta del binding HTTP e l'indirizzo di servizio assume il formato `jms:/queue?name=value&name=value...`, piuttosto che `http://authority/path?query#fragment`. `jms:` non è uno schema IANA URI ufficialmente registrato.

Descrizione di un servizio Web

SOAP consente ai programmi scritti in linguaggi differenti, in esecuzione su piattaforme differenti, di comunicare utilizzando vari protocolli di trasporto. SOAP è la specifica del protocollo. Un servizio Web è un'applicazione che fornisce un servizio tramite un'interfaccia SOAP a cui è possibile accedere utilizzando protocolli Internet.

Un obiettivo importante di SOAP è fornire servizi che i client possono utilizzare facilmente. Una volta progettato un client per utilizzare un servizio, è possibile programmare la chiamata per richiamare il servizio senza riferimento alla documentazione esterna. Le interfacce del servizio sono descritte in XML, in un documento WSDL. La query, `http://authority/path?wsdl`, restituisce la descrizione WSDL di un servizio SOAP.

Suggerimento: Quando si distribuisce un servizio Web per utilizzare WebSphere MQ, distribuire anche il servizio su HTTP in modo che la query WSDL standard funzioni.

Sviluppo di servizi Web

I servizi Web hanno un cliente e una parte di servizio. Il servizio viene scritto per primo, iniziando dalla descrizione dell'interfaccia in WSDL o seguendo le regole per la scrittura della classe di servizio. I toolkit dei servizi Web dispongono di programmi di utilità per generare WSDL dalla definizione di interfaccia di una classe; ad esempio, **`java2wsdl`** o **`disco`**. Dispongono anche di strumenti per generare o classificare le strutture dalle descrizioni dell'interfaccia WSDL; ad esempio **`wsdl2java`**, **`wsimport`**o **`wsdl`**. Il primo è conosciuto come sviluppo dal basso verso l'alto, e il secondo dall'alto verso il basso.

Il comando **`amqwdeployWMQService`** nel trasporto WebSphere MQ per SOAP utilizza questi strumenti per generare WSDL, stub client e proxy client.

I servizi Web sono generalmente scritti utilizzando un ambiente di sviluppo integrato destinato a un ambiente server delle applicazioni particolare:

Eclipse IDE per Java EE Developers
Crea servizi Web per Axis 2. Supporta JAX - RPC e JAX - WS

Rational Application Developer V7.5
Crea servizi Web per WebSphere Application Server V7 e versioni precedenti e anche per Axis. Supporta JAX - RPC e JAX - WS.

WebSphere Integration Developer V6.2
Crea i servizi web per WebSphere Process Server e WebSphere ESB. Supporta JAX - RPC e JAX - WS.

Visual Studio 2008 (Versione 9)
Crea servizi Web per .NET Framework 3.5 e versioni precedenti (Windows Communication Foundation)

Visual Studio 2005 (Versione 8)
Crea servizi Web per .NET Framework 2 e versioni precedenti

È possibile utilizzare uno qualsiasi di questi strumenti in combinazione con il trasporto WebSphere MQ per SOAP. Dopo aver sviluppato un servizio da utilizzare con HTTP, utilizzare lo strumento **amqwdeployWMQService** per distribuire i servizi per utilizzare WebSphere MQ come trasporto. È possibile scrivere un nuovo client utilizzando l'output dello strumento oppure modificare i client esistenti per utilizzare il trasporto WebSphere MQ per SOAP.

Se il trasporto WebSphere MQ per SOAP è integrato nell'ambiente dell'applicazione, non è necessario utilizzare lo strumento **amqwdeployWMQService** o modificare il codice client. Il livello SOAP del client indirizza le richieste client che hanno un URI con il prefisso `jms:` al trasporto WebSphere MQ per SOAP. Il livello SOAP del server richiama il trasporto WebSphere MQ per SOAP per attendere le richieste SOAP `jms:` e restituisce risposte al trasporto WebSphere MQ per SOAP.

Generalmente, i servizi .NET sono stati sviluppati dal basso verso l'alto utilizzando le annotazioni del servizio Web nel codice e i servizi Java dall'alto verso il basso, utilizzando le definizioni dell'interfaccia WSDL. La differenza negli approcci si sta riducendo, poiché Java Standard Edition Versione 6 supporta JAX - WS 2.0e utilizza le annotazioni per qualificare la definizione delle interfacce del servizio. Ora è facile sviluppare i servizi Java dal basso verso l'alto come dall'alto verso il basso. Quale approccio si sceglie è una questione di metodo di sviluppo.

Il client dei servizi Web viene scritto dopo il servizio, utilizzando la definizione del servizio WSDL e i proxy e gli stub client generati. In alcune applicazioni, la definizione servizio non è nota quando il client viene scritto. Il client richiama il WSDL del servizio e crea le richieste di servizio in modo dinamico. Più comunemente la definizione del servizio è nota, ma l'indirizzo a cui viene distribuito il servizio non lo è. Il toolkit del servizio Web genera interfacce che il client può utilizzare per effettuare richieste di servizio. Il client fornisce l'indirizzo di servizio quando è richiesto. Nel terzo caso, il WSDL contiene tutte le informazioni necessarie al client. WSDL contiene l'interfaccia e l'indirizzo del servizio. Il codice generato dal toolkit del servizio Web contiene tutte le informazioni necessarie al client per effettuare richieste di un servizio.

È possibile utilizzare uno qualsiasi di questi tre stili con il trasporto WebSphere MQ per SOAP.

Ambienti di applicazioni di servizi web

I toolkit del servizio Web richiedono un'associazione dalla definizione WSDL di un servizio ai flussi di byte trasferiti nelle richieste e risposte SOAP. Il flusso di byte è definito dalla specifica SOAP ed è contenuto nella busta SOAP. La busta SOAP viene visualizzata in Figura 166 a pagina 953.

```
<?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>
```

Figura 166. Busta SOAP

L'associazione dall'envelope SOAP al bind del linguaggio e viceversa è parte standardizzata e parte proprietaria. L'associazione è fondamentale per l'architettura .NET e viene fornita come parte di CLR (Common Language Runtime). L'associazione è standardizzata in Java dalle specifiche JAX. Poiché le mappature Java sono standardizzate, i servizi e i client del servizio Web Java sono portabili tra diversi ambienti di applicazioni basati su Java. JAX - RPC (a volte chiamato JAX - WS 1.0) è la mappatura più utilizzata oggi. È supportato da Axis 1.4. JAX - WS (a volte chiamato JAX - WS 2.0) è uno standard notevolmente migliorato ed è probabile che sostituisca JAX - RPC rapidamente. JAX - WS è supportato da Axis 2.0. WebSphere MQ 7.0.1 non supporta JAX - WS e Axis 2.

Il trasporto WebSphere MQ per SOAP non modifica il contenuto dell'envelope SOAP e il contenuto non influenza il trasporto. I bind di lingua non influiscono sul trasporto WebSphere MQ per SOAP. WebSphere MQ 7.0.1 supporta .NET Framework 1, .NET Framework 2 e Axis 1.4 utilizzando il codice e le utilità forniti con il trasporto WebSphere MQ per SOAP. Il supporto per il trasporto WebSphere per SOAP in .NET

Framework 3 e 3.5 è implementato utilizzando il canale personalizzato WebSphere MQ per Windows Communication Foundation.

Altri ambienti di runtime e di sviluppo SOAP potrebbero fornire il supporto per il trasporto WebSphere MQ per SOAP e supportare lingue differenti. Ad esempio, i servizi Web in esecuzione su CICS supportano linguaggi come COBOL e PL/1.

Nota: La mappatura utilizzata non fa alcuna differenza per l'interoperabilità dei servizi Web. È possibile combinare e associare client e servizi scritti utilizzando associazioni .NET, JAX - RPC e JAX - WS.

Cos' è il trasporto WebSphere MQ per SOAP?

WebSphere MQ transport for SOAP è un bind SOAP e un toolkit di servizi Web. Insieme, consentono alle applicazioni di scambiare messaggi SOAP utilizzando WebSphere MQ piuttosto che HTTP. Figura 167 a pagina 954 mostra WebSphere MQ come alternativa a HTTP come trasporto SOAP.

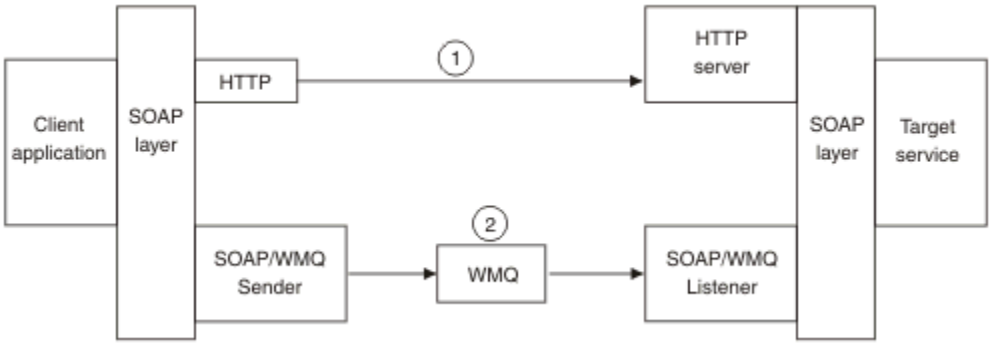


Figura 167. Panoramica del trasporto WebSphere MQ per SOAP

SOAP su HTTP viene mostrato come (1) nel diagramma. Il livello SOAP del client converte una richiesta in messaggio SOAP e il componente HTTP invia su TCP/IP. Il componente del server HTTP è in attesa di richieste HTTP, generalmente sulla porta TCP/IP 80. Se la richiesta è per un servizio SOAP, il componente del server HTTP richiama il livello SOAP per convertire la richiesta SOAP in una chiamata al metodo. Quindi restituisce la risposta.

SOAP su WebSphere MQ viene visualizzato come (2). L'applicazione client registra il componente mittente SOAP WebSphere MQ come gestore per il protocollo `jms:` con il livello SOAP. Il livello SOAP passa i messaggi SOAP indirizzati a `jms:` al mittente SOAP WebSphere MQ . Il mittente utilizza l'URI nel messaggio per posizionare il messaggio nella coda di richieste con le QoS (quality of service) richieste. Il listener SOAP WebSphere MQ corrispondente attende i messaggi sulla coda di richiesta e richiama il livello SOAP per elaborare richieste e restituire risposte.

Il mittente e il listener SOAP sono normali programmi WebSphere MQ . Possono essere connessi allo stesso gestore code, come in Figura 168 a pagina 955, o a gestori code differenti; consultare Figura 169 a pagina 956. Il client può essere collegato mediante una connessione client.

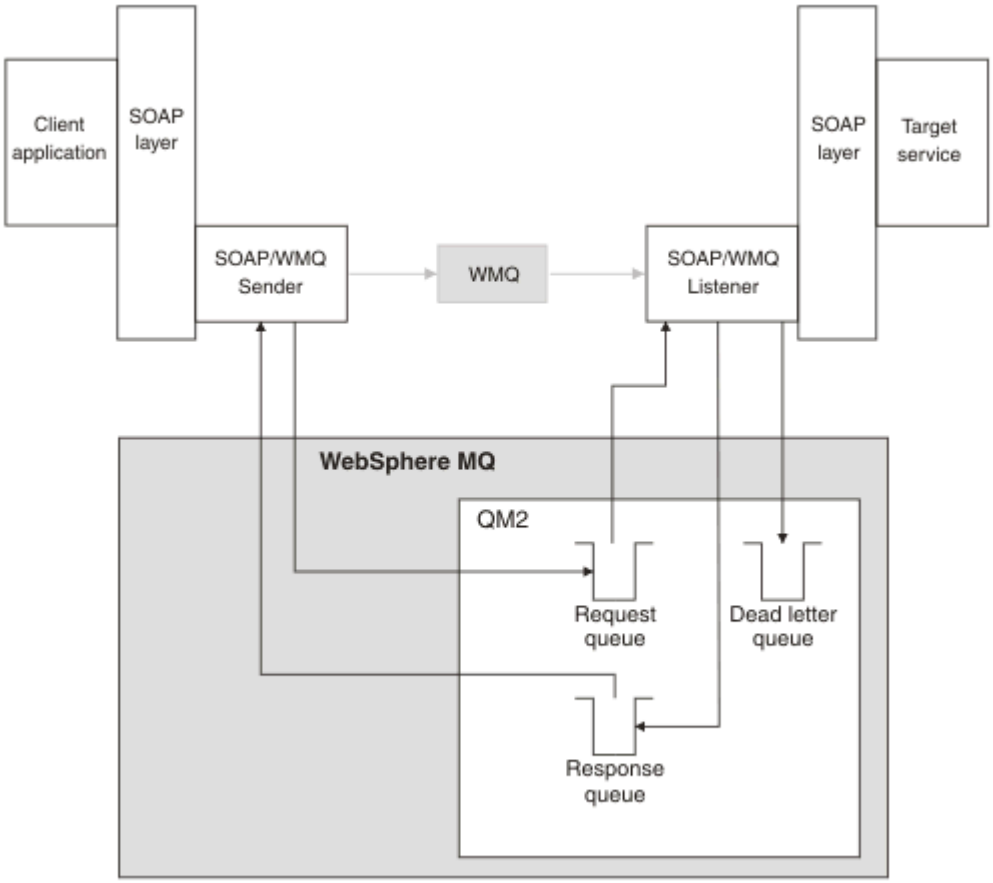


Figura 168. Code utilizzate da SOAP/WebSphere MQ (singolo gestore code)

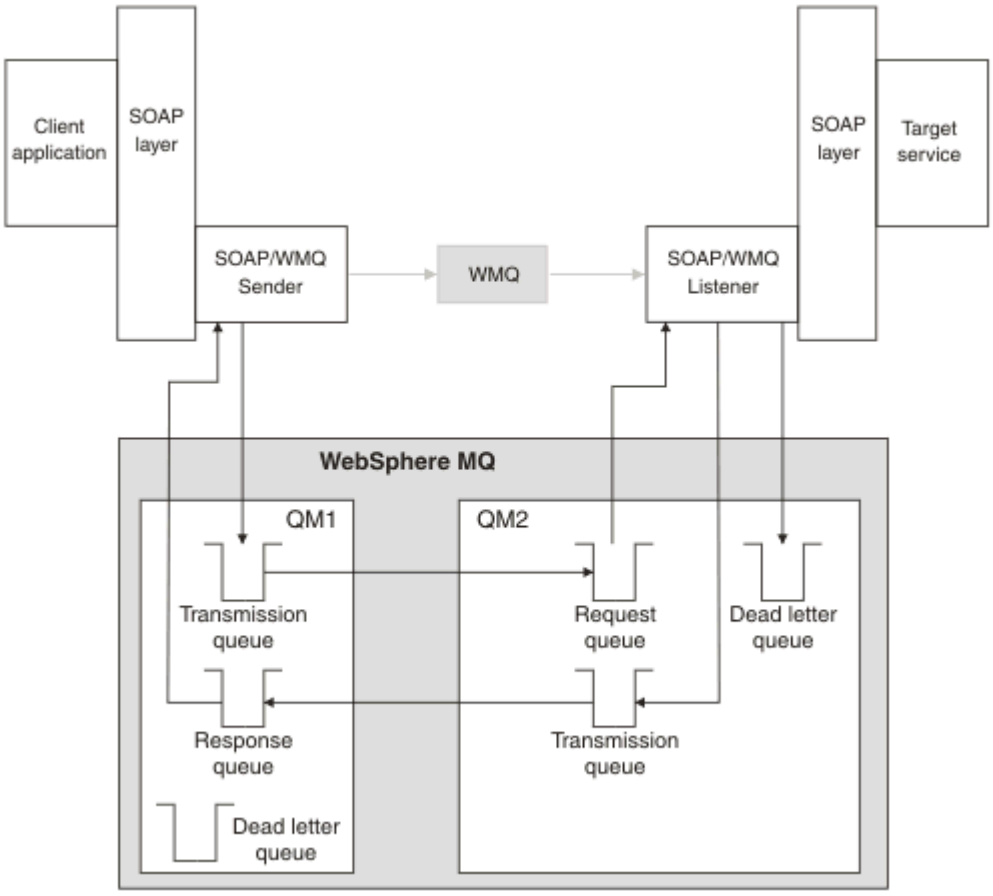


Figura 169. Code utilizzate da SOAP/WebSphere MQ (gestori code separati)

Raccomandazione del candidato W3C per il bind di SOAP a JMS.

La raccomandazione del candidato W3C definisce il binding SOAP su JMS; SOAP su JMS (Java Message Service) 1.0. Anche utile per i suoi esempi è Schema URI per Java (tm) Message Service 1.0[11].

Alcuni framework dell'applicazione, come WebSphere Application Server v7, supportano il suggerimento del candidato W3C . Inviare le richieste SOAP formattate con un URI compatibile con il suggerimento del candidato W3C utilizzando il client Axis2 ; consultare W3C SOAP su JMS URI per il client WebSphere MQ Axis 2 . Il client Axis2 invia una richiesta SOAP formattata con un trasporto W3C o WebSphere MQ per SOAP basato sull'URI nella richiesta SOAP.

Il supporto client Axis2 per il consiglio W3C è stato introdotto nel fixpack 7.0.1.3 . Il supporto per altri client e per i listener SOAP forniti da WebSphere MQ non viene fornito.

Concetti correlati

Implementazione del trasporto WebSphere per SOAP su .NET Framework 1, .NET 2 e Axis 1.4
È possibile scrivere il proprio listener e il mittente SOAP WebSphere MQ . Utilizzare l'implementazione del trasporto WebSphere MQ per SOAP su .NET Framework 1, .NET Framework 2 e Axis 1.4 come guida.

Trasporto WebSphere MQ per SOAP e messaggistica affidabile dei servizi Web
La messaggistica affidabile dei servizi Web è un protocollo per lo scambio affidabile di richieste e risposte dei servizi Web tramite una connessione non affidabile. È più adatto a risolvere problemi di interruzione di connessione di breve durata.

11 Cercare *URI Scheme for JMS* nei riferimenti alla specifica W3C per la bozza più recente.

Implementazione del trasporto WebSphere per SOAP su .NET Framework 1, .NET 2 e Axis 1.4

È possibile scrivere il proprio listener e il mittente SOAP WebSphere MQ . Utilizzare l'implementazione del trasporto WebSphere MQ per SOAP su .NET Framework 1, .NET Framework 2 e Axis 1.4 come guida.

1. Un programma client utilizza il framework dei servizi Web appropriato nello stesso modo in cui utilizza il trasporto HTTP. Deve anche registrare il prefisso `jms:` . Il prefisso viene registrato utilizzando il metodo Java `com.ibm.mq.soap.Register.extension()` o il metodo CLR `IBM.WMQSOAP.Register.Extension()` .
2. Il framework Axis 1.4 o .NET Framework 1 o 2 indica la chiamata in un messaggio di richiesta SOAP esattamente come per SOAP/HTTP.
3. Un servizio WebSphere MQ è identificato da un URI con prefisso `jms:`. Quando il framework identifica l'URI `jms:` , richiama il codice mittente del trasporto WebSphere MQ ; `com.ibm.mq.soap.transport.jms.WMQSender` (per Axis 1.4) o `IBM.WMQSOAP.MQWebRequest` (per .NET1 e 2). Se il framework rileva un URI con un prefisso `http:` , richiama il SOAP standard sul mittente HTTP.
4. Il messaggio SOAP viene trasportato tramite il mittente SOAP WebSphere MQ utilizzando la coda di richiesta. **`SimpleJavaListener`** (per Java) o **`amqwSOAPNETListener`** (per .NET) riceve il messaggio di richiesta.

   I listener SOAP WebSphere MQ sono processi autonomi e sono a più thread con un numero personalizzabile di thread.
5. Il listener WebSphere MQ SOAP legge la richiesta SOAP in entrata e la trasmette all'infrastruttura del servizio Web appropriata.
6. L'infrastruttura del servizio web analizza il messaggio di richiesta SOAP e richiama il servizio. La procedura è la stessa di un messaggio arrivato su un trasporto HTTP.
7. L'infrastruttura formatta la risposta in un messaggio di risposta SOAP e la restituisce al listener SOAP WebSphere MQ .
8. Il listener inserisce il messaggio nella coda di risposta e il messaggio viene trasferito al programma di invio SOAP WebSphere MQ . Il mittente lo trasmette all'infrastruttura del servizio web client.
9. L'infrastruttura client analizza il messaggio SOAP di risposta e consegna il risultato all'applicazione client.

Ogni contesto applicazione viene servito da una coda di richiesta WebSphere MQ separata.

Il contesto dell'applicazione è controllato in Axis 1.4 assicurando che il listener e il servizio SOAP WebSphere MQ vengano eseguiti nella directory appropriata. Axis 1.4 imposta il CLASSPATH corretto per la directory.

Il contesto dell'applicazione è controllato in .NET dal listener SOAP di WebSphere MQ che esegue il servizio in un contesto creato da una chiamata a `ApplicationHost.CreateApplicationHost`. La chiamata specifica la directory di esecuzione di destinazione. Ogni servizio opera quindi nella directory in cui è stato distribuito.

`amqwdeployWMQService` genera le code di richiesta e risposta. Inoltre, genera l'infrastruttura necessaria per la gestione delle code e la distribuzione dei servizi su Axis 1.4.

Concetti correlati

Integrazione di SOAP e WebSphere MQ

Trasporto WebSphere MQ per SOAP e messaggistica affidabile dei servizi Web

La messaggistica affidabile dei servizi Web è un protocollo per lo scambio affidabile di richieste e risposte dei servizi Web tramite una connessione non affidabile. È più adatto a risolvere problemi di interruzione di connessione di breve durata.

Trasporto WebSphere MQ per SOAP e messaggistica affidabile dei servizi Web

La messaggistica affidabile dei servizi Web è un protocollo per lo scambio affidabile di richieste e risposte dei servizi Web tramite una connessione non affidabile. È più adatto a risolvere problemi di interruzione di connessione di breve durata.

WebSphere MQ per SOAP trae vantaggio dall'utilizzo di una rete gestita e affidabile WebSphere MQ per la trasmissione di messaggi SOAP. I trasporti come HTTP e FTP non sono gestiti. Le reti non gestite sono ideali per connessioni imprevedibili, in cui le difficoltà e i costi di gestione delle connessioni superano i vantaggi di non perdere richieste e risposte.

Per superare il problema della perdita di file quando le connessioni si rompono in reti non gestite, i servizi come l'FTP gestito creano un livello di gestione sopra l'FTP. Il livello di gestione si assume l'onere di controllare che i file siano stati trasferiti correttamente dagli utenti e ritrasmette i file mancanti, se necessario. Per utilizzare l'FTP gestito, è necessaria l'installazione del software di gestione su entrambe le estremità della connessione.

WSRM (Web services reliable messaging) utilizza un approccio diverso per risolvere il problema delle connessioni non affidabili. Il suo scopo è quello di trasferire le richieste e le risposte del servizio Web in modo affidabile, senza che entrambe le estremità della connessione debbano utilizzare lo stesso software. Qualsiasi software, implementando il protocollo di messaggistica affidabile dei servizi Web, può scambiare messaggi in modo affidabile con altri software.

Quando una connessione ha esito negativo, un mittente e un destinatario devono conservare il contesto del trasferimento del messaggio WSRM, utilizzando un URI generato come chiave. Il mittente e il destinatario tentano di stabilire una nuova connessione. Se una nuova connessione viene stabilita correttamente, il trasferimento viene completato. La specifica WSRM non specifica il modo in cui viene conservato il contesto o quando viene tentata una nuova connessione.

Si potrebbe decidere che solo le interruzioni di breve durata sono di interesse. Per interruzioni più lunghe, si potrebbe essere pronti a scartare i trasferimenti che non è stato possibile riavviare dopo un periodo di tempo. Allo stesso modo, potresti essere pronto a scartare i trasferimenti se il client o il servizio falliscono. Lasciando l'utente responsabile di assicurare i trasferimenti, pone meno richieste sulla gestione del coordinamento del client e del servizio.

Se le interruzioni di rete sono di lunga durata, più o meno di 30 minuti, o se il client o il server ha esito negativo, vi è una maggiore probabilità che alcune connessioni non vengano mai ristabilite. Non è più possibile fare affidamento sul ripristino automatico del trasferimento messaggi da parte di WSRM in modo non gestito. È necessario considerare la gestione delle connessioni WSRM non riuscite, che implica lo sviluppo di software per gestire la rete di client e servizi.

L'utilizzo di WSRM per superare brevi interruzioni potrebbe ridurre in modo significativo la gestione dei messaggi persi su una rete mobile. Se non è necessario assicurare la consegna dei messaggi, i benefici della riduzione della perdita di messaggi potrebbero giustificare il costo aggiuntivo dello sviluppo di una implementazione WSRM.

SOAP su JMS fornisce la consegna sicura dei messaggi e gestisce le interruzioni più lunghe del client, del server e della rete. Se si sta cercando una QoS (quality of service) più affidabile per SOAP rispetto a HTTP, quale soluzione scegliere: WebSphere MQ transport for SOAP o WSRM? La risposta dipende da molti fattori. Alcuni dei fattori da considerare sono elencati:

1. Se l'inaffidabilità è dovuta a un errore di connessione.
2. Quanto sono lunghi gli errori di connessione.
3. Se è possibile gestire sia il lato client che il lato server della connessione.
4. Se l'utente o un amministratore è in ultima analisi responsabile della consegna dei messaggi.

Concetti correlati

Integrazione di SOAP e WebSphere MQ

Implementazione del trasporto WebSphere per SOAP su .NET Framework 1, .NET 2 e Axis 1.4

È possibile scrivere il proprio listener e il mittente SOAP WebSphere MQ . Utilizzare l'implementazione del trasporto WebSphere MQ per SOAP su .NET Framework 1, .NET Framework 2 e Axis 1.4 come guida.

Installazione e verifica dei servizi Web WebSphere MQ

Utilizzare le istruzioni contenute in questi argomenti per installare e verificare il trasporto WebSphere MQ per SOAP.

Installazione del trasporto Web WebSphere MQ per SOAP

Utilizzare queste istruzioni per installare il trasporto Web WebSphere MQ per SOAP. L'installazione crea gli strumenti per eseguire i servizi o i client del servizio Web utilizzando WebSphere MQ come trasporto SOAP. Gli strumenti vengono utilizzati in ambienti .NET Framework 1, .NET 2, Axis 1.4o Axis2 SOAP.

Prima di iniziare

Controllare i prodotti prerequisiti all'indirizzo Requisiti di sistema per IBM WebSphere MQ. Il processo di installazione non controlla la presenza e disponibilità del software prerequisito. È necessario verificare che i prerequisiti siano installati.

WebSphere MQ fornisce una copia del runtime Axis 1.4 . Utilizzare questa versione con WebSphere MQ piuttosto che con qualsiasi altra versione installata. IBM non fornisce supporto tecnico per Apache Axis. In caso di problemi tecnici, contattare Apache Software Foundation.

Per eseguire i servizi Web nell'ambiente .NET Framework 3 SOAP, WebSphere MQ utilizza Windows Communication Foundation. Il canale personalizzato WebSphere MQ per Windows Communication Foundation esegue i servizi e i client del servizio Web utilizzando WebSphere MQ come trasporto per i messaggi SOAP.

Informazioni su questa attività

È possibile installare il trasporto Web WebSphere MQ per SOAP come un client WebSphere MQ MQI o un'applicazione server. Se WebSphere MQ è già stato installato come client o server sul computer, verificare di aver installato i componenti elencati.

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Effettuare le seguenti operazioni di installazione.

Procedura

1. Selezionare il componente "Messaggistica Java e. Net e servizi Web" per l'installazione.
2. In Solaris e HP-UX, selezionare il componente "Java Runtime Environment" per l'installazione.
3. Selezionare il toolkit di sviluppo per l'installazione.
4. Installare e verificare WebSphere MQ come descritto nella Guida operativa per la propria piattaforma.
5. Copiare il runtime di Apache Axis 1.4 , `axis.jar` dalla directory `prereqs/axis` sul supporto di installazione WebSphere MQ . Copiarlo nella directory di installazione descritta in Tabella 138 a pagina 960, Tabella 139 a pagina 960 o Tabella 140 a pagina 960.

   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
   ```

   Directory di installazione di HP-UX, Solaris e Linux (tutte le piattaforme)

   ```
   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. Su Windows 2003, eseguire **Aspnet_regiis.exe** per aggiornare le associazioni di script in modo che puntino alla versione di Common Language Run che si sta utilizzando.

   Ricercare il programma di utilità **Aspnet_regiis.exe** in %SystemRoot%Microsoft.NET/Framework/*version-number* .
7. Impostare la variabile di ambiente, WMQSOAP_HOME , in modo che punti alla directory di installazione di WebSphere MQ .

Risultati

Tabella 138. Directory di installazione di Windows

Ubicazione	Contenuto
MQ_INSTALLATION_PATH\programs\bin	File binari, comandi, DLL e eseguibili
MQ_INSTALLATION_PATH\programs\java\lib	.jar files
MQ_INSTALLATION_PATH\programs\java\lib\soap	SOAP .jar files
MQ_INSTALLATION_PATH\programs\soap\samples	Campioni e IVT

Tabella 139. Directory di installazione di AIX

Ubicazione	Contenuto
MQ_INSTALLATION_PATH/bin	Script shell
MQ_INSTALLATION_PATH/java/lib	.jar files
MQ_INSTALLATION_PATH/java/lib/soap	axis.jar e altri file JAX - RPC .jar
MQ_INSTALLATION_PATH/samp/soap	Campioni e IVT

Tabella 140. Directory di installazione di HP-UX, Solaris e Linux (tutte le piattaforme)

Ubicazione	Contenuto
MQ_INSTALLATION_PATH/bin	Script shell
MQ_INSTALLATION_PATH/java/lib	.jar files
MQ_INSTALLATION_PATH/java/lib/soap	axis.jar e altri file JAX - RPC .jar
MQ_INSTALLATION_PATH/samp/soap	Campioni e IVT

Operazioni successive

1. Solo per .NET, è necessario registrare il trasporto WebSphere MQ per i file SOAP con Global Assembly Cache. Se .NET è già installato quando si installa WebSphere MQ, la registrazione viene eseguita automaticamente al momento dell'installazione. Se si installa .NET dopo WebSphere MQ, la registrazione viene eseguita automaticamente quando l'IVT viene eseguito per la prima volta.

   È possibile eseguire **amqiregisterdotnet.cmd** per eseguire la registrazione degli assembly .NET. Puoi anche eseguire **amqiregisterdotnet.cmd** per forzare la riregistrazione in qualsiasi fase. Una volta effettuata, questa registrazione sopravvive ai riavvii del sistema e la successiva riregistrazione non è normalmente necessaria.
2. Eseguire il test di verifica dell'installazione, come descritto in "Verifica del trasporto IBM WebSphere MQ per SOAP" a pagina 961.

3. Se si intende sviluppare il client Axis2 , è necessario scaricare Axis2 1.4.1 da Apache; consultare “Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse” a pagina 984.

Verifica del trasporto IBM WebSphere MQ per SOAP

Verificare il trasporto IBM WebSphere MQ per SOAP utilizzando il comando **runivt** . Il comando esegue diverse applicazioni dimostrative e garantisce che l'ambiente sia impostato correttamente dopo l'installazione.

Prima di iniziare

Prima di eseguire il comando **runivt** , assicurarsi di disporre dei seguenti ambienti di runtime:

- Per eseguire solo su Axis: è necessario disporre di un Java SDK (in SOE) disponibile sul sistema. È necessario includere anche l'ubicazione dei comandi `java.exe` e `javac.exe` nella variabile di ambiente dei sistemi **PATH** .
- Per eseguire un test solo su .NET (supportato solo su Windows): è necessario disporre di un SDK Java e di strumenti e compilatori .NET sul sistema. A tale scopo, accedere al prompt dei comandi di Visual Studio o al prompt dei comandi di Microsoft Windows SDK, quindi aggiungere l'ubicazione dei file `java.exe` e `javac.exe` alla variabile di ambiente **PATH** .
- Per eseguire tutti i test disponibili: per le piattaforme Windows , l'ambiente deve essere configurato come descritto nell'esecuzione del test .NET. Su piattaforme UNIX and Linux , l'ambiente deve essere configurato come descritto nell'esecuzione di test solo Axis.

Informazioni su questa attività

Invece di eseguire il test di verifica su .NET e Axis, è possibile eseguire il test solo su Axis o solo su .NET.

Se si verificano problemi con i test e si desidera ricominciare:

1. Arrestare il gestore code `WMQSOAP.DEMO.QM` utilizzando l'opzione `immediate` .
2. Arrestare il listener che è stato avviato in una finestra diversa.
3. Eliminare il gestore code.
4. Eliminare la directory degli esempi temporanei creata e riavviare.

Su piattaforme UNIX and Linux , è necessario eseguire il comando utilizzando una sessione di sistema X Windows .

Il comando **runivt** modifica il contenuto della directory `soap/samples` . Per mantenere invariata l'immagine di installazione, copiare la directory degli esempi in una posizione temporanea ed eseguire il test di verifica dalla posizione temporanea.

È possibile eseguire la verifica dell'installazione tutte le volte che si desidera.

Effettuare le seguenti operazioni per verificare l'installazione del trasporto IBM WebSphere MQ per SOAP su .NET Framework 1, .NET Framework 2 e Axis 1.4:

Procedura

1. Copiare la struttura di directory `./tools/soap/samples` in un'ubicazione temporanea.
2. Avviare una finestra comandi con la directory temporanea come directory corrente.
3. Utilizzare il comando **runivt** per avviare il test di installazione. Lo script `runivt` compila una classe di test, un client di esempio e i servizi prima di distribuirli ed eseguirli. Per la classe di test, il client di esempio e i servizi da eseguire, completare i passi di installazione descritti in Installazione del trasporto Web WebSphere(r) MQ per SOAP e assicurarsi che il prompt dei comandi utilizzato per eseguire il comando `runivt` abbia l'ambiente di runtime richiesto impostato. Utilizzare uno dei seguenti metodi per eseguire il comando **runivt** :
   - Eseguire un test solo su Axis `runivt Axis`.
   - Eseguire un test solo su .NET (supportato solo su Windows): `runivt DotNet`.

- Eseguire tutti i test disponibili`runivt`.

  Per ulteriori informazioni sulla sintassi del comando `runivt` e sui parametri, consultare **`runivt`**: IBM WebSphere MQ transport for SOAP installation verification test. I test che è possibile eseguire sono elencati nel file `ivttests.txt` su Windows e `ivttests_unix.txt` su piattaforme UNIX and Linux .

Riferimenti correlati

runivt: Trasporto WebSphere MQ per il test di verifica dell'installazione SOAP

Sviluppo di servizi Web per il trasporto WebSphere MQ per SOAP

Utilizzare il normale ambiente di sviluppo del servizio Web per sviluppare servizi da utilizzare con il trasporto WebSphere MQ per SOAP.

Prima di iniziare

1. Se si intende utilizzare gli strumenti della riga comandi forniti con il trasporto WebSphere MQ per SOAP:
   a. Creare una directory di installazione per il servizio.
   b. Avviare una finestra comandi nella directory.
   c. Per .NET, csc.exe e wsdl.exe devono trovarsi nel percorso e devono essere della stessa versione di .NET Framework.
   d. Per Java,
      i) Eseguire il comando **amqwsetcp** per impostare il percorso classi.
      ii) Un JRE IBM e un JDK allo stesso livello di versione devono trovarsi nel percorso corrente. il livello di versione deve essere almeno 5.0.
      iii) Personalizza il percorso classi per includere le ubicazioni di eventuali librerie `.jar` aggiuntive e le directory contenenti i pacchetti .java, anche per il servizio che stai sviluppando. Inserire la directory corrente "`.`" nel percorso classi.
      iv) Creare una directory, relativa alla directory corrente della finestra comandi, corrispondente al nome pacchetto del servizio che si sta sviluppando.
2. In alternativa, utilizzare gli strumenti del workbench che supportano lo sviluppo dei servizi Web. Le attività di sviluppo di esempio utilizzano Microsoft Visual Studio 2008, Eclipse IDE per sviluppatori Java EE e WebSphere Application Server Community Edition.

Informazioni su questa attività

I servizi Web esistenti non richiedono alcuna modifica per funzionare con il trasporto WebSphere per SOAP. Gli strumenti forniti con il trasporto WebSphere MQ per la distribuzione SOAP di un servizio Web ed eseguirlo utilizzando un listener SOAP WebSphere MQ . Gli strumenti generano anche stub client WSDL, .NET e classi proxy `.java` per sviluppare il trasporto WebSphere MQ per i client SOAP.

Attieniti a questi passi per creare un servizio e prepararlo per la distribuzione e la generazione di client. Seguire i passi nelle attività correlate per creare un servizio utilizzando Eclipse o Microsoft Visual Studio 2008.

Procedura

1. Sviluppa il servizio utilizzando il normale ambiente di sviluppo.
2. Verifica il servizio utilizzando un client dei servizi web HTTP
3. Attenersi alla seguente procedura per preparare la directory di distribuzione:
   - Per Java
     a. Copiare il file `.java` che definisce l'interfaccia del servizio nella directory di distribuzione.
     b. Copiare tutti i file `.class` per il servizio nella directory corrispondente al nome del pacchetto.

c. Verificare che il percorso classi sia in grado di individuare tutte le classi richieste: compilare il file `.java` del servizio utilizzando **javac**.

- Per .NET
  a. Copiare il file `.asmx` che definisce il servizio nella directory di installazione e
  b. Se si utilizza il modello di codice, copiare i file `.dll` in una directory *deployment directory*`\bin` .

Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse

Sviluppare un servizio Web Axis 1.4 da eseguire utilizzando WebSphere MQ come provider del servizio. Utilizzare il normale ambiente di sviluppo del servizio Web per creare un servizio per la distribuzione su Axis 1.4.

Prima di iniziare

Considerare i requisiti per la distribuzione di un server Web al listener WebSphere MQ SOAP per Axis 1.4.

- Il listener WebSphere MQ SOAP per Axis 1.4 richiede un JRE IBM versione 5.0 o superiore. JRE e JDK utilizzati per lo sviluppo devono essere allo stesso livello di versione.
- Il listener SOAP WebSphere MQ per Axis 1.4 richiede `axis.jar` installato con WebSphere MQ. Modificare il percorso di generazione nel proprio ambiente di sviluppo in modo che faccia riferimento al file `axis.jar` installato con WebSphere MQ, piuttosto che ai file `axis.jar` installati con l'ambiente di sviluppo.
- Fino a, e includendo, WebSphere MQ V7.0.1, il WSDL generato per il servizio distribuito è RPC/codificato. Da V7.1, è anche possibile richiedere lo stile RPC/literal WSDL. Il WSDL generato viene utilizzato solo per la distribuzione. È possibile definire il servizio utilizzando WSDL conforme a WS-I.

Informazioni su questa attività

Creare il servizio utilizzando il normale ambiente di sviluppo dei servizi Web.

In questa attività, utilizziamo l'IDE Eclipse Java EE open source per sviluppatori Web, noto come Galileo. Per il server delle applicazioni, viene utilizzato WebSphere Application Server Community Edition v2.1 (Community Edition), basato su Geronimo. Consultare le attività correlate per informazioni su come ottenere, installare e configurare l'IDE e il server.

Verificare il servizio utilizzando HTTP come trasporto utilizzando Esplora servizi Web fornito nell'IDE precedente. In alternativa, generare un client proxy HTTP e verificare il servizio utilizzando il proprio codice client.

È possibile seguire questa procedura per sviluppare un servizio Web dal basso verso l'alto. Utilizzare il programma di esempio `StockQuoteAxis.java` come esempio.

Procedura

1. Avviare Eclipse IDE per Java EE Developers con un nuovo spazio di lavoro.
2. Configurare lo spazio di lavoro per utilizzare Java50,
   WebSphere Application Server Community Edition 2.1.4 non funziona con Java60.
   a) **Finestra** > **Preferenze** > **Java** > **JRE installati** > **Aggiungi ...** > **VM standard** > **Avanti** > **Directory ...**
   b) Ricercare la directory di installazione di **Java50** > **OK** > **Fine**
   c) Verificare il JRE **Java50** > OK
3. Aggiungere l'ambiente di runtime Community Edition e avviare Community Edition.
   a) **Finestra** > **Preferenze** > **Server** > **Ambienti di runtime** > **Aggiungi ...**

b) Selezionare **IBM WASCE v2.1** dall'elenco **Nuovo ambiente di runtime server** > Selezionare **Crea un nuovo server locale** > **Avanti**

Se **IBM WASCE 2.1** non è presente nell'elenco, è necessario completare altre due attività:

i) Installare WebSphere Application Server Community Edition.

ii) Installare l'aggiornamento Eclipse per Community Edition.

Per i dettagli, consultare WebSphere Application Server Community Edition

c) Passare alla directory di installazione di Application Server> **OK** > **Fine** > **OK**.

d) Fare clic con il pulsante destro del mouse su **IBM WASCE v2.1 server** nella vista Server> **Avvia**

Suggerimento: Ô possibile gestire WASCE in Eclipse: fare clic con il tastino destro del mouse su **IBM WASCE v2.1 server** > **Avvia console WASCE** . le **Username** e **Password** predefinite sono `system` e `manager` .

4. Impostare il server e il tempo di esecuzione per servizi Web.

a) **Finestra** > **Preferenze** > **Servizi Web** > **Server e Runtime**

b) Selezionare **IBM WASCE v2.1 Server** come server.

c) Lasciare **Apache Axis** come runtime del servizio Web.

5. Creare un Progetto Web dinamico.

a) **File** > **Nuovo** > **Progetto Web dinamico**.

Denominare il progetto `StockQuoteAxis`.

b) Selezionare **Aggiungi progetto a un EAR** > **Nuovo ...**

c) Nella pagina **Progetto applicazione EAR** , immettere **Project name** `StockQuoteAxisEAR` > **Fine**

Rispondere **OK** in risposta alla finestra di dialogo suggerendo di passare alla prospettiva Java EE oppure mantenere la prospettiva Java per seguire esattamente queste istruzioni.

d) **IBM WASCE 2.1 server** è selezionato come runtime di destinazione. Accettarlo e gli altri valori predefiniti> **Fine**.

Rispondere **OK** in risposta alla finestra di dialogo suggerendo di passare alla prospettiva Java EE oppure mantenere la prospettiva Java per seguire esattamente queste istruzioni.

6. Importazione del programma di esempio `StockQuoteAxis.java`

a) Aprire il progetto Web **StockQuoteAxis** > Fare clic con il tasto destro del mouse sulla cartella **src** > **Importa ...**

b) Selezionare **Generale** > **File System** > **Avanti**

c) Passare a *MQ_INSTALLATION_PATH*`\tools\soap\samples\java\server` > check **StockQuoteAxis.java** > **Fine**

MQ_INSTALLATION_PATH rappresenta la directory di alto livello in cui è installato WebSphere MQ . È necessario evidenziare la directory del server per visualizzare i file che contiene.

7. Correggere l'errore di compilazione spostando `StockQuoteAxis.java` nel pacchetto corretto.

a) Aprire `StockQuoteAxis.java` e fare clic con il tasto destro del mouse sul problema> **Fix rapida**

b) Fare doppio clic su **Move 'StockQuoteAxis.java' to package 'soap.server'** > **Salva**.

8. Creare un servizio Web da `StockQuoteAxis.java` .

a) Fare clic con il pulsante destro del mouse su `StockQuoteAxis.java` > **Servizi Web** > **Crea servizio Web** > **Avanti**.

Accettare la configurazione predefinita per il servizio:

Tipo di servizio Web
Servizio Java beanWeb bottom up

Implementazione del servizio
soap.server.StockQuoteAxis

Server
IBM WASCE v2.1 server

Runtime del servizio web
Asse Apache

Progetto di servizio
Asse StockQuote

Progetto EAR di servizio
StockQuoteAxisEAR

Configurazione
Nessuna generazione client

9. Selezionare i metodi a cui accedere e lo stile del servizio Web> **Avanti**.

   Avviare il server, se richiesto.

   a) Lascia tutti i metodi selezionati.

   b) Selezionare lo stile documento/letterale (a capo).

10. **Fine**

    Una volta distribuito il servizio, cercare nella cartella `WebContent\wsdl` nel progetto Web `StockQuoteAxis` e trovare il file `StockQuoteAxis.wsdl` generato.

11. Verificare il servizio utilizzando HTTP con Esplora servizi Web.

    a) Fare clic con il pulsante destro del mouse su `StockQuoteAxis.wsdl` > **Verifica con Esplora servizi Web**.

    b) Fare clic sulle azioni **getQuote** in **StockQuoteAxisSoapBinding** nella finestra **Esplora servizi Web** .

    c) Immettere `ibm` nel campo di immissione **simbolo** > **Vai**

12. Verificare il servizio utilizzando il trasporto WebSphere MQ per SOAP.

    Per distribuire il servizio, utilizza **`SimpleJavaListener`**, che si trova in `com.ibm.mq.soap.jar`. È necessario aggiungere le librerie Java WebSphere MQ Java e SOAP al percorso build.

    a) Fare clic con il tasto destro del mouse sul progetto Web **StockQuoteAxis** > **Percorso build** > **Configura percorso build ...**

    b) Fare clic su **Librerie** > **Aggiungi jar esterni ...**. Passare a *`MQ_INSTALLATION_PATH`*`\java\lib` . e selezionare tutti i file `.jar` > **Apri** > **Aggiungi Jars esterni ...** Passare a *`WMQ Install directory`*`\java\lib\soap` e selezionare tutti i file `.jar` > **Apri** > **OK**.

    `MQ_INSTALLATION_PATH` rappresenta la directory di alto livello in cui è installato WebSphere MQ .

    c) In Explorer progetti, fare clic su `StockQuoteAxis\Java Resources\Libraries\com.ibm.mq.soap.jar\com.ibm.mq.soap.transport.jms\SimpleJavaListener.class\ SimpleJavaListener` > **Esegui come ...**> **Esegui configurazioni ...**

    Suggerimento:

    Se non esiste alcuna configurazione per `SimpleJavaListener` , fare clic su **Nuova configurazione** nella pagina **Crea, gestisci ed esegui configurazioni** della procedura guidata **Esegui configurazioni** ,

    `SimpleJavaListener` non dispone di alcun comando per arrestarlo. Per monitorare o arrestare **`SimpleJavaListener`**, aprire la **Prospettiva di debug** in Eclipse.

    d) Aprire la scheda **(x) = Argomenti** . Nell'area di immissione **Argomenti del programma** immettere i parametri in **`SimpleJavaListener`**.

    Per questo esempio, immettere

```
-u "jms:/queue?destination=REQUESTAXIS@QM1&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

```
&targetService=StockQuoteAxis
&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE" -n 10
```

Nota: Il servizio di destinazione è `StockQuoteAxis` per corrispondere al nome del servizio di destinazione creato nel descrittore di distribuzione del servizio, `StockQuoteAxis\WebContent\WEB-INF\server-config.wsdd` . amqwdeployWMQService crea un servizio di destinazione denominato `soap.server.StockQuoteAxis`. In questo esempio, si utilizzano gli stessi `StockQuoteAxis.class` e `service-config.wsdd` del server HTTP.

e) Nella stessa scheda, configurare **Directory di lavoro** per fare riferimento al file `server-config.wsdd` : `${workspace_loc:StockQuoteAxis/WebContent/WEB-INF}`

a) **Esegui**

Gli errori vengono scritti nella console. Se la console rimane vuota, **`SimpleJavaListener`** è stato avviato correttamente.

b) Per verificare la distribuzione, eseguire un client Axis StockQuote, sviluppato nell'attività, “Sviluppo di un client JAX - RPC per il trasporto WebSphere per SOAP utilizzando Eclipse” a pagina 976.

Esempio: programma di esempio StockQuoteAxis

Il servizio Web Java di esempio, `StockQuoteAxis.java`, è installato in *`WMQ install directory`*`\tools\soap\samples\java\server` . `StockQuoteAxis.java`, Figura 170 a pagina 967, ha quattro metodi:

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);
    }
}
```

Figura 170. Asse StockQuote

Operazioni successive

Distribuire il servizio utilizzando WebSphere MQ Transport for SOAP, invece di HTTP utilizzando il comando **amqwdeployWMQService**.

Il comando ha un'opzione, `axisDeploy`, che distribuisce il servizio creando un descrittore di distribuzione Apache Axis 1.4 . Il listener SOAP WebSphere MQ esegue il servizio. Il listener SOAP è denominato SimpleJavaListener e viene fornito con il trasporto WebSphere MQ per SOAP.

Attività correlate

Sviluppo di un servizio .NET 1 o 2 per il trasporto WebSphere MQ per SOAP utilizzando Microsoft Visual Studio 2008
Sviluppa il servizio Web SampleStockQuote per .NET 1 o .NET 2 utilizzando Microsoft Visual Studio 2008

Sviluppo di un servizio Web EJB JAX - WS per SOAP W3C su JMS
Un servizio Web collegato al suggerimento candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server di applicazioni JEE. Questa attività è il passo 2 della connessione di un client del servizio Web Axis2 e di un servizio Web distribuito su WebSphere Application Server utilizzando il protocollo SOAP W3C su JMS.

Sviluppo di un servizio .NET 1 o 2 per il trasporto WebSphere MQ per SOAP utilizzando Microsoft Visual Studio 2008

Sviluppa il servizio Web SampleStockQuote per .NET 1 o .NET 2 utilizzando Microsoft Visual Studio 2008

Informazioni su questa attività

Crea il servizio StockQuote con una implementazione di codice utilizzando Visual Studio 2008.

Procedura

1. Creare un modello per il servizio e verificare che venga eseguito su HTTP.
   a) Avviare Visual Studio 2008> **File** > **Nuovo** > **Progetto**. Selezionare **C#** Tipo di progetto **, NET Framework 2**e **ASP.NET Applicazione servizio Web**NET. Immettere il nome **:** e il nome della soluzione **:** `StockQuoteDotNet` > **OK**
   b) Fare clic con il pulsante destro del mouse su **Service1.asmx** in **Esplora soluzioni** > **Rinomina** > `StockQuote.asmx`.
   c) Modificare il frammenti di codice `public class Service1` in `public class StockQuote`.
   d) Fare clic con il tasto destro del mouse su **StockQuote.asmx** in **Esplora soluzioni** > **Apri con ...** > **Editor XML**. Modificare `Class="StockQuoteDotNet.Service1"` in `Class="StockQuoteDotNet.StockQuote"`
   e) Modificare il frammenti di codice `[WebService(Namespace = "http://tempuri.org/")]` in `[WebService(Namespace = "http://stock.samples/")]`.
   f) Rimuovere la riga di codice `[ToolboxItem(false)]`.
   g) Verificare che tutto sia corretto: **Debug** > **Avvia debug (F5)**. Verificare l'output in Explorer.
2. Aggiungi i metodi dall'esempio SQDNNonInline`.asmx.cs`e verifica il servizio su HTTP.
   a) Aprire *MQ_INSTALLATION_PATH*`\tools\soap\samples\dotnet\SQDNNonInline.asmx.cs` e sostituire il metodo `HelloWorld` con i quattro metodi `Quote` ; consultare Figura 171 a pagina 969. *MQ_INSTALLATION_PATH* rappresenta la directory in cui è installato WebSphere MQ .
   b) **Crea** > **Ricrea** la soluzione> Fare clic con il tasto destro del mouse su uno dei **Thread** in errore> **Risolvi** > Utilizzo di **System.Threading**.
   c) Premere F5 per avviare il debug.

      Il servizio non è conforme a WS-I Basic Profile v1.1. È possibile modificare l'annotazione WebMethod da `[SoapRpcMethod]` a `[SoapDocumentMethod]`o rimuovere l'annotazione `[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]`.
   d) Premere F5 per verificare l'implementazione utilizzando HTTP.
3. Generare WSDL, client ed eseguire il servizio utilizzando il trasporto WebSphere MQ per SOAP.
   a) Aprire una finestra comandi nella struttura ad albero della directory del progetto, in cui è memorizzato `StockQuote.asmx` .
   b) (Facoltativo) Utilizzare amqwdeployWMQService per generare le risorse. Il gestore code deve essere avviato:

      ```
      amqswdeployWMQService -f StockQuote.asmx
      -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
      &connectionFactory=()
      &destination=REQUESTDOTNET@QM1
      &targetService=StockQuote.asmx"
      StockQuote.asmx StockQuote.wsdl
      ```

      Tutte le risorse utente vengono create nella struttura di directory `./generated` .
   c) (Facoltativo) Generare solo il WSDL per richiamare il servizio utilizzando il trasporto WebSphere MQ per SOAP.

      ```
      amqswsdl -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
      &connectionFactory=()
      &destination=REQUESTDOTNET@QM1
      &targetService=StockQuote.asmx"
      StockQuote.asmx StockQuote.wsdl
      ```
   a) Eseguire il listener .NET. Utilizzare `.\generated\server\startWMQNListener.cmd` o immettere il comando:

      ```
      amqSOAPNETListener -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
      &connectionFactory=()
      &destination=REQUESTDOTNET@QM1
      &targetService=StockQuote.asmx"
      ```

4. Verificare il servizio utilizzando un client generato da WSDL o utilizzando i client generati da **amqwdeployWMQService**.

Codice d'esempio

Il servizio Web .NET di esempio, `StockQuoteDotNet`, è installato in *MQ_INSTALLATION_PATH*`\tools\soap\samples\dotnet` *MQ_INSTALLATION_PATH* è la directory in cui è installato WebSphere MQ . Il bind del servizio Web degli esempi pubblicati differisce leggermente dal bind utilizzato nell'attività. L'attività utilizza i valori predefiniti utilizzati in Visual Studio 2008.

Esistono due esempi di servizi Web .NET Framework 1 e .NET Framework 2. `StockQuoteDotNet.asmx`, è un servizio in linea. `SQDNNoninline.asmx`, è un servizio Web con codice implementato da `SQDNNoninline.asmx.cs`.

`StockQuoteDotNet` ha quattro metodi:

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;
     }
}
```

Figura 171. Servizio in linea: `StockQuoteDotNet.asmx`

```
<%@ WebService Language="C#" Codebehind="SQDNNonInline.asmx.cs" Class="SQDNNonInline" %>
```

Figura 172. Code - behind: Progettazione `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;
    }
}
```

Figura 173. Code - behind: Implementazione: SQDNNonInline.asmx.cs

Attività correlate

Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse
Sviluppare un servizio Web Axis 1.4 da eseguire utilizzando WebSphere MQ come provider del servizio. Utilizzare il normale ambiente di sviluppo del servizio Web per creare un servizio per la distribuzione su Axis 1.4.

Sviluppo di un servizio Web EJB JAX - WS per SOAP W3C su JMS
Un servizio Web collegato al suggerimento candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server di applicazioni JEE. Questa attività è il passo 2 della connessione di un client del servizio Web Axis2 e di un servizio Web distribuito su WebSphere Application Server utilizzando il protocollo SOAP W3C su JMS.

Sviluppo di un servizio Web EJB JAX - WS per SOAP W3C su JMS

Un servizio Web collegato al suggerimento candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server di applicazioni JEE. Questa attività è il passo 2 della connessione di un client del servizio Web Axis2 e di un servizio Web distribuito su WebSphere Application Server utilizzando il protocollo SOAP W3C su JMS.

Prima di iniziare

Utilizzare Rational Application Developer per creare il servizio Web EJB. La procedura guidata dei servizi Web in Rational Application Developer dispone di un'opzione per creare un servizio Web utilizzando il suggerimento candidato W3C per il bind SOAP su JMS. Rational Application Developer 7.54 è obbligatorio. L'esercitazione ha utilizzato Rational Application Developer incluso in Rational Software Architect per WebSphere Software v7.5.5.1,

L'EJB viene distribuito su WebSphere Application Server da Rational Application Developer come parte di questa attività. È necessario completare “Configurazione di WebSphere Application Server per utilizzare SOAP W3C su JMS” a pagina 1008

Per creare il WSDL effettivamente utilizzato nell'attività, è necessario prima completare l'attività, “Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse” a pagina 963. Quindi, è possibile importare il WSDL dal progetto Web dinamico nello spazio di lavoro Eclipse Galileo o dal servizio Web HTTP in esecuzione distribuito a WASCE.

WebSphere Application Server potrebbe essere ancora in esecuzione come risultato dell'esecuzione di “Configurare le WebSphere Application Server” a pagina 1010. In caso contrario, è possibile avviarlo dalla vista Server in RAD.

Informazioni su questa attività

In questa attività, si ridistribuirà il servizio `StockQuoteAxis` dall'esecuzione come servizio dell'asse JAX - RPC eseguito da **`SimpleJavaListener`** utilizzando il trasporto WebSphere MQ per SOAP al servizio JAX - WS in esecuzione in WebSphere Application Server utilizzando il protocollo SOAP over JMS W3C .

Esistono due parti per migrare il servizio da **`SimpleJavaListener`** a WebSphere Application Server:

1. Generare l'interfaccia del servizio Web dal WSDL per il servizio utilizzando la procedura guidata del servizio Web EJB di livello superiore in Rational Application Developer.
2. Implementazione del servizio importando l' esempio SOAP WebSphere MQ `StockQuoteAxis.java`.

Un approccio alternativo sarebbe stato generare il servizio dal basso verso l'alto, da `StockQuoteAxis.java`. Tuttavia, per essere certi che l'interfaccia per il servizio migrato sia identica, l'approccio dall'alto verso il basso è migliore, in quanto utilizza lo stesso WSDL.

Il servizio Web è sviluppato per il contenitore EJB e non per il contenitore Web poiché il supporto JMS fa parte del contenitore EJB.

Procedura

1. Avviare Rational Application Developere verificare che WebSphere Application Server sia in esecuzione.
   a) Avviare Rational Application Developer in un nuovo workspace.
   b) Aprire la prospettiva Java EE .
   c) Aprire il separatore **Server** e controllare che WebSphere Application Server sia in esecuzione.
      - Se non è presente alcun WebSphere Application Server v7.0 nella vista, fare clic con il pulsante destro del mouse nella vista> **Nuovo** > **server**. Seguire le scelte nella procedura guidata per creare un'istanza WebSphere Application Server v7.0 .
      - Se il server è presente, ma non avviato, fare clic sulla freccia per avviarlo.
      - Per verificare le proprietà e ottenere un rapido accesso ai log del server, fare clic con il tasto destro del mouse su **WebSphere Application Server v7.0 in localhost** > **Proprietà** > **WebSphere Application Server**.
      - Per gestire il server, utilizzare un browser esterno e aprire l'URL `http://localhost:9061/ibm/console/unsecureLogon.jsp`oppure fare clic con il tasto destro del mouse su **WebSphere Application Server v7.0 su localhost** > **Esegui console di gestione**.
      - L'impostazione predefinita è la pubblicazione automatica. Molte persone preferiscono distribuire manualmente gli aggiornamenti al server. Fare doppio clic su **WebSphere Application Server v7.0 su localhost**ed espandere la freccia **Pubblicazione** nella finestra**Panoramica** . Fare clic su **Non pubblicare mai automaticamente**.
      - Un altro valore predefinito che si potrebbe voler modificare è quello di deselezionare la casella di spunta **Termina server all'arresto del workbench** nella finestra **Panoramica** .
2. Creare i progetti JEE

È necessario creare un progetto EAR (Enterprise Application Project) e un progetto EJB (Enterprise Java Bean).

a) **File** > **Nuovo** > **Enterprise Application Project**. Denominare il progetto `W3CJMSEAR` > **Fine**.

I valori predefiniti devono identificare WebSphere Application Server v7.0 come runtime di destinazione e EAR versione 5.0. È necessario selezionare la configurazione predefinita.

b) **File** > **Nuovo** > **Progetto EJB**. Denominare il progetto `W3CJMSEJB`. Selezionare `W3CEARJMS` come **Nome progetto EAR** > **Avanti**.

La versione del modulo EJB predefinito è 3.0 e la configurazione predefinita viene riutilizzata.

c) Deselezionare la casella di spunta **Crea un modulo JAR client EJB** > **Fine**.

3. Generare e distribuire il servizio Web EJB dal WSDL `StockQuoteAxis` .

a) **Esegui** > **Avviare Esplora servizi Web**.

b) Selezionare la pagina WSDL utilizzando le icone nella finestra**Esplora servizi Web** > fare clic su **WSDL principale** in Navigator.

c) Nella finestra **Azioni** , immettere o ricercare l'URL WSDL per `StockQuoteAxis.wsdl`.

Se WASCE è in esecuzione con `StockQuoteAxis` distribuito come servizio HTTP, l'URL è:

```
http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl
```

Se si dispone di WSDL nel file system, l'URL potrebbe essere:

```
File:\Dirpath\StockQuoteAxis\WebContent\wsdl\StockQuoteAxis.wsdl
```

d) Fare clic sulla riga contenente l'URL importato nella struttura ad albero Navigator .

È la riga che segue immediatamente **WSDL principale**, se è il primo WSDL importato in Esplora servizi Web.

e) Nella finestra **Azioni** , fare clic su **Avvia procedura guidata servizio Web** > **Scheletro servizio Web** > **Vai**

f) Nella procedura guidata Servizio Web, selezionare **Servizio Web EJB dall'alto in basso**

Selezionare o verificare la configurazione utilizzando le informazioni da Tabella 141 a pagina 972 Controlla **Sovrascrivi file senza avvertenza** > **Avanti**.

Tabella 141. Configurazione del servizio Web EJB dall'alto verso il basso

Campo	**Valore**
Server	`WebSphere Application Server v7.0`
Runtime del servizio Web	`IBM WebSphere JAX-WS`
Progetto di servizio	`W3CJMSEJB`
Progetto EAR di servizio	`W3CJMSEAR`
Configurazione:	`No client generation`

g) Nella pagina sottotitolata, **Specificare opzioni per creare un WebSphere JAX - WS EJB Top Down Web Service**, selezionare la casella **Passa al bind JMS**. Selezionare inoltre **Abilita stile wrapper**, **Copia WSDL nel progetto** e **Genera descrittore di distribuzione del servizio web** > **Avanti**.

h) Nella pagina intitolata **WebSphere JAX - WS JMS Binding Configuration**, selezionare **Use SOAP/JMS interoperability protocol** e fornire i valori da Tabella 142 a pagina 972, lasciando vuoti gli altri campi> **Next**.

Tabella 142. WebSphere Configurazione bind JMS JAX - WS

Campo	**Valore**
Destinazione JMS	queue

Tabella 142. WebSphere Configurazione bind JMS JAX - WS (Continua)	
Campo	**Valore**
Nome JNDI di destinazione:	`requestaxis`
Factory connessioni JMS	`qm1`
Nome di risposta	`W3CJMSEAR`
Configurazione:	`replyaxis`

a) Nella pagina intitolata **WebSphere JAX - WS Router Project Configuration**, immettere `qm1as` nel campo **ActivationSpec Nome JNDI** > **Avanti**.

RAD impiega da circa 30 secondi a un minuto per creare e distribuire il progetto.

b) Ignorare le opzioni nella pagina **Pubblicazione servizio Web** > **Fine**.

4. Controllare il WSDL generato.

È stato richiesto di generare e salvare nel progetto il WSDL specifico del servizio.

a) In Enterprise Explorer navigator, aprire la cartella **W3CJMSEJB** > **ejbmodule** > **META - INF** > **wsdl**. Fare doppio clic su `StockQuoteAxis.wsdl` per aprirlo nell'editor WSDL.

Esaminare il binding; viene visualizzato l'URL JMS:

```
jms:jndi:requestaxis?jndiConnectionFactoryName=qm1&targetService=StockQuoteAxis
```

5. Passo facoltativo: collegare l'EJB a SOAP su HTTP utilizzando JAX - WS.

Fornendo due collegamenti all'EJB, i client possono scegliere i collegamenti SOAP per richiamare il servizio Web. Fornisce inoltre ai client i mezzi per interrogare il server Web per ottenere il relativo WSDL, utilizzando HTTP.

I passaggi per collegare un EJB a SOAP su HTTP non sono inclusi come parte dell'attivit ....

6. Implementare e ridistribuire `StockQuoteAxis` utilizzando l'esempio `StockQuoteAxis.java`

a) In Enterprise Explorer, aprire la cartella **W3CJMSEJB** > **Servizi** fare doppio clic `StockQuoteAxisService` per aprire la classe di implementazione in un editor Java.

b) Aprire il programma di esempio `StockQuoteAxis.java` nella cartella *`WebSphere MQ Installation directory`*`\tools\soap\samples\java\server` > Selezionare tutti i metodi, ma non il nome classe> **Copia**.

c) In `StockQuoteAxisSoapBindingImpl.java` selezionare tutti i metodi, ma non il nome classe, e incollare i metodi da `StockQuoteAxis.java`.

d) Aggiungere un'istruzione di stampa all'emissione nella console di WebSphere Application Server quando viene chiamato il servizio.
Modificare il metodo getQuote(String symbol):

```
public float getQuote(String symbol) {
    System.out.println("StockQuoteAxisSoapBindingImpl called with symbol: "
                       + symbol);
    return ((float) 55.25);
```

e) Correggere le importazioni: **Origine** > **Organizza importazioni** > **Salva**.

f) Correggere i tre errori perché l'implementazione non corrisponde all'interfaccia.

Gli errori sono dovuti a tre dei metodi in `StockQuoteAxis.java` che generano eccezioni e al WSDL per il servizio che non contiene alcun messaggio di errore. Il problema viene diagnosticato come una mancata corrispondenza tra le firme del metodo e le annotazioni del servizio Web del metodo.

Annotare i metodi con @WebFault e rigenerare il WSDL oppure mantenere invariata l'interfaccia e rimuovere le eccezioni.

Per mantenere la stessa interfaccia, rimuovere i tre `throws exception` dalle firme del metodo> **Salva**.

Operazioni successive

“Distribuzione su un client Axis2 utilizzando SOAP W3C su JMS” a pagina 1019

Attività correlate

Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse
Sviluppare un servizio Web Axis 1.4 da eseguire utilizzando WebSphere MQ come provider del servizio. Utilizzare il normale ambiente di sviluppo del servizio Web per creare un servizio per la distribuzione su Axis 1.4.

Sviluppo di un servizio .NET 1 o 2 per il trasporto WebSphere MQ per SOAP utilizzando Microsoft Visual Studio 2008
Sviluppa il servizio Web SampleStockQuote per .NET 1 o .NET 2 utilizzando Microsoft Visual Studio 2008

Sviluppo di client di servizi Web WebSphere MQ per il trasporto WebSphere MQ per SOAP

Utilizzare il normale ambiente di sviluppo per sviluppare client di servizi Web da utilizzare con il trasporto WebSphere MQ per SOAP.

Prima di iniziare

Crea il servizio. È possibile utilizzare uno degli esempi in “Sviluppo di servizi Web per il trasporto WebSphere MQ per SOAP” a pagina 962.

Effettuare delle scelte su come sviluppare, distribuire e utilizzare i client e su dove ottenere il WSDL per la generazione del client.

Decidere il proprio approccio allo sviluppo di client e servizi per il trasporto WebSphere MQ per SOAP.

Ci sono due approcci.

1. Utilizzare gli strumenti di sviluppo standard, sviluppare un servizio HTTP e un client, quindi utilizzare l'URL per il trasporto WebSphere MQ per SOAP.
2. Utilizzare gli strumenti e gli esempi forniti con il trasporto WebSphere MQ per SOAP.

Se si utilizza l'instradamento HTTP, è possibile eseguire il servizio su un server HTTP e anche eseguirlo utilizzando il trasporto WebSphere MQ per SOAP. Per eseguirla utilizzando il trasporto WebSphere MQ per SOAP, configurare il listener WebSphere MQ appropriato per SOAP e impostare i percorsi e i descrittori di distribuzione per eseguire il servizio. Gli strumenti forniti dal trasporto WebSphere MQ per SOAP effettuano la configurazione per conto dell'utente. In alternativa, è possibile configurare l'ambiente per eseguire i listener.

Gli strumenti forniti con il trasporto WebSphere MQ per SOAP sono utili per iniziare e imparare a distribuire il trasporto. Per il lavoro di produzione, ci sono vantaggi nell'utilizzo degli strumenti standard e nella distribuzione dello stesso servizio accessibile a diversi trasporti SOAP.

Decidere il tipo di client da sviluppare

È necessario decidere quale tipo di client del servizio Web sviluppare. La scelta dipende dal fatto che si conosca l'interfaccia del servizio e l'indirizzo del servizio.

Se l'interfaccia è nota, utilizzare gli strumenti Axis o .NET per generare le classi del client proxy dall'interfaccia del servizio. Le classi del client proxy rendono più semplice scrivere un client per richiamare il servizio. Se l'ubicazione del servizio è nota quando si sviluppa il client, utilizzare l'interfaccia proxy statica. Se l'ubicazione del servizio viene modificata, ad esempio se il servizio viene ridistribuito su un server di produzione, utilizzare l'interfaccia proxy dinamica.

Se l'interfaccia del servizio non è nota quando si sviluppa un client, su Axis, è possibile creare un client DII (Dynamic Invocation Interface) per Axis 1.4. Un client DII utilizza un'interfaccia generica

per richiamare qualsiasi servizio. Per passare correttamente i parametri a un particolare servizio, devi creare l'interfaccia del servizio specifica in modo programmatico. Creare l'interfaccia in modo programmatico nel client o caricando il WSDL per il servizio nel client. Su Axis2, è possibile creare un client di invio. Il client di invio utilizza un modello di documento per descrivere la richiesta del client, mentre un client DII utilizza un modello di chiamata. Entrambi lavorano sulla creazione dinamica della richiesta.

Ottenere il file WSDL per il servizio

Ad eccezione del caso dell'interfaccia del servizio creata in modo programmatico, è necessario prima ottenere il WSDL del servizio per creare un client del servizio Web. Il WSDL del servizio è ottenibile da tre origini diverse:

1. Direttamente dall'implementazione del servizio Web utilizzando uno strumento come **java2wsdl** (Axis) o **disco** (.NET).
2. Interrogando il servizio Web utilizzando l'URL: *Web service http url*?wsdl.
3. Da un file, su un file system o da un registro come UDDI o WebSphere Service Registry and Repository.

Nota: Se il servizio non è accessibile utilizzando HTTP, la query WSDL non funziona. Il servizio stesso potrebbe essere disponibile solo utilizzando il trasporto WebSphere MQ per SOAP.

Il WSDL generato da **amqwdeployWMQService** non è uguale al WSDL generato utilizzando **java2wsdl** o **disco**. Il WSDL generato è diverso da qualsiasi WSDL con cui è stato avviato per creare il servizio "Top Down". Su Axis, il descrittore di distribuzione server-config.wsdd associa il messaggio SOAP prodotto da un client a un'operazione e a un servizio. **amqwdeployWMQService** genera un descrittore di distribuzione differente da Eclipse.

Il WSDL utilizzato per creare i client dipende dalla modalità di distribuzione del servizio:

Distribuito utilizzando amqwdeployWMQService
Utilizzare il WSDL generato da **amqwdeployWMQService** . Specificare l'indicatore -w e selezionare WSDL rpcLiteral . Per compatibilità, è possibile selezionare rpcEncoded WSDL. rpcEncoded WSDL funziona solo con i client .NET e Axis 1.4 .

Distribuzione manuale mediante SimpleJavaListener
Utilizzare uno dei seguenti file WSDL:

1. WSDL utilizzato per definire il servizio o archiviato in un repository.
2. WSDL generato dal servizio da **java2wsdl** .
3. WSDL sottoposto a query utilizzando l'URL *Web service http url* ?wsdl, se disponibile da un server HTTP. È possibile eseguire uno strumento come Esplora servizi Web per importare la definizione del servizio direttamente in Eclipse.

Potrebbe essere necessario modificare l'URI per il servizio. Modificarlo dall'indirizzo del servizio HTTP all'URI per il trasporto WebSphere MQ per SOAP.

Distribuzione manuale utilizzando amqSOAPNETListener.
Utilizzare uno dei seguenti file WSDL:

1. WSDL utilizzato per definire il servizio o archiviato in un repository.
2. WSDL ottenuto dalla classe di servizio .NET (.asmx). utilizzando **disco**.
3. WSDL sottoposto a query utilizzando l'URL *Web service http url* ?wsdl, se disponibile. È possibile eseguire uno strumento come Esplora servizi Web per importare la definizione del servizio direttamente in Eclipse.
4. WSDL ottenuto eseguendo **amqswsdl** rispetto alla classe di servizi .NET (.asmx).

Potrebbe essere necessario modificare l'URI per il servizio. Modificarlo dall'indirizzo del servizio HTTP all'URI per il trasporto WebSphere MQ per SOAP.

Distribuito in Windows Communication Foundation
Ottenere il WSDL del servizio utilizzando l'URL *Web service http url*?wsdl. Il servizio deve essere definito con la configurazione del comportamento serviceMetaData come parte della definizione del servizio.

Distribuzione su una piattaforma server differente.
Seguire le istruzioni fornite con la piattaforma su come ottenere il WSDL del servizio corretto.

Informazioni su questa attività

Sviluppare i clienti utilizzando strumenti di sviluppo standard. Le seguenti attività illustrano come creare client per .NET 1 e 2, Axis 1.4 (JAX - RPC) e Axis2 (JAX - WS). Per Windows Communication Foundation, consultare i collegamenti delle attività correlate.

Sviluppo di un client JAX - RPC per il trasporto WebSphere per SOAP utilizzando Eclipse

Sviluppare un client del servizio Web Axis 1.4 da eseguire utilizzando il trasporto WebSphere MQ per SOAP.

Prima di iniziare

È necessario disporre del servizio disponibile. Se si sta seguendo l'attività come un esercizio pratico, utilizzare lo spazio di lavoro e il servizio creati nell'attività, “Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse” a pagina 963. È necessario disporre di un application server in esecuzione in Eclipse che supporti i servizi Web Axis 1.4 . In questa sezione viene utilizzato WebSphere Application Server Community Edition Versione 2.1.4. È configurato come parte dell'attività “Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse” a pagina 963. È anche possibile utilizzare Tomcat 6, che è un server delle applicazioni open source più piccolo.

Informazioni su questa attività

L'attività mostra lo sviluppo di tre tipi di client per il servizio Axis StockQuotedi esempio utilizzando Eclipse in esecuzione su Windows. I client sono un client statico e dinamico sviluppato utilizzando il client proxy e un client DII.

Vengono illustrati due approcci alternativi per la generazione dei proxy client da WSDL:

1. Generazione di proxy client utilizzando **amqwdeployWMQService** .
2. Importazione di WSDL in Eclipsee utilizzo della procedura guidata del servizio Web per generare i proxy client.

Procedura

1. Avviare Eclipse IDE per sviluppatori Java EE .
2. Creare un progetto Java denominato StockQuoteAxisClient:
   a) Passare alla prospettiva Java> **File** > **Nuovo** > **Progetto Java**. Nel campo **Project name** del tipo **Crea una pagina Progetto Java** , StockQuoteAxisEclipseClient. Assicurarsi che l'ambiente di esecuzione sia **J2SE1-1.4** o **J2SE-1.5** > **Avanti** .
   b) Nella pagina **Impostazioni Java** , selezionare la scheda **Librerie** > **Aggiungi JAR esterni ...**
   c) Passare a *MQ_INSTALLATION_PATH*/java/lib e selezionare tutti i file .jar > **Apri**.
   MQ_INSTALLATION_PATH è la directory in cui è installato WebSphere MQ .
   d) Passare a *MQ_INSTALLATION_PATH*/java/lib/soap e selezionare tutti i file .jar > **Apri**. È necessario avere installato axis.jar dal supporto di installazione di WebSphere MQ in questa directory.
   MQ_INSTALLATION_PATH è la directory in cui è installato WebSphere MQ .
   e) La scheda **Libreria** fa riferimento a tutti i file .jar necessari per creare il client> **Fine**.

3. Seguire uno di questi due approcci per creare proxy in Eclipse per il servizio Web dell'asse StockQuotedi esempio:

   - Generare i proxy client utilizzando **amqwdeployWMQService** .

     a. Creare il gestore code. Per l'attività creare QM1 come gestore code predefinito.

     b. Creare una directory di lavoro, samples. Copiare il programma di esempio StockQuoteAxis.java in samples/soap/server.

     c. Modificare amqwsetcp.cmd in *MQ_INSTALLATION_PATH*/bin per includere la directory corrente nel percorso classi. *MQ_INSTALLATION_PATH* è la directory in cui è installato WebSphere MQ .

     d. Aprire una finestra di comandi in samples ed eseguire il comando **amqwsetcp** modificato

     e. Creare WSDL per il servizio Axis StockQuoteeseguendo il comando,

     ```
     amqwdeployWMQService -f soap/server/StockQuoteAxis.java -c genAxisWsdl
     -u "jms:/queue?destination=REQUESTAXIS
     &initialContextFactory=com.ibm.mq.jms.Nojndi
     &connectionFactory=(connectQueueManager(QM1)binding(auto))"
     ```

     Attenzione: Utilizzare "/", invece di "." o "\" quando si utilizzano i comandi Java.

     Suggerimento: Invece di importare i proxy generati in Eclipse, è possibile importare il WSDL generato da .samples/generated. I proxy risultanti differiscono in due modi:

     i) I nomi dei package sono diversi - è possibile eseguire il refactoring.

     ii) I proxy generati da Eclipse includono una classe helper aggiuntiva, StockQuoteAxisProxy.java

     f. Creare i proxy client per il servizio Axis StockQuoteeseguendo il comando:

     ```
     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. Importare i proxy client in StockQuoteAxisClient:

     i) Fare clic con il tasto destro del mouse su **StockQuoteAxisClient\src** > Seleziona **File System** > **Avanti** > **Sfoglia ...** > trova la cartella .\samples\generated\client\remote\soap\server > **OK**.

     ii) Selezionare **server** nella pagina**Importa** > **Fine**.

     h. Eseguire il refactoring del nome package in soap.server.

     i) Fare clic con il pulsante destro del mouse sul pacchetto contenente i proxy client> **Refactor** > **Rinomina**. Immettere **New name**: soap.server > lascia i valori predefiniti selezionati per le altre scelte> **OK** . Tutti gli errori sono corretti.

   - Generare i proxy client utilizzando Eclipse.

     Si dispone di una scelta di modi per ottenere il file WSDL per il servizio. In questo esempio, il servizio è stato distribuito in Community Edition WebSphere Application Server e si ottiene il WSDL dal server Web. La distribuzione è descritta nell'attività "Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse" a pagina 963,

     a. In Eclipse, passare alla prospettiva Web e verificare che il server WebSphere Application Server Community Edition v2.1 sia in esecuzione e che l'asse StockQuotesia distribuito e sincronizzato.

     b. Importare il WSDL in Esplora servizi Web:

     i) Fare clic sull'icona **Esplora servizi Web** nella barra delle azioni oppure fare clic su **Esegui** > **Avvia Esplora servizi Web**.

     ii) Fare clic sull'icona della pagina WSDL in Esplora servizi Web per passare alla pagina WSDL.

     iii) Fare clic su **WSDL principale** nella finestra Navigator di Esplora servizi Web.

iv) Immettere l'URL del servizio Web, seguito da ?WSDL . L'URL per l'asse StockQuote, distribuito nell'attività “Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse” a pagina 963è:

```
http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl
```

c. Generare i proxy client:

i) Nel navigator Esplora servizi Web, fare clic su **http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl** .

ii) Nella finestra **Azioni** , fare clic su **Avvia wizard servizio Web** > lasciare selezionato **Client servizio Web** > **Vai**.

iii) Nella prima pagina della procedura guidata, fare clic sul link del progetto **Client** nella configurazione> Seleziona **StockQuoteAxisClient** progetto client> **OK**.

Suggerimento: La finestra della procedura guidata potrebbe perdere lo stato attivo. È necessario riportarlo a fuoco manualmente.

iv) Il runtime del servizio Web deve essere Apache Axis per generare un client JAX - RPC.

v) Fare clic su **Fine**.

vi) Modificare l'URL statico del servizio per puntare al trasporto WebSphere MQ per indirizzo SOAP per il servizio Axis StockQuote. È possibile scegliere di ignorare questo passo, fino a quando non viene eseguito il test del client con un server HTTP.

a) Aprire `StockQuoteAxisServiceLocator.java` e trovare la dichiarazione per `StockQuoteAxis_address`.

b) Modificare l'URL in

```
"jms:/queue?destination=REQUESTAXIS
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
&amp;connectionFactory=(connectQueueManager(QM1)binding(auto))"
```

Suggerimento: Eclipse trasforma automaticamente & in `&`, e viceversa, quando si copiano e incollano le stringhe nel codice `.java` .

d. Creare tre classi client Java, ciascuna con un metodo principale:

i) Crea pacchetto. Fare clic con il pulsante destro del mouse **StockQuoteAxisClient/src** > **Nuovo pacchetto**. Denominarlo `soap.client` > **Fine**.

ii) Selezionare **soap.client** > **Nuovo** > **Classe**. Denominare la classe `SQAStaticClient` > Check **public static void main (string [] args)** > Fine

iii) Ripetere la procedura per creare `SQADynamicClient.java` e `SQADIIClient.java`

e. Scrivere il codice client.

Figura 177 a pagina 982 tramite Figura 181 a pagina 984 forniscono esempi dei tre stili di codice client. Gli esempi utilizzano un URL HTTP per verificare il client utilizzando il servizio Axis StockQuotedistribuito su un server HTTP. Per eseguire i client rispetto al servizio Axis StockQuotedistribuito utilizzando il trasporto WebSphere MQ per SOAP, modificare l'URL in:

```
"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"
```

– Figura 177 a pagina 982 e Figura 179 a pagina 983 utilizzano il proxy generato da Eclipse, che dispone della classe helper Axisproxy StockQuoteaggiuntiva che semplifica la codifica.

– Figura 178 a pagina 983 e Figura 180 a pagina 983 utilizzano il proxy generato da **amqwdeployWMQService** .

– Figura 181 a pagina 984 non utilizza classi proxy.

Ogni client richiama `com.ibm.mq.soap.Register.extension()` per collegarsi al trasporto WebSphere MQ per SOAP. L'estensione è registrata nel descrittore distribuzione client. La distribuzione client su Axis 1.4 viene descritta in "Distribuzione di un client del servizio Web su Axis 1.4 per utilizzare il trasporto IBM WebSphere MQ per SOAP" a pagina 1014.

f. Eseguire i client inviando la richiesta SOAP all'asse StockQuoteospitato dal server WebSphere Application Server Community Edition configurato nello spazio di lavoro.

   i) Verificare che il server sia attivo, che l'asse StockQuotesia distribuito e sincronizzato.

   ii) Selezionare o aprire il client che si desidera verificare> Fare clic su **Esegui** nella barra delle azioni. In alternativa, fare clic sull'icona di esecuzione verde oppure fare clic sul client nel navigator> **Esegui come** > **Esegui configurazioni ...**. Configurare i parametri richiesti per eseguire il client.

g. Eseguire il client utilizzando il trasporto WebSphere MQ per SOAP.

   La procedura utilizza **amqwdeployWMQService** per distribuire il servizio e funziona solo con il client che utilizza il WSDL o i proxy creati da **amqwdeployWMQService** . Per eseguire il client utilizzando il WSDL originale o i proxy creati da Eclipse, distribuire il servizio con il relativo descrittore di distribuzione creato da Eclipse. Avviare manualmente **SimpleJavaListener** utilizzando il nome di bind della porta servizio come `targetServiceName` .

   i) Seguire le istruzioni in "Distribuzione di un servizio su Axis 1.4 da utilizzare per il trasporto WebSphere per SOAP utilizzando amqwdeployWMQService" a pagina 1003 per distribuire il servizio al listener SOAP Java semplice WebSphere MQ . La distribuzione del servizio funziona solo per il client che utilizza i proxy client o WSDL creati da **amqwdeployWMQService**.

   ii) In una finestra comandi, eseguire **amqwclientconfig** per creare il file descrittore di distribuzione client, `client-deploy.wsdd` .

   iii) Importare `client-deploy.wsdd` nella root del progetto Java che si desidera verificare utilizzando il trasporto WebSphere MQ per SOAP.

      a) Fare clic con il tasto destro del mouse sul progetto Java **StockQuoteAxisEclipseClient** > **Importa** > **File system** > **Avanti** > **Sfoglia ...**

      b) Passare alla directory contenente `client-deploy.wsdd` > **Apri** > Seleziona la directory nella pagina della procedura guidata **Importa** > controlla `client-deploy.wsdd` nel riquadro destro.

      c) Verificare che **Into folder:** abbia `StockQuoteAxisEclipseClient` immesso> **Fine**.

   iv) Confermare che la directory di lavoro per l'esecuzione di un'applicazione Java in questo progetto sia la directory `StockQuoteAxisEclipseClient` :

      Fare clic con il tastino destro del mouse sul progetto Java **StockQuoteAxisEclipseClient** > **Esegui come ....** > **Esegui configurazioni ...** > Selezionare la scheda **(x) = Argomenti** > Verificare che nella directory di lavoro il pulsante di opzione **Predefinito** sia selezionato e il percorso sia `StockQuoteAxisEclipseClient` . In alternativa, effettuare una delle seguenti scelte per selezionare un'altra ubicazione o un file contenente la configurazione del client:

      - Selezionare **Altro:** > digitare un percorso di directory a scelta.
      - Nella finestra **Argomenti VM** , immettere `-Daxis.ClientConfigFile=`*`full path to client deployment descriptor file`*

   v) Assicurarsi che l'URL sia configurato in modo da puntare al servizio distribuito utilizzando il trasporto WebSphere MQ per SOAP. Eseguire il client come descritto nel passo ii.

   Suggerimento: In genere, è possibile che si verifichi uno dei seguenti errori:

   i) `Exception: No client transport named 'jms' found!` .

   ii) Errore di connessione 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`

Spiegazioni:

i) `client-config.wsdd` non è stato trovato o include la riga `<transport name="jms" pivot="java:com.ibm.mq.soap.transport.jms.WMQSender"/>` in `client-config.wsdd`.

ii) È possibile che si sia verificato un problema del percorso build - non includendo i file .jar in *MQ_INSTALLATION_PATH*`/java/lib` . *MQ_INSTALLATION_PATH* è la directory in cui è installato WebSphere MQ .

iii) Problema di distribuzione del servizio, con `server-config-wsdd` o con i parametri passati a **SimpleSoapListener** .

iv) Mancata corrispondenza tra il descrittore di distribuzione e l'implementazione del servizio.

Se hai difficoltà ad eseguire il client in Eclipse, prova a utilizzare una finestra dei comandi:

i) Passare alla directory `StockQuoteAxisEclipseClient\bin` nella struttura di directory dello spazio di lavoro.

ii) Eseguire **amqwsetcp** e **amqwclientconfig** .

iii) Eseguire `java soap/client/SQAStaticClient`.

Client del servizio Web JAX - RPC di esempio

I client del servizio Web Java di esempio forniti con WebSphere MQ sono installati in *MQ_INSTALLATION_PATH* `\tools\soap\samples\java\clients`. *MQ_INSTALLATION_PATH* è la directory in cui è installato WebSphere MQ .

SQAxis2Axis.java
`SQAxis2Axis.java`, Figura 174 a pagina 981 , è un client proxy dinamico per richiamare il servizio `StockQuoteAxis` . È possibile sovrascrivere l'URL del servizio, che viene compilato nel proxy dinamico, fornendo un URL sulla riga comandi.

SQAxis2DotNet.java
`SQAxis2DotNet.java`, Figura 175 a pagina 981, è un client proxy dinamico per richiamare il servizio `StockQuoteDotNet` . È possibile sovrascrivere l'URL del servizio, che viene compilato nel proxy dinamico, fornendo un URL sulla riga comandi.

WsdlClient.java
`WsdlClient.java`, Figura 176 a pagina 982, è un client di richiamo dinamico per richiamare il servizio `StockQuoteDotNet` o `StockQuoteAxis` . Il client richiama il servizio `StockQuoteAxis` per impostazione predefinita. Aggiungi l'opzione della riga di comando -D richiama il servizio `StockQuoteDotNet` e -w per fornire una porta diversa a quella in `.\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);
      }
   }
```

Figura 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);
      }
   }
}
```

Figura 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);
      }
   }
}
```

Figura 176. WsdlClient.java

I client di esempio utilizzati in questa attività:

```
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);
    }
  }
}
```

Figura 177. Client statico che utilizza il proxy generato da Eclipse

```
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);
    }
  }
}
```

Figura 178. Client statico che utilizza il proxy generato 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);
    }
  }
}
```

Figura 179. Client dinamico che utilizza il proxy generato da 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);
    }
  }
}
```

Figura 180. Il client dinamico che utilizza il proxy generato amqwdeployWMQService

```
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);
    }
  }
}
```

Figura 181. Client DII (nessun proxy)

Attività correlate

Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse
Sviluppare un client del servizio Web Axis2 da eseguire utilizzando il trasporto WebSphere MQ per SOAP. Vengono elencati i client Axis2 di esempio forniti con il trasporto WebSphere MQ per SOAP e il comando **wsimport** utilizzato per generare i proxy.

Sviluppo di un client .NET 1 o 2 per il trasporto WebSphere per SOAP utilizzando Microsoft Visual Studio 2008
Sviluppare un client di servizi Web .NET 1 o 2 da eseguire utilizzando il trasporto WebSphere MQ per SOAP.

Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse

Sviluppare un client del servizio Web Axis2 da eseguire utilizzando il trasporto WebSphere MQ per SOAP. Vengono elencati i client Axis2 di esempio forniti con il trasporto WebSphere MQ per SOAP e il comando **wsimport** utilizzato per generare i proxy.

Prima di iniziare

Ottenere le librerie Axis2 e configurare un ambiente di sviluppo e di test per eseguire il client.

Nota: La denominazione delle versioni e release utilizzate da Axis crea confusione. Normalmente Axis 1.4 fa riferimento all'implementazione JAX-RPC e Axis2 all'implementazione JAX-WS.

Axis 1.4 è un livello di versione. Se si ricerca Axis 1.4 su Internet, si verrà indirizzati su http://ws.apache.org/axis/. Questa pagina contiene un elenco delle versioni precedenti di Axis (1.2 e 1.3) e, con data 22 aprile 2006, la release finale di Axis 1.4. Esistono release successive di Axis 1.4, che risolvono i bug, ma sono tutte raggruppate sotto il nome Axis 1.4. Si tratta di una di queste release di correzione dei bug fornite con WebSphere MQ. Per Axis 1.4, utilizzare la versione di `axis.jar` fornita con WebSphere MQ anziché quella ottenibile da http://ws.apache.org/axis/.

Il sito Web Axis utilizza anche Axis 1.1 per fare riferimento a tutte le versioni più generalmente note come Axis 1.4. Axis 1.2 fa invece riferimento ad Axis2.

Axis 1.5 non è una release successiva di Axis 1.4, bensì una release Axis2. Se si ricerca Axis 1.5, si verrà indirizzati su http://ws.apache.org/axis2/.https://ws.apache.org/axis2/download.cgi contiene un elenco delle versioni di release di Axis2, etichettate da 0.9 a 1.5.1 (e includendo, creando confusione, la versione 1.4). La versione di release di Axis2 da utilizzare con WebSphere MQ transport for SOAP è 1.4.1. Scaricare Axis2 1.4.1 da http://ws.apache.org/axis2/download/1_4_1/download.cgi.

È possibile scegliere di generare i proxy per i client del servizio Web per il trasporto WebSphere MQ per SOAP utilizzando **wsimport** o gli strumenti forniti con un IDE. Eclipse IDE per Java EE Developer 3.5 SR1

utilizza **wsdl2java**.**wsimport** viene fornito con Java 6. È possibile utilizzare Java 5 per eseguire i proxy client generati con **wsimport** o **wsdl2java**.

I client del servizio Web di esempio Axis2 forniti con il trasporto WebSphere MQ per SOAP sono stati sviluppati utilizzando **wsimport**; consultare "Client Axis2 di esempio" a pagina 990.

L'attività che segue dimostra come generare e utilizzare i proxy prodotti dalla procedura guidata dei servizi Web fornita con Eclipse IDE per gli sviluppatori Java EE . I client di esempio mostrano come utilizzare i proxy prodotti da **wsimport**.

Per utilizzare la procedura guidata dei servizi Web, è necessario aggiungere un server delle applicazioni che supporti Axis2 al workbench. La procedura mostra come configurare WASCE per supportare Axis2 utilizzando il workbench.

1. Configurare il server delle applicazioni utilizzato in Eclipse IDE per Java EE Developers per supportare Axis2. In questo esempio, configurare il server delle applicazioni WASCE 2.1.4, che fa parte dell'area di lavoro creata in "Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse" a pagina 963.
   a. Aprire le preferenze dello spazio di lavoro per configurare il server: Aprire **Finestra** > **Preferenze**.
   b. Verificare che JRE installato sia `Java50`: fare clic su **JRE installati**.
   c. Aggiungere WASCE come server: fare clic su **Server** > **Ambienti di runtime** > **Aggiungi** > **IBM** > **WASCE v2.1** > Avanti. Il JRE deve essere Java50 > Vai alla directory di installazione WASCE> **OK** > **Fine**. È necessario aver installato il plugin WASCE per Eclipse Java EE IDE for Web Developers.
   d. Aggiungere Axis2: fare clic su **Servizi Web** > **Axis2 Preferenze**. Nella scheda **Axis2 Runtime** > **Sfoglia** Aprire la directory contenente molti file jar `Axis2` > **Apply**.
   e. Associare WASCE a Axis2: Fare clic su **Servizi Web** > **Server e runtime**. In **Server** selezionare **IBM WASCE v2.1 server**e in **Web service runtime**, selezionare **Apache Axis2** > **Applica** > **OK**
   f. Avviare il server: aprire la prospettiva Web e la vista Server. Fare clic con il pulsante destro del mouse nella vista Server> **Nuovo** > **Server**. **IBM WASCE v2.1 Server** è selezionato e configurato> **Fine**. Avviare il server.
2. Verificare di aver distribuito il servizio Axis StockQuotea WASCE per eseguire la procedura guidata del servizio Web.
3. Per verificare il servizio con il trasporto WebSphere MQ per il servizio SOAP, distribuire il servizio a un trasporto WebSphere MQ per il listener SOAP per Axis 1.4; consultare "Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse" a pagina 963.

Informazioni su questa attività

Gli sviluppatori Eclipse IDE per Java EE utilizzano Java50 e la procedura guidata dei servizi Web per generare le classi proxy per il servizio. Le classi proxy sono differenti dalle classi create dallo strumento **wsimport** fornito con Java 6. Un approccio alternativo consiste nel creare le classi proxy utilizzando **wsimport** e importare i package creati in Eclipse Java EE IDE for Web Developers.

La procedura guidata dei servizi Web in Eclipse IDE per gli sviluppatori Java EE crea un client del servizio Web in un progetto Web. È possibile eseguire il client come un'applicazione Java semplice; non richiede un application server. È anche possibile trasferire il codice a un progetto Java e configurare il percorso build per includere i file JAR Axis2 .

Procedura

1. Creare un progetto Web in un nuovo progetto Enterprise:
   a) Con nessun elemento selezionato in Esplora progetti> Fare clic con il tasto destro del mouse sullo spazio vuoto> **Nuovo** > **Progetto applicazione enterprise** > denominalo `StockQuoteAxis2EAR` > **Fine**. Rispondere No alla finestra fornendo l'opzione di apertura della prospettiva Java EE .

   I valori predefiniti sono impostati per utilizzare WASCE.

b) Fare clic con il pulsante destro del mouse su StockQuoteAxis2EAR > **Nuovo** > **Progetto Web dinamico**. Denominare il progetto StockQuoteAxis2WebClient > Selezionare la casella di appartenenza EAR per aggiungere il progetto a **StockQuoteAxis2EAR**. WASCE 2.1 è selezionato come runtime di destinazione.

c) Nella sezione Configurazione della pagina **Nuovo progetto Web dinamico** > **Modifica ...** > Controllare il facet del progetto dei servizi Web Axis2 . **Dynamic Web Module 2.5**, **Java 6.0**e **WASCE deployment 1.2** sono già controllati. > **OK** > **Fine**. Rispondere No alla finestra fornendo l'opzione di apertura della prospettiva Java EE .

2. Importare il WSDL per il servizio nello spazio di lavoro e generare il proxy client:

   In questo esempio, il documento WSDL contiene il collegamento del servizio HTTP e diventa la destinazione per il proxy del client Web statico. È possibile modificare l'URL nel binding del servizio Web per puntare al trasporto WebSphere MQ per l'URL SOAP prima di generare il proxy client. Il proxy del client Web statico è quindi il servizio distribuito nel trasporto WebSphere MQ per SOAP.

   a) Avviare Esplora servizi Web: utilizzare l'icona nella barra delle azioni oppure **Esegui** > **Avviare Esplora servizi Web**.

   b) Selezionare Esplora WSDL facendo clic sull'icona WSDL nella finestra **Esplora servizi Web** > Fare clic su **WSDL principale** nella finestra Navigator > Immettere l'URL del file WSDL dell'asse StockQuote> **Vai**.
   In questo esempio, ottenere WSDL direttamente dal servizio HTTP: `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl`

   c) In Navigator, fare clic sulla riga con l'URL del servizio Web. Nella finestra **Azioni** , selezionare **Importa WSDL nel workbench** > Selezionare **StockQuoteAxis2WebClient** come **Progetto workbench** > Immettere il **nome file WSDL**, `StockQuoteAxisHTTP.wsdl` > **Vai**.

   d) fare clic con il tasto destro del mouse su **StockQuoteAxisHTTP.wsdl** > **Servizi Web** > **Genera client**. Verificare che le informazioni di configurazione sulla pagina dei servizi Web della procedura guidata sono le seguenti: Server: IBM WASCE v2.1 Server, runtime del servizio Web: Apache Axis2, progetto client: StockQuoteAxis2WebClient, progetto EAR client: StockQuoteAxisEAR. Per correggere la configurazione, fare clic sulle righe errate.

   e) Fare clic su **Avanti** > verificare le impostazioni di creazione del codice> **Fine**.

      Notare che viene creato un nuovo package, `soap.server`, che contiene i proxy richiesti.

3. Configurare il progetto per eseguire il trasporto WebSphere MQ per SOAP come trasporto JMS.

   WebSphere MQ transport for SOAP fornisce un `transportSender`, ma nessun `transportReceiver`. In altre parole, il trasporto WebSphere MQ per SOAP supporta i client Axis2 . Attualmente non supporta i servizi Axis2 .

   a) Nel progetto **StockQuoteAxis2WebClient** , fare clic con il pulsante destro del mouse su `WebContent\WEB-INF\conf\axis2.xml` > **Apri con ...** > **editor XML**.

   b) Ricercare `transportSender` (verso la fine del file) e trovare il JMS commentato `transportSender` > Fare clic con il tastino destro del mouse sulla linea> **Aggiungi prima ...** > **transportSender**.

   c) Fare clic con il pulsante destro del mouse su **transportSender** > **Aggiungi attributo** > **Nome** > Fare clic con il pulsante destro del mouse su **transportSender** > **Aggiungi attributo** > **Classe**.

   d) Fare clic con il pulsante destro del mouse su **Nome** > **Modifica attributo** > Immettere il **Valore** `jms`

   e) Fare clic con il tastino destro del mouse su **Classe** > **Modifica attributo** > Immettere il **Valore:** `com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender`.> Salva.

   f) Aggiungere `com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender` al percorso build: fare clic con il tasto destro del mouse su **StockQuoteAxis2WebClient** > **Percorso build** > **Configura percorso build ...** > Fare clic sulla scheda **Librerie** > **Aggiungi JAR esterni ...**. Selezionare tutti i JAR in *MQ_INSTALLATION_PATH*`\java\lib` > **OK**.

      MQ_INSTALLATION_PATH è la directory in cui è installato WebSphere MQ .

4. Creare un client statico sincrono, verificarlo utilizzando HTTP, quindi convertire il proxy per eseguire il client statico utilizzando il trasporto WebSphere MQ per SOAP.

a) Fare clic con il tasto destro del mouse su **Risorse Java: src** > **Nuovo** > **Pacchetto** > Denominazione del pacchetto `soap.client` > Fine

b) Fare clic con il pulsante destro del mouse su **soap.client** > **Nuovo** > **Classe** > Denominazione della classe `SQA2StaticClient` > **Fine**.

c) Sostituire la classe con il seguente codice, quindi fare clic su **Salva**.

Figura 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. Verificare il client con il servizio Axis StockQuotedistribuito in WASCE e con il trasporto WebSphere MQ per SOAP.

   a) In Esplora progetti, fare clic con il pulsante destro del mouse su **SQA2StaticClient** > **Esegui come** > **Applicazione Java**.

   Il risultato, `Response is 55.25`, viene visualizzato nella vista Console. È anche possibile selezionare la finestra della console WASCE nella vista Console e visualizzare l'emissione sul server WASCE, `StockQuoteAxis called with parameter: ibm`.

   b) Il proxy è stato creato con l'indirizzo di servizio, `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis`, quindi il client statico richiama il servizio in esecuzione su HTTP. È possibile modificare il client statico per richiamare il servizio utilizzando il trasporto WebSphere MQ per SOAP. Le seguenti istruzioni modificano l'indirizzo di servizio in `StockQuoteAxisServiceStub.java` senza ricreare il proxy e configurano `SQA2StaticClient` parametri di runtime da caricare `axis2.xml`. `axis2.xml` configura Axis2 per utilizzare il trasporto WebSphere MQ per SOAP.

   c) Aprire `StockQuoteAxisServiceStub.java` >
   Sostituire le due ricorrenze di `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis` con:

```
jms:/queue?destination=REQUESTAXIS@QM1
&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
&targetService=StockQuoteAxis
&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
```

   d) Se si esegue `SQA2StaticClient` ora, viene generata un'eccezione perché non è stato trovato un `transportSender` configurato per JMS
   L'eccezione è:

```
Exception: null java.lang.NullPointerException at
soap.server.StockQuoteAxisServiceStub.getQuote(StockQuoteAxisServiceStub.java:547)
at soap.client.SQA2StaticClient.main(SQA2StaticClient.java:11)
```

   e) In Explorer progetti, fare clic con il pulsante destro del mouse su **SQA2StaticClient** > **Esegui come ...** > **Esegui configurazioni ...**. Passare al separatore **(x) = Argomenti** e nell'area di immissione **Argomenti VM** , immettere il percorso del file `axis2.conf` > **Applica** > **Esegui**.

L'argomento VM è: `-Daxis2.xml=${workspace_loc:StockQuoteAxis2WebClient/WebContent/WEB-INF/conf}/axis2.xml`. Oppure è possibile fornire un percorso standard al file di configurazione Axis2 .

f) Eseguire nuovamente `SQA2StaticClient` . In questa esecuzione, si utilizza il trasporto WebSphere MQ per SOAP. Confermare verificando che non vi sia alcun nuovo output nella console WASCE. Aprire la console o la finestra dei comandi associata al listener SimpleJavae l'output è `StockQuoteAxis called with parameter: ibm`.

6. Creare un client dinamico per il trasporto HTTP e WebSphere MQ per SOAP e verificarlo.

a) Fare clic con il pulsante destro del mouse su **soap.client** > **Nuovo** > **Classe** > Denominazione della classe `SQA2DynamicClient` > **Fine**.

b) Sostituire la classe con il seguente codice, quindi fare clic su **Salva**.

```
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) Creare una configurazione di esecuzione per `SQA2DynamicClient.javae` aggiungere il percorso a `axis2.xml`: `-Daxis2.xml=${workspace_loc:StockQuoteAxis2WebClient/WebContent/WEB-INF/conf}/axis2.xml`

d) Eseguire `SQA2DynamicClient`. Controllare l'output della console per `SQA2DynamicClient`, WASCE e **`SimpleJavaListener`**.

7. Creare un client asincrono e accedere al risultato in un gestore callback e nel thread del programma principale.

I proxy client asincroni creati dalla procedura guidata dei servizi Web per Eclipse Java EE IDE for Web Developers differiscono dai proxy creati da **`wsimport`**. I proxy **`wsimport`** utilizzano tipi generici `Future`, `Responsee` `AsyncHandler` .

La procedura guidata del servizio Web per Eclipse Java EE IDE for Web Developers crea una classe astratta `StockQuoteAxisServiceCallbackHandler` . È necessario estendere `StockQuoteAxisServiceCallbackHandler` e creare un gestore callback.

a) Fare clic con il pulsante destro del mouse su **soap.client** > **Nuovo** > **Classe** > Denominazione della classe `SQA2CallbackHandler` > **Fine**.

b) Sostituire la classe con il codice seguente.

```
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) Fare clic con il pulsante destro del mouse su **soap.client** > **Nuovo** > **Classe** > Denominazione della classe SQA2AsyncClient > **Fine**.

d) Sostituire la classe con il codice seguente.

Figura 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();
     }
   }
 }
```

L'output della console è il seguente:

```
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
```

Client Axis2 di esempio

I proxy di esempio vengono generati utilizzando lo strumento **wsimport** fornito con Java 6. Sono forniti sei campioni:

1. DynamicProxyClientSync.java
2. DynamicProxyClientAsyncPolling.java
3. DynamicProxyClientAsyncCallback.java
4. DispatchClientSync.java
5. DispatchClientAsyncPolling.java
6. DispatchClientAsyncCallback.java

Gli esempi client vengono generati per il server dell'asse StockQuotedi esempio. Generare il WSDL con il comando **amqwdepoyWMQServer** , specificando lo switch -w per selezionare lo stile `rpcLiteral` . Utilizzare il seguente comando per generare i proxy per gli esempi:

```
wsimport soap.server.StockQuoteAxis_Wmq.wsdl -d generated -keep -p com.ibm.mq.axis2.samples
```

Figura 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
  }
}
```

Figura 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
  }
}
```

Figura 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
  }

}
```

Figura 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
  }
}
```

Figura 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
  }
}
```

Figura 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
  }

}
```

Attività correlate

Sviluppo di un client JAX - RPC per il trasporto WebSphere per SOAP utilizzando Eclipse
Sviluppare un client del servizio Web Axis 1.4 da eseguire utilizzando il trasporto WebSphere MQ per SOAP.

Sviluppo di un client .NET 1 o 2 per il trasporto WebSphere per SOAP utilizzando Microsoft Visual Studio 2008
Sviluppare un client di servizi Web .NET 1 o 2 da eseguire utilizzando il trasporto WebSphere MQ per SOAP.

Sviluppo di un client .NET 1 o 2 per il trasporto WebSphere per SOAP utilizzando Microsoft Visual Studio 2008

Sviluppare un client di servizi Web .NET 1 o 2 da eseguire utilizzando il trasporto WebSphere MQ per SOAP.

Prima di iniziare

È possibile avviare lo sviluppo di un client .NET 1 o 2 in diversi modi:

1. Utilizzare **amqwdeployWMQService** per generare stub client da un servizio Web e importarli in Visual Studio.
2. Utilizzare **java2wsdl** per creare WSDL da un'implementazione Java di un servizio Web, quindi utilizzare `wsdl.exe`, fornito con .NET, per generare stub client.
3. Generare WSDL da un'implementazione .NET `.asmx` del servizio utilizzando **amqswsdl**, quindi utilizzare `wsdl.exe`.
4. Se è stato sviluppato e distribuito il servizio per HTTP, utilizzare **Aggiungi riferimento Web ...** in Visual Studio per configurare il client per accedere al servizio HTTP. Modificare l'URL che fa riferimento al servizio distribuito al trasporto WebSphere MQ per SOAP.

L'attività utilizza il servizio sviluppato in “Sviluppo di un servizio .NET 1 o 2 per il trasporto WebSphere MQ per SOAP utilizzando Microsoft Visual Studio 2008” a pagina 967.

Informazioni su questa attività

Seguire questa procedura per creare un client .NET 1 o 2 per HTTP e un trasporto WebSphere MQ per SOAP.

Procedura

1. Creare l'applicazione della console client e modificarla per richiamare il servizio Web HTTP StockQuote .
   a) Fare clic con il pulsante destro del mouse su **Soluzione 'StockQuoteDotNet'** in **Esplora soluzioni** > Aggiungi ...> Nuovo progetto. Selezionare il **Tipo di progetto C# .NET Framework 2.0**e **Console Application**. Denominare il progetto `StockQuoteClientDotNet` > **OK**
   b) Fare clic con il pulsante destro del mouse su **Soluzione 'StockQuoteDotNet'** in **Esplora soluzioni** > Aggiungi ...> Nuovo progetto. Selezionare il **Tipo di progetto C# .NET Framework 2.0**e **Console Application**. Denominare il progetto `StockQuoteClientDotNet` > **OK**
   c) Fare clic con il pulsante destro del mouse su **StockQuoteClientDotNet** > **Imposta come progetto di avvio**.
   d) Fare clic con il pulsante destro del mouse su **StockQuoteClientDotNet** > **Aggiungi riferimento Web ...** > Sfoglia servizi Web in questa soluzione> Seleziona **StockQuote** > **Aggiungi riferimento**.

Notare di aver aggiunto un riferimento Web all'host locale e un nuovo file di configurazione `app.config`.

e) In Esplora soluzioni, modificare il nome dell'applicazione della console da `Program.cs` a `StockQuoteClientDotNet.cs` > Fare clic su **OK** per modificare tutti gli usi di `Program.cs` in `StockQuoteClientDotNet.cs`.

f) Sostituire il contenuto di `StockQuoteClientDotNet.cs` con il codice in Figura 190 a pagina 999.

```
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();
    }
  }
}
```

Figura 190. Programma HTTP StockQuoteClientDotNet

g) Avviare StockQuoteClientDotNet per verificare il servizio `StockQuote.asmx` :

i) Premere **F5**, fare clic sulla freccia verde nella barra delle azioni oppure su **Debug** > **Avvia debug (F5)**.

Se il progettoDOtNet StockQuotesi trova nella stessa soluzione, viene avviato automaticamente. In caso contrario, è necessario avviare prima il servizio.

La finestra comandi con i risultati si apre dietro l'area di lavoro. L'istruzione `Console.ReadLine();` impedisce la chiusura fino a quando non si preme **Invio**.

Suggerimento: Verificare che `StockQuote.asmx` sia la pagina iniziale nel progetto `StockQuoteDotNet` .

2. Modificare StockQuoteClientDotNet per chiamare il servizio `StockQuote.asmx` utilizzando WebSphere MQ transport for SOAP.

a) Aggiungere le righe visualizzate in grassetto al client.

```
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();
    }
  }
}
```

Figura 191. Programma StockQuoteClientDotNet modificato

In alternativa, modificare l'URL predefinito. Aprire **StockQuoteClientDotNet** > **Properties** > **Settings.settings** e modificare il valore della proprietà `StockQuoteClientDotNet_localhost_StockQuote` nel trasporto WebSphere MQ per l'URL SOAP.

b) Aggiungere un riferimento a `amqsoap.dll`

i) Nel progetto **StockQuoteClientDotNet** in **Esplora soluzioni**, fare clic con il pulsante destro del mouse su **Riferimenti** > **Aggiungi riferimento ...** > fare clic sulla scheda **Sfoglia** > Sfoglia *MQ_INSTALLATION_PATH*\bin > Seleziona **amqsoap.dll** > **OK**. *MQ_INSTALLATION_PATH* è la directory in cui è installato WebSphere MQ .

3. Verifica il client con il servizio `StockQuote.asmx` utilizzando il trasporto WebSphere MQ per SOAP.

a) Aprire una finestra comandi nella directory del progetto StockQuoteDotNet : `.\StockQuoteDotNet\StockQuoteDotNet` > verificare che `.bin\StockQuoteDotNet.dll` esista. In caso contrario, ricreare la soluzione.

b) Immettere il comando **amqwRegisterdotNet**.

È necessario eseguire **amqwRegisterdotNet** una sola volta per installazione.

c) Se è stato eseguito **amqwdeployWMQServer** con `genAsmxWMQBits`, eseguire il listener SOAP .NET:
`generated\server\startWMQNListener`

d) In alternativa, eseguire direttamente il listener:

```
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. In Visual Studio 2008, premere **F5** per eseguire StockQuoteClientDotNet.

Client di servizi Web .NET Framework 1 e .NET Framework 2

I client .NET di esempio forniti con il trasporto WebSphere MQ per SOAP utilizzano gli stub generati per chiamare i servizi Axis e .NET di esempio.

Per i client .NET Framework 1 e .NET Framework 2 WebSphere MQ fornisce l'accesso ai servizi Web utilizzando i client .NET. Il comando **amqwdeployWMQService** ha un'opzione, `genProxiestoDotNet`,

che genera stub client .NET Framework 1 o .NET Framework 2 per un servizio Web. È inoltre possibile utilizzare gli stub client generati dallo strumento .NET **wsdl** o da Microsoft Visual Studio 2005 o 2008.

I client del servizio Web .NET Framework 1 e .NET di esempio sono installati in *MQ_INSTALLATION_PATH*\tools\soap\samples\dotnet. *MQ_INSTALLATION_PATH* è la directory in cui è installato WebSphere MQ .

SQVB2Axis.vb
SQVB2Axis.vb, Figura 192 a pagina 1001, è il client Visual Basic per richiamare il servizio **StockQuoteAxisService** .

SQVB2DotNet.vb
QVB2DotNet.vb, Figura 193 a pagina 1001, è il client Visual Basic per richiamare il servizio **StockQuoteDotNet** .

SQCS2Axis.cs
SQCS2Axis.cs, Figura 194 a pagina 1002, è il client C# per richiamare il servizio **StockQuoteAxisService** . Puoi sovrascrivere l'URL del servizio fornendo un URL sulla riga di comando.

SQCS2DotNet.cs
SQCS2DotNet.cs, Figura 195 a pagina 1002, è il client C# per richiamare il servizio **StockQuoteDotNet** . Puoi sovrascrivere l'URL del servizio fornendo un URL sulla riga di comando.

```
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
```

Figura 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
```

Figura 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());
      }
   }
}
```

Figura 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());
      }
   }
}
```

Figura 195. SQCS2DotNet

Attività correlate

Sviluppo di un client JAX - RPC per il trasporto WebSphere per SOAP utilizzando Eclipse
Sviluppare un client del servizio Web Axis 1.4 da eseguire utilizzando il trasporto WebSphere MQ per SOAP.

Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse
Sviluppare un client del servizio Web Axis2 da eseguire utilizzando il trasporto WebSphere MQ per SOAP. Vengono elencati i client Axis2 di esempio forniti con il trasporto WebSphere MQ per SOAP e il comando **wsimport** utilizzato per generare i proxy.

Distribuzione di servizi Web mediante il trasporto WebSphere MQ per SOAP

Distribuire un servizio Web a uno dei diversi ambienti server e connettersi ad esso utilizzando il trasporto WebSphere MQ per SOAP.

Prima di iniziare

Sviluppare un Web service e verificarlo utilizzando SOAP su HTTP nell'ambiente di destinazione.

Informazioni su questa attività

È possibile distribuire un servizio Web da eseguire con il trasporto WebSphere MQ per SOAP in diversi ambienti di runtime SOAP. È possibile distribuire un servizio ad Axis 1.4 utilizzando solo il software installato con WebSphere MQ. Per gli altri ambienti di runtime, è necessario installare software aggiuntivo.

L'utente non è limitato all'esecuzione del trasporto WebSphere MQ per SOAP sui server per cui sono presenti istruzioni di distribuzione. Utilizzare le istruzioni per distribuire un servizio a uno degli ambienti elencati.

Nota: Alcuni ambienti integrati offrono SOAP su JMS utilizzando il bind JMS SOAP consigliato da W3C e il trasporto WebSphere MQ per il bind SOAP. I rilasci di WebSphere MQ, fino a 7.0.1.2incluso, supportano solo il trasporto WebSphere MQ per il bind SOAP. A partire da 7.0.1.3 , è possibile distribuire i client Axis2 utilizzando un URI conforme al suggerimento candidato W3C per SOAP su JMS. Consultare l'esercitazione, Develop a SOAP/JMS JAX - WS Web services application with WebSphere Application Server V7 e Rational Application Developer V7.5.

Distribuzione di un servizio su Axis 1.4 da utilizzare per il trasporto WebSphere per SOAP utilizzando amqwdeployWMQService

Distribuire un servizio Axis 1.4 al trasporto WebSphere MQ per SOAP creando una directory di distribuzione, eseguendo il comando **amqwdeployWMQService** e avviando il listener Axis 1.4 .

Prima di iniziare

1. Seguire le istruzioni per installare il trasporto WebSphere MQ per SOAP
2. Verificare l'installazione e l'ambiente utilizzando il comando **runivt** .
3. Per ridistribuire un servizio:
   a. Eliminare la sottodirectory `./generated` e tutte le relative sottodirectory.
   b. Rimuovere le richieste dalla coda di destinazione ed eliminarle.
   c. Procedere con le istruzioni del passo “2” a pagina 1003.

Informazioni su questa attività

Queste istruzioni sono per distribuire un servizio Axis 1.4 per la prima volta. Per riavviare un servizio Axis 1.4 , rieseguire il listener SOAP Axis 1.4 : passo “11” a pagina 1004.

Utilizzare le istruzioni riportate di seguito per distribuire un nuovo servizio Axis 1.4 al trasporto WebSphere MQ per SOAP:

Procedura

1. Creare una directory *deployDir* per contenere i file di distribuzione.

   Il programma di utilità di installazione richiede che ogni servizio venga distribuito da una directory separata.
2. Aprire una finestra di comandi su Windowso una shell di comandi utilizzando X Window System su sistemi UNIX and Linux , in *deployDir* per eseguire **amqwdeployWMQService**.
3. Eseguire **amqwsetcp** per impostare il percorso classi.

   JRE e JDK devono trovarsi nel percorso classi, alla versione 5.0 o successiva e allo stesso livello di versione.
4. Copiare l'origine della classe, *className*`.java`, in *deployDir*
5. Copiare tutti i file di origine Java nello stesso package di *className* in *deployDir*`/`*packageName*, dove *packageName* è una struttura di directory corrispondente al nome del package.
6. Esegui **javac** *packageName*`.`*className*`.`

   Potrebbe essere necessario aggiungere un percorso alla directory corrente "`.`"o alla directory *packageName* per **javac** per trovare le altre classi.

7. Creare il WSDL dell'asse per il servizio:

```
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. Crea le risorse WebSphere MQ per il servizio:

```
amqwdeployWMQService -f packageName.className.java -c genAxisWMQBits
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

Suggerimento:

Se si desidera configurare un nuovo gestore code e le risorse di cui ha bisogno per eseguire lo sviluppo e la verifica, eseguire **setupWMQSOAP**.

Se si desidera impostare il nuovo gestore code come predefinito, eseguire una copia di **setupWMQSOAP** dalla directory *WMQ install directory*\tools\soap\samples e aggiungere il parametro -q alla riga

```
call :try -q crtmqm %QMGR%
```

9. Creare il listener Axis e distribuire il servizio:

```
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. Se è necessario generare il WSDL per il servizio, generare stub client o proxy client, eseguire **amqwdeployWMQService** con uno dei seguenti parametri:

- genAsmxWsdl
- genAxisWsdl
- genProxiesToDotNet
- genProxiestoAsse

Nota: È necessario generare WSDL prima di generare i proxy. L'opzione AllAxis ha esito negativo se CLASSPATH non è impostato per trovare tutte le classi importate per compilare *className*.java. Se ci sono più file Java nel pacchetto contenente *className*.java, è necessario compilarli prima utilizzando **javac**. **amqwdeployWMQService** -f *packageName.className*.java -c CompileJava compila solo *className*.java.

11. Avviare il listener dell'asse generato.

```
.\generated\server\startWMQJListener.cmd
```

Attività correlate

Distribuzione di un servizio al servizio .NET Framework 1 o 2 per utilizzare il trasporto WebSphere MQ per SOAP

Distribuire un servizio .NET Framework 1 o 2 al trasporto WebSphere MQ per SOAP. Creare una directory di distribuzione, eseguire il comando **amqwdeployWMQService** e avviare il listener .NET.

Distribuzione di un servizio in CICS Transaction Server per l'utilizzo di WebSphere Transport per SOAP

Il trasporto WebSphere MQ per SOAP è integrato nel supporto dei servizi Web CICS Transaction Server 4.1 .

Distribuzione di un servizio in WebSphere Application Server per utilizzare WebSphere Transport for SOAP

Il trasporto WebSphere MQ per SOAP è integrato nel SIB (Service Integration Bus) su WebSphere Application Server.

Configurazione di WebSphere Application Server per utilizzare SOAP W3C su JMS

Un servizio Web collegato alla raccomandazione del candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server delle applicazioni Java EE . Questa attività è il passo 1 di connessione di un client del servizio Web Axis2 e di un servizio Web distribuito sul server delle applicazioni WebSphere mediante il protocollo SOAP over JMS W3C . Configurare le risorse WebSphere MQ e WebSphere Application Server per sviluppare e distribuire il servizio Web collegato a W3C SOAP su JMS come trasporto.

Distribuzione di un servizio all'endpoint del servizio WebSphere ESB e Process Server per utilizzare WebSphere Transport for SOAP
Il trasporto WebSphere MQ per SOAP non è direttamente supportato da WebSphere ESB e Process Server. È necessario configurare un'esportazione personalizzata.

Distribuzione di un servizio al servizio .NET Framework 1 o 2 per utilizzare il trasporto WebSphere MQ per SOAP

Distribuire un servizio .NET Framework 1 o 2 al trasporto WebSphere MQ per SOAP. Creare una directory di distribuzione, eseguire il comando **amqwdeployWMQService** e avviare il listener .NET.

Prima di iniziare

1. Seguire le istruzioni per installare il trasporto WebSphere MQ per SOAP
2. Verificare l'installazione e l'ambiente utilizzando il comando **runivt** .
3. È necessario impostare il percorso dei file .NET framework `wsdl.exe` e `csc.exe` . Le copie di `wsdl.exe` e `csc.exe` identificate dalla variabile PATH devono essere allo stesso livello di .NET Framework. Se sono installati più framework .NET o si sta utilizzando Visual Studio, controllare attentamente la variabile PATH .
4. Per ridistribuire un servizio:
   a. Eliminare la sottodirectory `./generated` e tutte le relative sottodirectory
   b. Rimuovere le richieste dalla coda di destinazione ed eliminarle.
   c. Procedere con le istruzioni del passo “2” a pagina 1005.

Informazioni su questa attività

Queste istruzioni sono per distribuire un servizio .NET per la prima volta. Per riavviare un servizio .NET, eseguire nuovamente il listener SOAP .NET, passo “9” a pagina 1006.

Utilizzare le seguenti istruzioni per distribuire un nuovo servizio .NET Framework 1 o .NET Framework 2 nel trasporto WebSphere MQ per SOAP:

Procedura

1. Creare una directory *deployDir* per contenere i file di distribuzione.
   Il programma di utilità di installazione richiede che ogni servizio venga distribuito da una directory separata.
2. Aprire una finestra comandi in *deployDir* per eseguire **amqwdeployWMQService**.

```
C:\IBM\ID\QuoteClient>
```

3. Eseguire **amqwsetcp** per impostare il percorso classi.
   Un percorso classi è necessario solo per client Axis.
4. Copiare il servizio .NET, *className*`.asmx`, in *deployDir*
5. Creare l'implementazione del servizio in una libreria (`.dll`).
   L'implementazione del servizio in linea si trova in *className*`.asmx`. L'implementazione del servizio code - behind potrebbe essere *className*`.asmx.cs`.

Figura 196 a pagina 1006 mostra un esempio di un comando per creare un servizio V2 di .NET Framework come libreria.

```
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
```

Figura 196. Comando di build per il servizio V2 .NET Framework

6. Copiare *className*.dll in *deployDir**bin*.
7. Imposta le risorse WebSphere MQ e crea il listener richiesto per il servizio:

```
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. Se è necessario generare il WSDL per il servizio, generare stub client o proxy client, eseguire **amqwdeployWMQService** con uno dei seguenti parametri:
   - genAsmxWsdl
   - genAxisWsdl
   - genProxiesToDotNet
   - genProxiestoAsse

   Nota: È necessario generare WSDL prima di generare i proxy.
9. Avviare il listener .NET generato.

```
.\generated\server\startWMQNListener.cmd
```

Attività correlate

Distribuzione di un servizio su Axis 1.4 da utilizzare per il trasporto WebSphere per SOAP utilizzando amqwdeployWMQService
Distribuire un servizio Axis 1.4 al trasporto WebSphere MQ per SOAP creando una directory di distribuzione, eseguendo il comando **amqwdeployWMQService** e avviando il listener Axis 1.4 .

Distribuzione di un servizio in CICS Transaction Server per l'utilizzo di WebSphere Transport per SOAP
Il trasporto WebSphere MQ per SOAP è integrato nel supporto dei servizi Web CICS Transaction Server 4.1 .

Distribuzione di un servizio in WebSphere Application Server per utilizzare WebSphere Transport for SOAP
Il trasporto WebSphere MQ per SOAP è integrato nel SIB (Service Integration Bus) su WebSphere Application Server.

Configurazione di WebSphere Application Server per utilizzare SOAP W3C su JMS
Un servizio Web collegato alla raccomandazione del candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server delle applicazioni Java EE . Questa attività è il passo 1 di connessione di un client del servizio Web Axis2 e di un servizio Web distribuito sul server delle applicazioni WebSphere mediante il protocollo SOAP over JMS W3C . Configurare le risorse WebSphere MQ e WebSphere Application Server per sviluppare e distribuire il servizio Web collegato a W3C SOAP su JMS come trasporto.

Distribuzione di un servizio all'endpoint del servizio WebSphere ESB e Process Server per utilizzare WebSphere Transport for SOAP

Il trasporto WebSphere MQ per SOAP non è direttamente supportato da WebSphere ESB e Process Server. È necessario configurare un'esportazione personalizzata.

Distribuzione di un servizio in CICS Transaction Server per l'utilizzo di WebSphere Transport per SOAP

Il trasporto WebSphere MQ per SOAP è integrato nel supporto dei servizi Web CICS Transaction Server 4.1 .

Prima di iniziare

Utilizzare gli stessi strumenti per sviluppare un client o un servizio per WebSphere MQ, come si farebbe per sviluppare HTTP. CICS dispone di strumenti corrispondenti a **`Java2wsdl`** e **`wsdl2Java`**:

- **`DFHWS2LS`** utilizza una descrizione del servizio Web come punto iniziale. Utilizza le descrizioni dei messaggi e i tipi di dati utilizzati in tali messaggi per creare strutture di dati in linguaggio di alto livello. È possibile utilizzare le strutture nei programmi di applicazione scritti in lingue diverse.
- **`DFHLS2WS`** prende come punto di partenza una struttura dati di linguaggio di alto livello. Utilizza la struttura per creare una descrizione dei servizi Web che contiene descrizioni dei messaggi. Crea inoltre schemi per i messaggi dalla struttura dati del linguaggio.

Seguire le istruzioni, Creazione di un Web service nella documentazione del prodotto CICS , per creare un Web service.

Informazioni su questa attività

Seguire le istruzioni, Configurazione di CICS per utilizzare il trasporto WebSphere MQ nella documentazione del prodotto CICS . Utilizzando le istruzioni, è possibile distribuire il servizio Web al trasporto WebSphere MQ per SOAP.

Attività correlate

Distribuzione di un servizio su Axis 1.4 da utilizzare per il trasporto WebSphere per SOAP utilizzando amqwdeployWMQService
Distribuire un servizio Axis 1.4 al trasporto WebSphere MQ per SOAP creando una directory di distribuzione, eseguendo il comando **`amqwdeployWMQService`** e avviando il listener Axis 1.4 .

Distribuzione di un servizio al servizio .NET Framework 1 o 2 per utilizzare il trasporto WebSphere MQ per SOAP
Distribuire un servizio .NET Framework 1 o 2 al trasporto WebSphere MQ per SOAP. Creare una directory di distribuzione, eseguire il comando **`amqwdeployWMQService`** e avviare il listener .NET.

Distribuzione di un servizio in WebSphere Application Server per utilizzare WebSphere Transport for SOAP
Il trasporto WebSphere MQ per SOAP è integrato nel SIB (Service Integration Bus) su WebSphere Application Server.

Configurazione di WebSphere Application Server per utilizzare SOAP W3C su JMS
Un servizio Web collegato alla raccomandazione del candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server delle applicazioni Java EE . Questa attività è il passo 1 di connessione di un client del servizio Web Axis2 e di un servizio Web distribuito sul server delle applicazioni WebSphere mediante il protocollo SOAP over JMS W3C . Configurare le risorse WebSphere MQ e WebSphere Application Server per sviluppare e distribuire il servizio Web collegato a W3C SOAP su JMS come trasporto.

Distribuzione di un servizio all'endpoint del servizio WebSphere ESB e Process Server per utilizzare WebSphere Transport for SOAP
Il trasporto WebSphere MQ per SOAP non è direttamente supportato da WebSphere ESB e Process Server. È necessario configurare un'esportazione personalizzata.

Distribuzione di un servizio in WebSphere Application Server per utilizzare WebSphere Transport for SOAP

Il trasporto WebSphere MQ per SOAP è integrato nel SIB (Service Integration Bus) su WebSphere Application Server.

Prima di iniziare

Utilizzare Rational Application Developer, WebSphere Integration Developer o un toolkit di servizi Web per sviluppare il servizio Web.

Informazioni su questa attività

Utilizzare le istruzioni riportate di seguito per distribuire un servizio utilizzando il trasporto WebSphere MQ per SOAP come un trasporto SOAP su WebSphere Application Server.

Procedura

1. Configurare WebSphere MQ come provider di messaggistica JMS per il SIB (Service Integration Bus) su WebSphere Application Server.
2. Configurare le risorse WebSphere MQ richieste dal servizio.
3. Seguire le istruzioni, Configurazione delle risorse JMS per il listener dell'endpoint JMS su SOAP sincrono, nella documentazione del prodotto WebSphere Application Server Network Deployment.

   Esistono istruzioni corrispondenti per altre piattaforme WebSphere Application Server.
4. Modificare l'URI del servizio in modo che sia conforme al trasporto WebSphere MQ per l'URI SOAP.
5. Distribuire il servizio a WebSphere Application Server.

Operazioni successive

Distribuire il servizio con HTTP come trasporto in modo che i client possano interrogare il servizio e ricevere il WSDL in risposta.

Attività correlate

Distribuzione di un servizio su Axis 1.4 da utilizzare per il trasporto WebSphere per SOAP utilizzando amqwdeployWMQService
Distribuire un servizio Axis 1.4 al trasporto WebSphere MQ per SOAP creando una directory di distribuzione, eseguendo il comando **amqwdeployWMQService** e avviando il listener Axis 1.4 .

Distribuzione di un servizio al servizio .NET Framework 1 o 2 per utilizzare il trasporto WebSphere MQ per SOAP
Distribuire un servizio .NET Framework 1 o 2 al trasporto WebSphere MQ per SOAP. Creare una directory di distribuzione, eseguire il comando **amqwdeployWMQService** e avviare il listener .NET.

Distribuzione di un servizio in CICS Transaction Server per l'utilizzo di WebSphere Transport per SOAP
Il trasporto WebSphere MQ per SOAP è integrato nel supporto dei servizi Web CICS Transaction Server 4.1 .

Configurazione di WebSphere Application Server per utilizzare SOAP W3C su JMS
Un servizio Web collegato alla raccomandazione del candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server delle applicazioni Java EE . Questa attività è il passo 1 di connessione di un client del servizio Web Axis2 e di un servizio Web distribuito sul server delle applicazioni WebSphere mediante il protocollo SOAP over JMS W3C . Configurare le risorse WebSphere MQ e WebSphere Application Server per sviluppare e distribuire il servizio Web collegato a W3C SOAP su JMS come trasporto.

Distribuzione di un servizio all'endpoint del servizio WebSphere ESB e Process Server per utilizzare WebSphere Transport for SOAP
Il trasporto WebSphere MQ per SOAP non è direttamente supportato da WebSphere ESB e Process Server. È necessario configurare un'esportazione personalizzata.

Configurazione di WebSphere Application Server per utilizzare SOAP W3C su JMS

Un servizio Web collegato alla raccomandazione del candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server delle applicazioni Java EE . Questa attività è il passo 1 di connessione di un client del servizio Web Axis2 e di un servizio Web distribuito sul server delle applicazioni WebSphere mediante il protocollo SOAP over JMS W3C . Configurare le risorse WebSphere MQ e WebSphere Application Server per sviluppare e distribuire il servizio Web collegato a W3C SOAP su JMS come trasporto.

Prima di iniziare

L'attività richiede WebSphere Application Server v7.0.0.9 e WebSphere MQ v7.0.1.3.

Informazioni su questa attività

L'attività è composta da due passi:

Procedura

1. “Configurare le risorse WebSphere MQ” a pagina 1009
2. “Configurare le WebSphere Application Server” a pagina 1010

Operazioni successive

“Configurare le risorse WebSphere MQ” a pagina 1009

Attività correlate

Distribuzione di un servizio su Axis 1.4 da utilizzare per il trasporto WebSphere per SOAP utilizzando amqwdeployWMQService

Distribuire un servizio Axis 1.4 al trasporto WebSphere MQ per SOAP creando una directory di distribuzione, eseguendo il comando **amqwdeployWMQService** e avviando il listener Axis 1.4 .

Distribuzione di un servizio al servizio .NET Framework 1 o 2 per utilizzare il trasporto WebSphere MQ per SOAP

Distribuire un servizio .NET Framework 1 o 2 al trasporto WebSphere MQ per SOAP. Creare una directory di distribuzione, eseguire il comando **amqwdeployWMQService** e avviare il listener .NET.

Distribuzione di un servizio in CICS Transaction Server per l'utilizzo di WebSphere Transport per SOAP

Il trasporto WebSphere MQ per SOAP è integrato nel supporto dei servizi Web CICS Transaction Server 4.1 .

Distribuzione di un servizio in WebSphere Application Server per utilizzare WebSphere Transport for SOAP

Il trasporto WebSphere MQ per SOAP è integrato nel SIB (Service Integration Bus) su WebSphere Application Server.

Distribuzione di un servizio all'endpoint del servizio WebSphere ESB e Process Server per utilizzare WebSphere Transport for SOAP

Il trasporto WebSphere MQ per SOAP non è direttamente supportato da WebSphere ESB e Process Server. È necessario configurare un'esportazione personalizzata.

Configurare le risorse WebSphere MQ

Prima di iniziare

Per il supporto Axis2 è stato richiesto WebSphere MQ 7.0.1.3 o versione successiva.

Informazioni su questa attività

Per semplicità, l'attività presume che WebSphere MQ sia installato sulla stessa stazione di lavoro dell'altro software e utilizzi connessioni di bind. Le configurazioni client WebSphere Application Server e Axis2 funzionano con le connessioni client. Per eseguire l'attività utilizzando le connessioni client, verificare che sia possibile inserire e richiamare i messaggi dalle code di richiesta e risposta sia dal client Axis2 che dai computer WebSphere Application Server.

Di nuovo, per semplicità, non viene utilizzata alcuna configurazione di protezione. L'ID utente dispone dell'autorizzazione `mqm` completa.

Procedura

1. Creare un gestore code predefinito, `QM1`.

Utilizzare WebSphere MQ Explorer per creare QM1 come gestore code predefinito. Configurarlo per avviarlo automaticamente e selezionare l'opzione per creare un listener. In alternativa, utilizzare i comandi riportati di seguito:

```
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. Definire una coda di richieste, REQUESTAXISe una coda di risposte, REPLYAXIS.

   Utilizzare Explorer o i seguenti comandi:

```
echo define ql(REQUESTAXIS) replace | runmqsc
echo define ql(REPLYAXIS) replace | runmqsc
```

Operazioni successive

"Configurare le WebSphere Application Server" a pagina 1010

Configurare le WebSphere Application Server

Prima di iniziare

Per il supporto W3C SOAP su JMS è necessario WebSphere Application Server v7. Questa configurazione è stata eseguita su WebSphere Application Server Versione 7.0 Test Environment v7.0.0.9 Aggiornamento 1. WebSphere Application Server è stato fornito con Rational Software Architect for WebSphere Software 7.5.4. Rational Software Architect è stato aggiornato a v7.5.5.1, applicando gli ultimi aggiornamenti disponibili.

Come parte del processo di installazione, creare un profilo per WebSphere Application Server. Nell'attività, la sicurezza amministrativa non è abilitata. Il nome profilo predefinito è `was70profile1`e il server è `server1`.

Informazioni su questa attività

Configurare WebSphere Application Server. È possibile avviare il server da Rational Application Developer e avviare la console di gestione dalla vista Server oppure è possibile avviare il server utilizzando un file di comandi e amministrare il server utilizzando un browser. L'attività utilizza il secondo metodo.

I file di comando del server si trovano nella cartella *Rational Installation Root*`\SDP\runtimes\base_v7\profiles\was70profile1\bin`. I file di log che si desidera esaminare si trovano in *Rational Installation Root*`\SDP\runtimes\base_v7\profiles\was70profile1\logs\server1`.

Per convenzione, tutti i nomi oggetto WebSphere MQ sono maiuscoli e tutti i nomi JNDI che fanno riferimento agli oggetti WebSphere MQ sono minuscoli.

Procedura

1. Avviare il server.

```
startServer server1
```

2. Avviare un browser, aprire la console di gestione e accedere.

```
http://localhost:9061/ibm/console/unsecureLogon.jsp
```

   Immettere qualsiasi stringa nel campo ID utente.

3. Creare una factory di connessione, qm1
   a) Nel navigator, aprire **Risorse** > **JMS** > **Factory di connessione**.
   b) Nella finestra Factory di connessione, selezionare l'ambito **Nodo =*nomenodo***, fare clic su **Nuovo**.

c) Selezionare **WebSphere MQ** > **OK**.

d) Fornire le informazioni di connessione del gestore code da Tabella 143 a pagina 1011 > **Avanti**.

Tabella 143. Informazioni sulla connessione del gestore code	
Nome campo	**Valore**
Nome	`qm1`
Nome JNDI	`qm1`

e) Selezionare **Immetti tutte le informazioni richieste in questa procedura guidata** come metodo di connessione> **Avanti**.

f) Immettere `QM1` come dettagli di connessione della coda> **Avanti**.

g) Immettere i dettagli di connessione da Tabella 144 a pagina 1011 > **Avanti**.

Tabella 144. Dettagli connessione	
Nome campo	**Valore**
Trasporto	`Bindings, then client`
Nome host	`localhost`
PORT	`1414`
Canale di connessione server	`SYSTEM.DEF.SVRCONN`

h) **Verifica connessione** > **Avanti** > **Fine** > **Salva**.

4. Creare la coda di richiesta JMS, `requestaxis`.

a) In Navigator, aprire **Risorse** > **JMS** > **Code**.

b) Nella finestra Factory di connessione, selezionare l'ambito **Nodo =*nomenodo***, fare clic su **Nuovo**.

c) Selezionare **WebSphere MQ** > **OK**.

d) Immettere i dettagli della coda da Tabella 145 a pagina 1011 > **OK** > **Salva**.

Tabella 145. Dettagli coda	
Nome campo	**Valore**
Nome	`requestaxis`
Nome JNDI	`requestaxis`
Nome coda	`REQUESTAXIS`
Nome del gestore code	`QM1`

5. Ripetere il passo "4" a pagina 1011 per creare la coda di risposta JMS, `replyaxis`.

6. Creare una specifica di attivazione, `qm1as`.

La specifica di attivazione attiva l'MDB (Message Driven Bean) del router del servizio Web quando un messaggio arriva sulla coda di richiesta. L'MDB è definito nel descrittore di distribuzione del servizio Web creato dalla procedura guidata del servizio Web di Rational Application Developer.

a) In Navigator, aprire **Risorse** > **JMS** > **Specifiche di attivazione**.

b) Nella finestra Factory di connessione, selezionare l'ambito **Nodo =*nomenodo***, fare clic su **Nuovo**.

c) Selezionare **WebSphere MQ** > **OK**.

d) Immettere gli attributi di base della specifica di attivazione da Tabella 146 a pagina 1012 > **Avanti**.

Tabella 146. Nome della specifica di attivazione	
Nome campo	**Valore**
Nome	`qm1as`
Nome JNDI	`qm1as`

e) Specificare le relative informazioni MDB da Tabella 147 a pagina 1012 > **Avanti**.

Tabella 147. Informazioni su MDB	
Nome campo	**Valore**
Nome JNDI di destinazione	`requestaxis`
Selettore messaggi	*Lasciato vuoto*
Tipo di destinazione	`Queue`

f) Selezionare **Immetti tutte le informazioni richieste in questa procedura guidata** come metodo di connessione> **Avanti**.

g) Immettere `QM1` come dettagli di connessione della coda> **Avanti**.

h) Immettere i dettagli di connessione da Tabella 144 a pagina 1011 > **Avanti**.

Tabella 148. Dettagli connessione	
Nome campo	**Valore**
Trasporto	`Bindings, then client`
Nome host	`localhost`
PORT	`1414`
Canale di connessione server	`SYSTEM.DEF.SVRCONN`

i) **Verifica connessione** > **Avanti** > **Fine** > **Salva**.

7. Creare un factory di connessione code, `jms/WebServicesReplyQCF`, per la coda di risposte.

   Il router dei servizi Web utilizza una factory di connessione code per accedere a una coda di risposte. Nel descrittore di distribuzione del servizio Web, al factory di connessione code viene assegnato il nome JNDI predefinito `jms/WebServicesReplyQCF`. È possibile modificare il nome del descrittore di distribuzione. In questa attività, aggiungere il nome predefinito alle definizioni delle risorse JMS.

   a) In Navigator, aprire **Risorse** > **JMS** > **Factory di connessione code**.

   b) Nella finestra Factory di connessione, selezionare l'ambito **Nodo =*nomenodo***, fare clic su **Nuovo**.

   c) Selezionare **WebSphere MQ** > **OK**.

   d) Immettere gli attributi di base del factory di connessione code da Tabella 149 a pagina 1012 > **Avanti**.

Tabella 149. Nome factory di connessione code	
Nome campo	**Valore**
Nome	`WebServicesReplyQCF`
Nome JNDI	`jms/WebServicesReplyQCF`

   e) Selezionare **Immetti tutte le informazioni richieste in questa procedura guidata** come metodo di connessione> **Avanti**.

   f) Immettere `QM1` come dettagli di connessione della coda> **Avanti**.

   g) Immettere i dettagli di connessione da Tabella 144 a pagina 1011 > **Avanti**.

Tabella 150. Dettagli connessione	
Nome campo	**Valore**
Trasporto	`Bindings, then client`
Nome host	`localhost`
PORT	`1414`
Canale di connessione server	`SYSTEM.DEF.SVRCONN`

h) **Verifica connessione** > **Avanti** > **Fine** > **Salva**.

Operazioni successive

"Sviluppo di un servizio Web EJB JAX - WS per SOAP W3C su JMS" a pagina 970

Distribuzione di un servizio all'endpoint del servizio WebSphere ESB e Process Server per utilizzare WebSphere Transport for SOAP

Il trasporto WebSphere MQ per SOAP non è direttamente supportato da WebSphere ESB e Process Server. È necessario configurare un'esportazione personalizzata.

Informazioni su questa attività

WebSphere Integration Developer fornisce una trasformazione dati SOAP che è possibile collegare all'esportazione JMS WebSphere MQ per creare un'esportazione SOAP JMS WebSphere MQ personalizzata.

Seguire le istruzioni per creare un'esportazione personalizzata per ricevere richieste SOAP su WebSphere MQ per SOAP.

Procedura

1. Leggere la sezione Panoramica delle importazioni ed esportazioni e Come connettersi a WebSphere MQ nella documentazione del prodotto WebSphere Process Server for Multiplatforms V6.2 .
2. Seguire l'attività Generazione di un bind di esportazione JMS di MQ nella documentazione del prodotto IBM Business Process Manager, Versione 8.6 .

   Utilizzare il bind dei dati SOAP descritto in Trasformazioni del formato dati JMS precompresso per formattare il messaggio SOAP.

Attività correlate

Distribuzione di un servizio su Axis 1.4 da utilizzare per il trasporto WebSphere per SOAP utilizzando amqwdeployWMQService
Distribuire un servizio Axis 1.4 al trasporto WebSphere MQ per SOAP creando una directory di distribuzione, eseguendo il comando **amqwdeployWMQService** e avviando il listener Axis 1.4 .

Distribuzione di un servizio al servizio .NET Framework 1 o 2 per utilizzare il trasporto WebSphere MQ per SOAP
Distribuire un servizio .NET Framework 1 o 2 al trasporto WebSphere MQ per SOAP. Creare una directory di distribuzione, eseguire il comando **amqwdeployWMQService** e avviare il listener .NET.

Distribuzione di un servizio in CICS Transaction Server per l'utilizzo di WebSphere Transport per SOAP
Il trasporto WebSphere MQ per SOAP è integrato nel supporto dei servizi Web CICS Transaction Server 4.1 .

Distribuzione di un servizio in WebSphere Application Server per utilizzare WebSphere Transport for SOAP
Il trasporto WebSphere MQ per SOAP è integrato nel SIB (Service Integration Bus) su WebSphere Application Server.

Configurazione di WebSphere Application Server per utilizzare SOAP W3C su JMS
Un servizio Web collegato alla raccomandazione del candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server delle applicazioni Java EE . Questa attività è il passo 1

di connessione di un client del servizio Web Axis2 e di un servizio Web distribuito sul server delle applicazioni WebSphere mediante il protocollo SOAP over JMS W3C . Configurare le risorse WebSphere MQ e WebSphere Application Server per sviluppare e distribuire il servizio Web collegato a W3C SOAP su JMS come trasporto.

Distribuzione dei client del servizio Web per l'utilizzo del trasporto WebSphere MQ per SOAP

Distribuire un client del servizio web in uno dei diversi ambienti client e connettersi a un servizio utilizzando il trasporto WebSphere MQ per SOAP.

Prima di iniziare

Sviluppare il servizio Web e distribuirlo per utilizzare il trasporto WebSphere MQ per SOAP.

Informazioni su questa attività

È possibile distribuire un client del servizio Web da eseguire con il trasporto WebSphere MQ per SOAP in diversi ambienti client. È possibile distribuire un client Java su Axis 1.4 utilizzando solo software installato con WebSphere MQ. Per gli altri ambienti client, è necessario installare software aggiuntivo.

Non si è limitati ad eseguire il trasporto WebSphere per SOAP negli ambienti client per cui sono presenti istruzioni di distribuzione. Utilizzare le istruzioni per distribuire un client in uno degli ambienti supportati.

Nota: Alcuni ambienti integrati offrono SOAP su JMS utilizzando il bind JMS SOAP consigliato da W3C e il trasporto WebSphere MQ per il bind SOAP. I rilasci di WebSphere MQ, fino a 7.0.1.2incluso, supportano solo il trasporto WebSphere MQ per il bind SOAP. A partire da 7.0.1.3 , è possibile distribuire i client Axis2 utilizzando un URI conforme al suggerimento candidato W3C per SOAP su JMS. Consultare l'esercitazione, Develop a SOAP/JMS JAX - WS Web services application with WebSphere Application Server V7 e Rational Application Developer V7.5.

Distribuzione di un client del servizio Web su Axis 1.4 per utilizzare il trasporto IBM WebSphere MQ per SOAP

Preparare una directory di distribuzione e un descrittore di distribuzione per il client. Fornire i proxy client e la classe client e configurare CLASSPATH. Configurare le code e canali di IBM WebSphere MQ , avviare il servizio e verificare il client.

Prima di iniziare

Suggerimento: Distribuire il servizio su HTTP, sviluppare e verificare il client per HTTP, quindi modificare il client per il trasporto IBM WebSphere MQ per SOAP:

1. Aggiungere la chiamata Register.extension() al client.
2. Modificare l'indirizzo del servizio Web statico nella classe del localizzatore proxy client per utilizzare l'URI per il trasporto IBM WebSphere MQ per SOAP.

Informazioni su questa attività

La distribuzione di un client Axis 1.4 per utilizzare il trasporto IBM WebSphere MQ per SOAP richiede un ulteriore passo di distribuzione rispetto a un client HTTP. È necessario creare un descrittore di distribuzione client, `client-config.wsdd`, per associare il `jms:` trasporto alla classe mittente `com.ibm.mq.soap.transport.jms.WMQSender`.

Se si utilizza il comando **`amqwdeployWMQService`** per generare i proxy client, è possibile distribuire il client utilizzando le directory generate dal comando.

Procedura

1. Creare una directory *deployDir* per conservare i file di distribuzione client.

2. Aprire una finestra di comandi su sistemi Windows o una shell di comandi utilizzando X Window System su sistemi UNIX and Linux , in *deployDir*.
3. Eseguire il comando **amqwsetcp.cmd** per impostare CLASSPATH
4. Eseguire il comando **amqwclientconfig.cmd** per creare un descrittore di distribuzione client Axis 1.4 , `client-config.wsdd` in *deployDir*.
5. Verificare che le classi nel pacchetto client, le classi proxy client e le librerie utilizzate dal client si trovino in CLASSPATH.

   amqwdeployWMQService inserisce i proxy del client .NET in `./generated/server/soap/client/remote/dotnetService` e i proxy Axis 1.4 in `./generated/server/soap/client/remote/`*`client package`*.

Esempio

L'esempio mostra la configurazione e l'output, Figura 199 a pagina 1016, da un client Axis 1.4 Java . Il client, Figura 198 a pagina 1015, chiama un servizio Web che ripete il rispettivo parametro di input. La definizione del servizio, Figura 197 a pagina 1015, mostra l'URI preso dal WSDL del servizio.

```
<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>
```

Figura 197. Definizione del servizio

```
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());
      }
   }
}
```

Figura 198. client 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
```

Figura 199. Output e configurazione client

Operazioni successive

1. Se si sta distribuendo il client come client IBM WebSphere MQ , configurare il canale di connessione client e server.
2. Se si sta distribuendo il client su un gestore code differente al servizio, è necessario rendere la coda di destinazione disponibile per il client. Configurare la coda di destinazione sul gestore code del servizio come una coda cluster o sul gestore code client come una definizione di coda remota.

Attività correlate

Distribuzione di un client del servizio Web su Axis2 per l'utilizzo del trasporto WebSphere MQ per SOAP
Preparare una directory di distribuzione e il file di configurazione Axis2 per il client. Fornire i proxy client e la classe client e impostare CLASSPATH. Configurare le code e canali WebSphere MQ , avviare il servizio e verificare il client.

Distribuzione su un client Axis2 utilizzando SOAP W3C su JMS
Un servizio Web collegato alla raccomandazione del candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server delle applicazioni Java EE . Questa attività è il passo 4 della connessione di un client del servizio Web Axis2 e di un servizio Web distribuito su WebSphere Application Server utilizzando il SOAP W3C su protocollo JMS. Modificare l'URL nel client Axis2 sviluppato per il trasporto WebSphere MQ per SOAP per utilizzare la raccomandazione del candidato W3C per SOAP su JMS.

Distribuzione di un client del servizio Web in .NET Framework 1 e 2 per utilizzare il trasporto WebSphere MQ per SOAP
Preparare una directory di distribuzione e un descrittore di distribuzione per il client. Fornire il proxy client e la classe client. Configurare le code e canali WebSphere MQ , avviare il servizio e verificare il client.

Distribuzione di un client del servizio Web su Axis2 per l'utilizzo del trasporto WebSphere MQ per SOAP

Preparare una directory di distribuzione e il file di configurazione Axis2 per il client. Fornire i proxy client e la classe client e impostare CLASSPATH. Configurare le code e canali WebSphere MQ , avviare il servizio e verificare il client.

Prima di iniziare

Suggerimento: Distribuire il servizio su HTTP. Sviluppare e verificare il client per HTTP, quindi modificare l'URL per fare riferimento al servizio utilizzando il trasporto WebSphere MQ per SOAP.

L'attività mostra come distribuire un client Axis2 non gestito in Java Standard Edition. È possibile distribuire un client Axis2 a un contenitore Web. In “Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse” a pagina 984, è stato sviluppato un client in un contenitore Web e distribuito in WebSphere Application Server Community Edition. Come parte della configurazione del server, è stato abilitato il facet Axis2 e incluso il facet nella configurazione del contenitore Web. Per configurare i contenitori Web su altri server delle applicazioni, fare riferimento alla documentazione di

Axis2 , http://ws.apache.org/axis2/1_4_1/installationguide.html#servlet_containero alla documentazione fornita con il server Web.

Nota: Axis2 utilizzare il termine contenitore servlet. Un contenitore servlet è uguale a un contenitore Web.

Informazioni su questa attività

La distribuzione di un client Axis2 per utilizzare WebSphere MQ per SOAP è come la distribuzione di un client Axis2 per utilizzare HTTP. Sono necessari ulteriori passi per fornire un percorso di classe ai file JAR di WebSphere MQ e per modificare il file di configurazione di Axis2 . Il file di configurazione Axis2 richiede una voce aggiuntiva per JMS. La voce fa riferimento al trasporto WebSphere MQ per il file JAR SOAP che implementa JMS `transportSender`.

Axis2 fornisce uno script, `axis2.bat` o `axis2.sh`, che semplifica la distribuzione del client; consultare gli esempi in Figura 203 a pagina 1019 e Figura 204 a pagina 1019.

Nota:

1. `axis2.bat` ha un bug che deve essere corretto. La stringa `-Djava.ext.dirs="%AXIS2_HOME%\lib\"` deve essere modificata in `-Djava.ext.dirs="%AXIS2_HOME%\lib\\"`.
2. In `axis2.bat` e `axis2.sh` `-Djava.ext.dirs` viene utilizzato come un modo rapido per fare riferimento a tutti i file JAR Axis2 , invece di aggiungerli separatamente al percorso classi. Purtroppo questo approccio è difettoso e funziona solo con alcuni JRE. Non funziona con i JRE IBM .

   Il parametro JVM, `-Djava.ext.dirs="%AXIS2_HOME%\lib\\"`, rende i file JAR dell'asse disponibili per JVM. La JVM tenta di creare un'istanza di alcuni file JAR di Axis e genera un errore, i cui dettagli dipendono dalla JVM. Di solito, è possibile visualizzare una delle seguenti righe nella traccia di stack:

   ```
   org.apache.axiom.om.util.UUIDGenerator.getInitialUUID(UUIDGenerator.java:76)
   ```

   o `org.apache.axis2.deployment.DeploymentException: java.security.NoSuchAlgorithmException: MD5 MessageDigest not available`

   Il modo corretto per eseguire un client Axis2 non gestito consiste nell'aggiungere i file JAR Axis2 al percorso di classe. Il percorso classi è disponibile solo per l'applicazione client e non per JVM.

La procedura descrive i passaggi generali per eseguire un client non gestito Axis2 senza utilizzare lo script axis2 . Gli esempi in Figura 201 a pagina 1018 e Figura 202 a pagina 1019 sono script per Windows e Linux.

Procedura

1. Scaricare Axis2 1.4.1 da http://ws.apache.org/axis2/download/1_4_1/download.cgi e decomprimere in una cartella, `Axis2-1.4.1`.
2. Aggiornare `axis2.xml` in `Axis2-1.4.1\conf`.
   a) Aggiornare `axis2.xml` in `Axis2-1.4.1\conf`. Aggiungere il trasporto WebSphere MQ per SOAP come `transportSender`:

   ```
   <transportSender name="jms"
   class="com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender"/>
   ```

   b) Se richiesto, modificare la dimensione del pool di connessioni dal valore predefinito di 10.

   ```
   <transportSender name="jms"
   class="com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender">
   <parameter name="ResourcePoolCapacity">20</parameter>
   </transportSender>
   ```

   `ResourcePoolResourcePool` definisce quante voci dell'endpoint del servizio vengono conservate nella cache. Il valore deve essere almeno 1. Se il numero di voci dell'endpoint del servizio supera la dimensione della cache, le voci vengono eliminate per creare spazio per le nuove

voci. La dimensione di una voce endpoint varia. Impostare un numero sufficientemente grande per evitare il thrashing della cache.

Consultare il passo 3 in “Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse” a pagina 984.

3. Creare una directory *deployDir*. In questa directory copiare la struttura di cartelle contenente i proxy client e client. *deployDir* equivale alla cartella *project*\bin in un progetto Java Eclipse .
4. Aprire una finestra di comando su Windowso una shell di comandi utilizzando X Window System su sistemi UNIX and Linux , in *deployDir*.
5. Aggiornare il percorso classi per includere la directory corrente, i file JAR Axis2 `com.ibm.mqjms.jar` e `com.ibm.mq.axis2.jar`.

   `com.ibm.mqjms.jar` fa riferimento a tutti gli altri file JAR WebSphere MQ richiesti.
6. Utilizzare il comando **Java** per avviare il programma client.

Esempi

Quattro esempi di esecuzione di un client Axis2 sono riportati in Figura 202 a pagina 1019 per Figura 204 a pagina 1019. Figura 200 a pagina 1018 mostra l'output dall'esecuzione del client asincrono elencato in Figura 183 a pagina 989.

```
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 . . .
```

Figura 200. Output dall'esecuzione di 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
```

Figura 201. runpojo.bat: Windows, utilizzando un percorso classi

```
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
```

Figura 202. runpojo.sh: Linux, utilizzando un percorso classi.

```
@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
```

Figura 203. runaxis2.bat: Windows, utilizzando axis2.bat

Nota

```
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
```

Figura 204. runaxis2.sh: Linux, utilizzo di axis2.sh

Nota

Attività correlate

Distribuzione di un client del servizio Web su Axis 1.4 per utilizzare il trasporto IBM WebSphere MQ per SOAP

Preparare una directory di distribuzione e un descrittore di distribuzione per il client. Fornire i proxy client e la classe client e configurare CLASSPATH. Configurare le code e canali di IBM WebSphere MQ , avviare il servizio e verificare il client.

Distribuzione su un client Axis2 utilizzando SOAP W3C su JMS

Un servizio Web collegato alla raccomandazione del candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server delle applicazioni Java EE . Questa attività è il passo 4 della connessione di un client del servizio Web Axis2 e di un servizio Web distribuito su WebSphere Application Server utilizzando il SOAP W3C su protocollo JMS. Modificare l'URL nel client Axis2 sviluppato per il trasporto WebSphere MQ per SOAP per utilizzare la raccomandazione del candidato W3C per SOAP su JMS.

Distribuzione di un client del servizio Web in .NET Framework 1 e 2 per utilizzare il trasporto WebSphere MQ per SOAP

Preparare una directory di distribuzione e un descrittore di distribuzione per il client. Fornire il proxy client e la classe client. Configurare le code e canali WebSphere MQ , avviare il servizio e verificare il client.

Distribuzione su un client Axis2 utilizzando SOAP W3C su JMS

Un servizio Web collegato alla raccomandazione del candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server delle applicazioni Java EE . Questa attività è il passo 4 della connessione di un client del servizio Web Axis2 e di un servizio Web distribuito su WebSphere Application Server utilizzando il SOAP W3C su protocollo JMS. Modificare l'URL nel client Axis2 sviluppato per il

trasporto WebSphere MQ per SOAP per utilizzare la raccomandazione del candidato W3C per SOAP su JMS.

Prima di iniziare

È necessario prima completare l'attività “Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse” a pagina 984 per richiamare **SimpleJavaListener** utilizzando un client Axis2 e il trasporto WebSphere MQ per il protocollo SOAP.

È inoltre necessario aver creato il servizio Web e configurato WebSphere MQ e WebSphere Application Server nelle attività precedenti:

1. “Configurare le risorse WebSphere MQ” a pagina 1009.
2. “Configurare le WebSphere Application Server” a pagina 1010.
3. “Sviluppo di un servizio Web EJB JAX - WS per SOAP W3C su JMS” a pagina 970.

Nell'attività, il client viene eseguito in Eclipse Galileo. È possibile eseguire il client dalla riga comandi modificando il file `Axis2.bat` fornito con Axis2.

Informazioni su questa attività

L'unica modifica da apportare al client statico Axis2 `StockQuoteAxis` esistente per richiamare il servizio Axis StockQuoteospitato da WebSphere Application Server consiste nel modificare l'URL passato al client. Poiché WSDL non è stato modificato, è possibile utilizzare le stesse classi proxy nel pacchetto `soap.server` .

Hai due approcci per definire l'URL da passare al client. È possibile utilizzare lo stesso URL del `StockQuoteAxis.wsdl`generato. Per accedere alla directory JNDI di WebSphere Application Server, è necessario aggiungere i parametri `jndiInitialContextFactory` e `jndiURL` . Un altro approccio consiste nel modificare l'URL e fornire al client l'accesso diretto alle code `REQUESTAXIS` e `REPLYAXIS` su `QM1`, senza utilizzare una ricerca JNDI.

I parametri di connessione definiti nell'URL passato al client Axis2 vengono utilizzati per connettersi al gestore code WebSphere MQ e alle code richieste per inviare e ricevere messaggi SOAP. I parametri di connessione passati al client Axis2 non vengono necessariamente utilizzati dal servizio. È possibile utilizzare le funzioni di accodamento distribuito di WebSphere MQ per separare il client e il servizio dall'utilizzo dello stesso gestore code o dello stesso server dei nomi.

Procedura

1. Salvare l'indirizzo URL da `StockQuoteAxis.wsdl` generato e chiudere Rational Application Developer per risparmiare memoria.

   Se la configurazione del server non è stata modificata, la chiusura di Rational Application Developer arresta il server delle applicazioni. In tal caso, avviare il server con il comando:

   ```
   startserver server1
   ```

2. Aprire Eclipse Galileo nel workspace con il progetto client Axis2 .
3. Aprire `SQA2StaticClient.java`.

   Vedere SQA2StaticClient.java.
4. Richiamare il servizio utilizzando la variante `queue` dell'URI.
   a) Modificare l'URL.

   Il nuovo URI è:

   ```
   jms:queue:REQUESTAXIS
             ?replyToName=REPLYAXIS
             &connectionFactory=connectQueueManager(QM1)Bind(Server)
             &targetService=StockQuoteAxis;
   ```

Confrontarlo con l'URL da `StockQuoteAxis.wsdl`:

```
jms:jndi:requestaxis
         ?jndiConnectionFactoryName=qm1
         &targetService=StockQuoteAxis
```

Figura 205. URL da `StockQuoteAxis.wsdl`

- `REQUESTAXIS` è ora in maiuscolo poiché è un nome coda e non un nome JNDI.
- La connessione a `QM1` è semplice.
- L'URI non contiene il nome della destinazione di risposta. Il client deve definire la coda su cui si prevede di ricevere risposte.

b) Eseguire `SQA2StaticClient.java` utilizzando lo stesso comando **Esegui come ...** configurazione come è stato fatto nell'attività, "Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse" a pagina 984.

5. Richiamare il servizio utilizzando la variante `jndi` dell'URI, utilizzando WebSphere Application Server come server di denominazione.

a) Utilizzare l'indirizzo URL da `StockQuoteAxis.wsdl`, Figura 205 a pagina 1021, fornendo i parametri mancanti per utilizzare il servizio di denominazione in WebSphere Application Server.

I parametri e i valori mancanti che è necessario fornire sono:

Tabella 151. Parametri JNDI aggiuntivi

Parametro	Valore utilizzato in questo esempio	Descrizione
`&jndiURL`	`iiop://localhost:2810` o `corbaname:iiop:localhost:2810`	URI del provider di denominazione. Per WebSphere Application Server il valore predefinito è 2809. È anche noto come numero di porta del connettore RMI e la porta di avvio. Il valore è elencato in `SystemOut.log` `00000000 NameServerImp A NMSV0018I: Name server available on bootstrap port 2810`
`&jndiInitialContextFactory`	`com.ibm.websphere.naming.WsnInitialContextFactory`	Il nome della factory di contesto iniziale utilizzato da WebSphereWebSphere Application Server.
`&replyToName`	`replyaxis`	Nome JNDI della coda `REPLYAXIS` .

```
jms:jndi:requestaxis?
         &jndiURL=iiop//localhost:2810
         &jndiConnectionFactoryName=qm1
         &jndiInitialContextFactory=com.ibm.websphere.naming.WsnInitialContextFactory
         &targetService=StockQuoteAxis
         &replyToName=replyaxis;
```

b) Aggiungere i file JAR richiesti dalla ricerca JNDI.

In questa configurazione, i seguenti file JAR sono stati aggiunti al percorso di generazione per eseguire l'attività utilizzando la variante `jndi` dell'URL JMS:

- `com.ibm.jaxws.thinclient_7.0.0.jar` da *`Rational install directory`*`\SDP\runtimes\base_v7\runtimes`.

- `com.ibm.ws.runtime.jar` da *`Rational install directory`*`\SDP\runtimes\base_v7\plugins`

Per un provider JNDI differente sono necessari file JAR differenti.

Gli altri file JAR nel percorso build sono:

i) Tutti i file JAR in *`WebSphere MQ Install directory`*`\java\lib`.

ii) Tutti i file JAR in *`Axis2-1.5.1`*`\lib`.

iii) JRE Java 6.0 .

c) Eseguire `SQA2StaticClient.java` utilizzando lo stesso comando **Esegui come ...** configurazione come è stato fatto nell'attività, “Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse” a pagina 984.

Risultati

In entrambi i casi, la risposta dal servizio viene visualizzata nella vista della console client.

Attività correlate

Distribuzione di un client del servizio Web su Axis 1.4 per utilizzare il trasporto IBM WebSphere MQ per SOAP
Preparare una directory di distribuzione e un descrittore di distribuzione per il client. Fornire i proxy client e la classe client e configurare CLASSPATH. Configurare le code e canali di IBM WebSphere MQ , avviare il servizio e verificare il client.

Distribuzione di un client del servizio Web su Axis2 per l'utilizzo del trasporto WebSphere MQ per SOAP
Preparare una directory di distribuzione e il file di configurazione Axis2 per il client. Fornire i proxy client e la classe client e impostare CLASSPATH. Configurare le code e canali WebSphere MQ , avviare il servizio e verificare il client.

Distribuzione di un client del servizio Web in .NET Framework 1 e 2 per utilizzare il trasporto WebSphere MQ per SOAP
Preparare una directory di distribuzione e un descrittore di distribuzione per il client. Fornire il proxy client e la classe client. Configurare le code e canali WebSphere MQ , avviare il servizio e verificare il client.

Distribuzione di un client del servizio Web in .NET Framework 1 e 2 per utilizzare il trasporto WebSphere MQ per SOAP

Preparare una directory di distribuzione e un descrittore di distribuzione per il client. Fornire il proxy client e la classe client. Configurare le code e canali WebSphere MQ , avviare il servizio e verificare il client.

Prima di iniziare

Suggerimento: Sviluppa e verifica il servizio e il client utilizzando Visual Studio. Modificare quindi il client per il trasporto WebSphere MQ per SOAP.

1. Se si sta distribuendo un servizio utilizzando .NET Framework 1 o 2, creare il servizio come libreria (`.dll`). Distribuire utilizzando il trasporto WebSphere MQ per SOAP.
2. Aggiungere la chiamata Register.Extension() al client.
3. Aggiungere un riferimento a amqsoap.dll, che si trova in *`MQ_Install`*`\bin`.
4. Modificare la proprietà statica `Url` nel costruttore della classe proxy client in `jms:/` URI, per il trasporto WebSphere MQ per SOAP.

Informazioni su questa attività

La distribuzione di un client del servizio Web per .NET Framework 1 o 2 per utilizzare il trasporto WebSphere MQ per SOAP richiede un ulteriore passo di distribuzione. È necessario registrare `amqsoap.dll` con .NET Framework. `amqsoap.dll` viene registrato automaticamente come parte dell'installazione del trasporto WebSphere MQ per SOAP, ma potrebbe essere necessario registrarlo di nuovo.

Se si utilizza il comando **amqwdeployWMQService** per generare i proxy client, è possibile distribuire il client utilizzando le directory generate dal comando.

Procedura

1. Creare una directory *deployDir* per conservare i file di distribuzione client.
2. Aprire una finestra comandi in *deployDir*.
3. Eseguire **amqwsetcp** per impostare CLASSPATH se il servizio deve essere eseguito su Axis 1.4.
4. Se necessario, eseguire **amqwRegisterDotNet** per registrare `amqsoap.dll` con .NET Framework.

Esempio

L'esempio mostra la configurazione e l'output, Figura 208 a pagina 1023, da un client .NET Framework V2 . Il client, Figura 207 a pagina 1023, richiama un servizio Web che ripete il relativo parametro di input. La definizione Url statica, Figura 206 a pagina 1023, mostra il costruttore per il client proxy.

```
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";
}
```

Figura 206. Costruttore proxy client statico

```
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);
            }
        }
    }
}
```

Figura 207. Programma client

```
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
```

Figura 208. Configurazione e output

Operazioni successive

1. Se si sta distribuendo il client come client WebSphere MQ MQI, configurare il canale di connessione client e server.
2. Se si sta distribuendo il client su un gestore code differente al servizio, è necessario rendere la coda di destinazione disponibile per il client. Configurare la coda di destinazione sul gestore code del servizio come una coda cluster o sul gestore code client come una definizione di coda remota.

Attività correlate

Distribuzione di un client del servizio Web su Axis 1.4 per utilizzare il trasporto IBM WebSphere MQ per SOAP
Preparare una directory di distribuzione e un descrittore di distribuzione per il client. Fornire i proxy client e la classe client e configurare CLASSPATH. Configurare le code e canali di IBM WebSphere MQ , avviare il servizio e verificare il client.

Distribuzione di un client del servizio Web su Axis2 per l'utilizzo del trasporto WebSphere MQ per SOAP
Preparare una directory di distribuzione e il file di configurazione Axis2 per il client. Fornire i proxy client e la classe client e impostare CLASSPATH. Configurare le code e canali WebSphere MQ , avviare il servizio e verificare il client.

Distribuzione su un client Axis2 utilizzando SOAP W3C su JMS
Un servizio Web collegato alla raccomandazione del candidato W3C per SOAP su JMS deve essere eseguito nel contenitore EJB di un server delle applicazioni Java EE . Questa attività è il passo 4 della connessione di un client del servizio Web Axis2 e di un servizio Web distribuito su WebSphere Application Server utilizzando il SOAP W3C su protocollo JMS. Modificare l'URL nel client Axis2 sviluppato per il trasporto WebSphere MQ per SOAP per utilizzare la raccomandazione del candidato W3C per SOAP su JMS.

Connettere un client Axis2 a un servizio JAX - WS utilizzando SOAP W3C su JMS e WebSphere Application Server

Una volta completata questa attività, verrà richiamato un servizio Web JAX - WS in esecuzione in WebSphere Application Server da un client Axis2 . Il client Axis2 e WebSphere Application Server utilizzano il suggerimento del candidato W3C per il protocollo SOAP over JMS in esecuzione su WebSphere MQ. Utilizzare Eclipse GALILEO e Rational Application Developer per creare rispettivamente il client del servizio Web e il servizio Web.

Prima di iniziare

L'attività richiede la versione 7 di Rational Software Development Environment e WebSphere Application Server. L'attività è stata creata utilizzando il package di Rational Application Developer con Rational Software Architect for WebSphere Software v7.5.5.1e WebSphere Application Server Versione 7.0 Test Environment v7.0.0.9 Update 1. È inoltre necessario WebSphere MQ v7.0.1.3.

L'attività si basa su altre due attività “Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse” a pagina 963e “Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse” a pagina 984. Per completare queste attività, l'ambiente di sviluppo dispone già di Eclipse Galileo, WASCE, il plugin Eclipse per WASCE e Axis2 1.4.1 installato. Non si richiede WASCE per questa attivit ....

Alcuni dei passaggi sono complessi. I passi presuppongono una certa familiarità con lo sviluppo di applicazioni di servizi Web per WebSphere Application Server utilizzando Rational Application Developer. Le richieste di processore e memoria dell'attività sono elevate. L'attività è stata eseguita in una macchina virtuale VMWare Windows XP SP3 assegnata 1.8GB di memoria.

Installare tutto il software prima di avviare l'attività. Il software impiega circa un giorno per scaricare e un giorno per installare, a seconda della larghezza di banda. L'attività richiede almeno mezza giornata.

Informazioni su questa attività

Lo scenario per questa attività è che è stato sviluppato un servizio Web di quotazione azionaria, `StockQuoteAxis`, utilizzando uno strumento open source, Eclipse Galileo. `StockQuoteAxis` viene distribuita utilizzando SOAP su HTTP in esecuzione su un server open source, WASCE.

Si desidera collegare i servizi Web distribuiti ad un trasporto di messaggistica basato su standard, ad esempio SOAP su JMS, o alla messaggistica affidabile dei servizi Web, oltre che a SOAP su HTTP. Si desidera che sia il servizio che il client utilizzino interfacce basate su standard. Per questo motivo, anche se il team di sviluppo dei progetti futuri ha implementato una soluzione utilizzando il trasporto WebSphere MQ per SOAP, non si è entrati nella produzione.

Il client Axis2 ha rimosso il problema che il client SOAP per il trasporto WebSphere MQ per SOAP richiedeva una modifica dal client HTTP. Il problema è rimasto che il servizio connesso dal trasporto IBM WebSphere MQ per SOAP è ospitato da un listener speciale fornito da WebSphere MQ: `SimpleJavaListener`.

Con lo standard W3C SOAP su JMS nello stato di raccomandazione del candidato, alcuni fornitori forniscono supporto per W3C SOAP su JMS. Il supporto consente di distribuire un servizio Web a un application server e di connettersi allo stesso servizio utilizzando una varietà di protocolli di connettività. Il supporto fornito da WebSphere Application Server v7 elimina il problema di dover ospitare il servizio Web separatamente per utilizzare un trasporto SOAP basato su messaggi. L'utilizzo di un'interfaccia di trasporto messaggi basata su standard, JMS, significa che è possibile sviluppare soluzioni utilizzando strumenti di fornitori diversi. Si spera che gli strumenti dei servizi Web in Eclipse includano i bind SOAP su JMS in futuro.

La maggior parte dei passi viene eseguita utilizzando Eclipseo gli strumenti di gestione forniti con i prodotti WebSphere . I passi sono descritti per un ambiente Windows . Con lievi modifiche ad alcuni comandi, è possibile eseguire i passaggi su altre piattaforme.

Vengono elencati i passi preliminari per la creazione del servizio Web HTTP e la connessione mediante Axis2 . Il client e WSDL da questi passi vengono utilizzati per creare la soluzione

Procedura

1. Connettersi al servizio Web Axis StockQuoteutilizzando il client Axis2 e il trasporto IBM WebSphere MQ per SOAP
   a) “Sviluppo di un servizio JAX - RPC per WebSphere MQ transport for SOAP utilizzando Eclipse” a pagina 963
   b) “Sviluppo di un client JAX - WS per il trasporto WebSphere per SOAP mediante Eclipse” a pagina 984
   c) “Distribuzione di un client del servizio Web su Axis2 per l'utilizzo del trasporto WebSphere MQ per SOAP” a pagina 1016
2. Collegarsi al servizio Web dell'asse StockQuoteutilizzando un client Axis2 e il suggerimento candidato W3C per SOAP su JMS.
   a) “Configurare le risorse WebSphere MQ” a pagina 1009
   b) “Configurare le WebSphere Application Server” a pagina 1010
   c) “Sviluppo di un servizio Web EJB JAX - WS per SOAP W3C su JMS” a pagina 970
   d) “Distribuzione su un client Axis2 utilizzando SOAP W3C su JMS” a pagina 1019

WebSphere MQ Bridge per HTTP

Con il bridge WebSphere MQ per l'HTTP, le applicazioni client possono scambiare messaggi con WebSphere MQ senza dover installare un client WebSphere MQ MQI. È possibile richiamare WebSphere MQ da qualsiasi piattaforma o lingua con funzionalità HTTP.

Introduzione al bridge WebSphere MQ per HTTP

Il bridge WebSphere MQ per HTTP è un'applicazione Web JEE (Java, Enterprise Environment). I client HTTP possono inviare richieste **POST**, **GET**e **DELETE** per inserire, ricercare ed eliminare messaggi dalle code WebSphere MQ . Il bridge WebSphere MQ per HTTP non è adatto per l'utilizzo con i messaggi, se è richiesta una consegna garantita.

Vantaggi

Con il bridge WebSphere MQ per HTTP, è possibile inviare e ricevere messaggi WebSphere MQ utilizzando HTTP da una vasta gamma di ambienti:

- Ambienti che supportano HTTP, ma non WebSphere MQ.

- Ambienti con spazio di archiviazione insufficiente per installare un client WebSphere MQ MQI.
- Ambienti troppo numerosi per installare il client WebSphere MQ MQ su ciascun sistema che richiede l'accesso a WebSphere MQ.
- Applicazioni basate sul Web da cui si desidera inviare o ricevere messaggi senza codificare il proprio bridge per WebSphere MQ.
- Applicazioni basate sul Web che si desidera migliorare, utilizzando tecniche asincrone come AJAX. WebSphere Il bridge MQ per HTTP rende disponibili le code e gli argomenti WebSphere MQ utilizzando REST (Representation State Transfer) su HTTP.

Il supporto HTTP può essere utilizzato sia con le topologie di messaggistica point-to-point che di pubblicazione / sottoscrizione.

Come funziona il supporto HTTP?

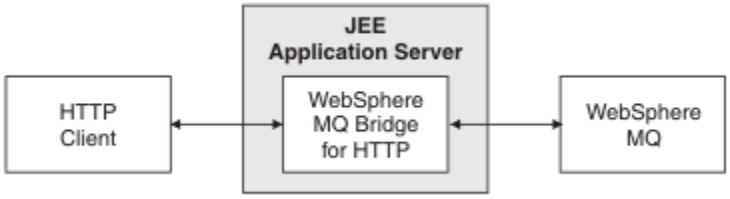


Figura 209. WebSphere MQ Bridge per HTTP

Il bridge WebSphere MQ per l'applicazione Web HTTP riceve richieste HTTP da uno o più client. Interagisce con WebSphere MQ per loro conto e restituisce risposte HTTP.

Il bridge WebSphere MQ per HTTP è un servlet JEE connesso a WebSphere MQ utilizzando un adattatore risorse. Il servlet HTTP gestisce tre diversi tipi di richieste: **POST**, **GET**e **DELETE**.

Tabella 152. WebSphere MQ bridge per i verbi HTTP

Richiesta HTTP	**Risultato**
PUBBLICA	Inserisce un messaggio in una coda o in un argomento.
GET	Sfoglia il primo messaggio su una coda. In linea con il protocollo HTTP, **GET** non elimina il messaggio dalla coda. Non utilizzare **GET** con la messaggistica di pubblicazione / sottoscrizione.
ELIMINA	Richiama ed elimina un messaggio da una coda o da un argomento.

Esempio di HTTP POST

HTTP **POST** inserisce un messaggio in una coda o una pubblicazione in un argomento. L'esempio **HTTPPOST** è un esempio di una richiesta HTTP **POST** di un messaggio a una coda. Invece di utilizzare Java, è possibile creare una richiesta HTTP **POST** utilizzando invece un modulo browser oppure un toolkit AJAX.

Figura 210 a pagina 1027 mostra una richiesta HTTP per inserire un messaggio su una coda denominata `myQueue`. Questa richiesta contiene l'intestazione HTTP `x - msg -correlId` per impostare l'ID di correlazione del messaggio 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.
```

*Figura 210. Esempio di una richiesta HTTP **POST** in una coda*

Figura 211 a pagina 1027 mostra la risposta inviata al client. Non vi è alcun contenuto della risposta.

```
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
```

Figura 211. Esempio di una risposta HTTP POST

Esempio di HTTP DELETE

HTTP **DELETE** richiama un messaggio da una coda e lo elimina oppure richiama ed elimina una pubblicazione. L'esempio Java **HTTPDELETE** è un esempio di una richiesta HTTP **DELETE** che legge un messaggio da una coda. Invece di utilizzare Java, è possibile creare una richiesta HTTP **DELETE** utilizzando invece un modulo browser oppure un toolkit AJAX.

Figura 212 a pagina 1027 è una richiesta HTTP per eliminare il prossimo messaggio sulla coda denominato `myQueue`. Come risposta, il corpo del messaggio viene restituito al client. Nei termini WebSphere MQ , HTTP **DELETE** è un richiamo distruttivo.

La richiesta contiene l'intestazione della richiesta HTTP `x - msg - wait`, che indica al bridge WebSphere MQ per HTTP per quanto tempo attendere l'arrivo di un messaggio nella coda. La richiesta contiene anche l'intestazione della richiesta `x-msg-require-headers`, che specifica che il client riceverà l'ID di correlazione del messaggio nella risposta.

```
DELETE /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Figura 212. Esempio di una richiesta HTTP **DELETE***

Figura 213 a pagina 1027, è la risposta restituita al client. L'ID di correlazione viene restituito al client, come previsto in `x-msg-require-headers` nella richiesta.

```
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.
```

*Figura 213. Esempio di una risposta HTTP **DELETE***

Esempio di HTTP GET

HTTP **GET** richiama un messaggio da una coda. Il messaggio rimane nella coda. Nei termini WebSphere MQ , HTTP **GET** è una richiesta di ricerca. Si può creare una richiesta HTTP **GET** utilizzando un client Java, un modulo browser o un toolkit AJAX.

Figura 214 a pagina 1028 è una richiesta HTTP per sfogliare il messaggio successivo sulla coda denominata `myQueue`.

La richiesta contiene l'intestazione della richiesta HTTP `x - msg - wait`, che indica al bridge WebSphere MQ per HTTP per quanto tempo attendere l'arrivo di un messaggio nella coda. La richiesta contiene anche l'intestazione della richiesta `x-msg-require-headers`, che specifica che il client riceverà l'ID di correlazione del messaggio nella risposta.

```
GET /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Figura 214. Esempio di una richiesta HTTP **GET***

Figura 215 a pagina 1028 è la risposta restituita al client. L'ID di correlazione viene restituito al client, come previsto in `x-msg-require-headers` nella richiesta.

```
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.
```

*Figura 215. Esempio di una risposta HTTP **GET***

Installazione, configurazione e verifica del bridge WebSphere MQ per HTTP

Ottenere il bridge WebSphere MQ per HTTP installando "Java Messaging and Web Services" dal client WebSphere MQ MQI o dai materiali di installazione del server. Distribuire WebSphere MQ bridge for HTTP su un server delle applicazioni adatto.

Prima di iniziare

Controllare i prodotti prerequisiti all'indirizzo Requisiti di sistema per IBM WebSphere MQ. Il processo di installazione non verifica la presenza e disponibilità del software prerequisito per l'esecuzione del bridge WebSphere MQ per HTTP. È necessario verificare che i prerequisiti siano installati.

WebSphere MQ bridge per HTTP viene eseguito su qualsiasi server delle applicazioni conforme a Java EE 1.4 , installando l'adattatore di risorse WebSphere MQ . È anche possibile eseguire il bridge WebSphere MQ per HTTP su una release di WebSphere Application Server precedente a 6.0.2.1. Utilizzare WebSphere Application Server Message Listener Port (MLP) per integrare WebSphere MQ come provider JMS.

Il supporto per il bridge WebSphere MQ per HTTP viene fornito solo per i seguenti server delle applicazioni:

- WebSphere Application Server 6.0.2.1 e successive.
- WebSphere Application Server Community Edition Versione 1.1 e successive.

Informazioni su questa attività

WebSphere Il bridge MQ per HTTP viene fornito come file `.war` , `WMQHTTP.war`.

- Su piattaforme UNIX e Linux,
  - `WMQHTTP.war` è incluso come parte dell'opzione di installazione "Java Messaging and Web Services" . L'opzione è disponibile sia nel materiale di installazione client che server.
  - `WMQHTTP.war` viene installato in *<mqmtop>*`/java/http/WMQHTTP.war`. < *mqmtop>* è la directory in cui è installato WebSphere MQ .
  - `WMQHTTP.samples` viene installato in *<mqmtop>*`/java/http/samples`. < *mqmtop>* è la directory in cui è installato WebSphere MQ .

Effettuare le seguenti operazioni di installazione per installare WebSphere MQ bridge for HTTP, distribuirlo e configurarlo e verificare la configurazione. I dettagli della procedura di configurazione variano a seconda dei server delle applicazioni. Utilizzare “Distribuzione e verifica di WebSphere MQ bridge for HTTP on WebSphere Application Server V6.1.0.9” a pagina 1030 come modello per i passi da seguire sul server delle applicazioni.

Procedura

1. Ottenere `WMQHTTP.war` installando il client o il server MQI WebSphere MQ .
2. Copiare `WMQHTTP.war` su un server da cui può essere distribuito su un server delle applicazioni.
3. Distribuire `WMQHTTP.war` su un server delle applicazioni.
4. Se necessario, installare WebSphere MQ come adattatore di risorse sul server delle applicazioni.

   Verificare se WebSphere MQ è già configurato come provider di messaggistica sul server delle applicazioni. Utilizzare lo strumento di gestione o di amministrazione fornito con il server delle applicazioni per ricercare WebSphere MQ. WebSphere MQ potrebbe essere trovato nel seguente percorso, **Risorse> JMS> Provider di messaggistica**.
5. Configurare una factory di connessione sul server delle applicazioni per connettersi a un gestore code che utilizza il trasporto client WebSphere MQ MQI[12].
6. Configurare l'applicazione Web `WMQHTTP.war` sul server delle applicazioni per utilizzare la factory di connessione
7. Verificare la configurazione.
   a) Impostare il gestore code denominato nella produzione connessioni e una coda locale.
   b) Inserire un messaggio nella coda locale.
   c) Creare il canale di connessione server denominato nel factory di connessione, con l'autorità di leggere e scrivere nella coda locale.
   d) Avviare il gestore code e il listener.
   e) Avviare il server delle applicazioni e `WMQHTTP.war`, se non sono già in esecuzione.
   f) Aprire un browser e digitare `http://`*`hostname`*`:`*`web port`*`/`*`Context root`*`/msg/queue/`*`local queue`*

Risultati

La finestra del browser visualizza il messaggio inserito nella coda locale.

Operazioni successive

1. Provare l'esempio, “Distribuzione e verifica di WebSphere MQ bridge for HTTP on WebSphere Application Server V6.1.0.9” a pagina 1030.
2. Eseguire le applicazioni Java HTTP di esempio.

[12] Inizialmente, almeno, configurare il trasporto client. Alcuni server delle applicazioni possono connettersi a WebSphere MQ utilizzando connessioni dirette o in modalità bind.

Distribuzione e verifica di WebSphere MQ bridge for HTTP on WebSphere Application Server V6.1.0.9

Utilizzare il seguente esempio per preparare una distribuzione del bridge WebSphere MQ per HTTP per eseguire i programmi Java HTTP di esempio. La distribuzione è su WebSphere Application Server V6.1.0.9.

Prima di iniziare

1. Seguire le istruzioni riportate in “Installazione, configurazione e verifica del bridge WebSphere MQ per HTTP” a pagina 1028, per copiare `WMQHTTP.war` su un server accessibile all'installazione di WebSphere Application Server.
2. Configurare un gestore code e una coda da utilizzare per verificare la configurazione:
   - Nell'esempio, il gestore code è configurato in modo da utilizzare i valori in Tabella 153 a pagina 1030:

Tabella 153. Configurazione del gestore code	
Oggetto	**Valore**
Nome host	`itso-01`
Gestore code	`QM1`
Coda locale	`HTTPTESTQ`
Canale di connessione server	`MYSVRCON`. Configurare un ID utente MCA con autorizzazione sufficiente per leggere e scrivere in `HTTPTESTQ`.
Porta listener	`1414`

3. Avviare il gestore code e il listener
4. Inserire un messaggio di test in `HTTPTESTQ`. Ad esempio:
   a. Avviare WebSphere MQ Explorer.
   b. Nell'elenco di code locali per QM1, fare clic con il tasto destro del mouse su **HTTPTESTQ > Inserisci messaggio di prova> tipo `First Message` > Inserisci messaggio> Chiudi**
5. Avviare il server delle applicazioni e collegarsi a Integrated Solutions Console.

Informazioni su questa attività

L'esempio mostra i passi da eseguire se si sta eseguendo WebSphere Application Server V6.1.0.9 come server delle applicazioni. Se si sta eseguendo una versione diversa di WebSphere Application Server o si sta eseguendo un server delle applicazioni differente, la procedura è diversa. WebSphere Application Server V6.1.0.9 è preconfigurato con WebSphere MQ installato come provider di messaggi, utilizzando le librerie del client MQI WebSphere MQ . Se WebSphere MQ non è preconfigurato come provider di messaggistica o se si desidera utilizzare i bind del server WebSphere MQ , è necessario installare e configurare l'adattatore di risorse WebSphere MQ per JEE nel server delle applicazioni.

Seguire le istruzioni per distribuire il bridge WebSphere MQ per HTTP in WebSphere Application Server V6.1.0.9e verificare la distribuzione utilizzando un browser:

Procedura

1. Nel riquadro di navigazione, fare clic su **Risorse> Provider JMS> Provider di messaggistica WebSphere MQ**.

   È possibile configurare a livello di nodo, cella o server, in base alla distribuzione di WebSphere Application Server. L'esempio utilizza la distribuzione a livello server.
2. Sotto **Proprietà aggiuntive**, fare clic su **Factory connessioni> Nuovo**.

3. Nel modulo dei provider JMS, fornire le informazioni in Tabella 154 a pagina 1031, o le alternative di propria scelta, fare clic su **Applica> Salva**.

Tabella 154. Impostare o modificare i seguenti campi

Campo	**Valore**
Nome	`WMQHTTPBridge`
Nome JNDI	`jms/WMQHTTPJCAConnectionFactory`
Gestore code	`QM1`
Host	`itso-01`
PORT	`1414`
Canale	`MYSVRCON`
Tipo trasporto	`CLIENT`

4. Nel riquadro di navigazione, fare clic su **Applicazioni> Installa nuova applicazione**.
5. Inserire il percorso di `WMQHTTP.war` nel form e fornire una root di contesto, fare clic su **Avanti**.
   a) La root di contesto è facoltativa. `mq` è la root di contesto predefinita per le applicazioni HTTP di esempio.
   b) La root di contesto fa parte dell'URI che identifica il bridge WebSphere MQ per HTTP. È possibile omettere la root di contesto o modificarla successivamente.
6. Nella pagina **Seleziona opzioni di installazione** della procedura guidata di installazione, non è necessario modificare i valori predefiniti, fare clic su **Avanti**.
7. Nella pagina **Associa moduli ai server** , selezionare un cluster o un server, selezionare la casella di selezione e fare clic su **Applica> Avanti**.
8. Nella pagina **Associa riferimenti risorsa alle risorse** , nel modulo **javax.jms.ConnectionFactory** , fare clic su **Sfoglia ...** su IBM WebSphere MQ bridge for HTTP row.
9. Nella pagina **Applicazioni enterprise> Risorse disponibili** , selezionare **WMQHTTPBridge**, fare clic su **Applica**.
10. Nel modulo **javax.jms.ConnectionFactory** , selezionare il metodo di autenticazione.
    a) Per l'esempio, scegliere **Nessuno**, fare clic su **Applica**. Le altre opzioni richiedono una configurazione aggiuntiva.
11. Selezionare la casella di spunta **Seleziona** per IBM WebSphere MQ bridge for HTTP, fare clic su **Avanti> Avanti> Fine> Salva**
12. Nel riquadro di navigazione, fare clic su **Applicazioni> Applicazioni enterprise**.
13. Selezionare la casella di selezione per `WMQHTTP.war`, fare clic su **Avvia**.
14. Aprire una finestra del browser. Immettere `http://itso-01:9080/mq/msg/queue/HTTPTESTQ`, utilizzando il nome host e la porta appropriati.

Risultati

La finestra del browser visualizza `First Message`, se la configurazione ha esito positivo.

Operazioni successive

Eseguire le applicazioni Java HTTP di esempio.

Pubblicazione / sottoscrizione mediante il bridge WebSphere MQ per HTTP

WebSphere MQ bridge for HTTP utilizza WebSphere MQ classes per l'interfaccia di pubblicazione / sottoscrizione JMS. HTTP **POST** crea una pubblicazione. HTTP **DELETE** crea una sottoscrizione gestita

non durevole. È necessario configurare la pubblicazione / sottoscrizione per JMS prima di utilizzare l'URI dell'argomento.

La pubblicazione / sottoscrizione è completamente integrata in WebSphere MQ nella versione 7. Prima della versione 7, un broker di pubblicazione / sottoscrizione separato gestiva le pubblicazioni e le sottoscrizioni. Si chiama pubblicazione / sottoscrizione "in coda" , per distinguerla dalla pubblicazione / sottoscrizione completamente integrata nella versione 7. La versione 7 emula la sottoscrizione di pubblicazione accodata utilizzando la pubblicazione / sottoscrizione integrata. L'emulazione consente alle applicazioni di pubblicazione / sottoscrizione accodate esistenti di coesistere con le applicazioni integrate in esecuzione sullo stesso gestore code. Le applicazioni di pubblicazione / sottoscrizione accodate possono anche interagire con le applicazioni integrate, condividendo gli stessi argomenti. Nella versione 6, il broker è stato fornito con WebSphere MQ; prima della versione 6 era disponibile come SupportPack.

Configurazione

Il bridge WebSphere MQ per HTTP utilizza l'interfaccia JMS per la pubblicazione e la sottoscrizione. Nella versione 7, è possibile controllare se le classi WebSphere MQ per JMS utilizzano la pubblicazione / sottoscrizione accodata o integrata, utilizzando la proprietà `PROVIDERVERSION` JMS.

Un'ulteriore considerazione è che è possibile utilizzare le librerie del client MQI WebSphere MQ con il bridge WebSphere MQ per HTTP o le librerie del server. Le librerie client versione 6 supportano solo la pubblicazione / sottoscrizione accodata, mentre le librerie versione 7 supportano sia la pubblicazione / sottoscrizione accodata che quella integrata. La maggior parte dei server delle applicazioni o Web che utilizzano WebSphere MQ come provider di messaggistica utilizzano le librerie client. Per utilizzare la pubblicazione / sottoscrizione integrata, sia il client WebSphere MQ MQI che le librerie server devono essere almeno alla versione 7. Se si sta eseguendo una versione precedente di WebSphere rispetto alla 7, è necessario configurare la pubblicazione / sottoscrizione accodata; consultare Tabella 155 a pagina 1032. Verificare quali librerie sono installate o configurate con il server Web o il server delle applicazioni che si sta utilizzando.

Tabella 155. Modalità di configurazione di pubblicazione / sottoscrizione

	Client V6 o precedente	**Client V7 o successivo**
Server V6 o precedente	1. Eseguire lo script `\java\bin\MQJMS_PSQ.mqsc`	Non supportato
Server V7 o successivo	1. Eseguire lo script `\java\bin\MQJMS_PSQ.mqsc` 2. Impostare il gestore code su `PSMODE=ENABLED`	1. Se PROVIDERVERSION = 7 a. Impostare il gestore code su `PSMODE=ENABLED` o `PSMODE=COMPAT` 2. Se PROVIDERVERSION = 6 a. Impostare il gestore code su `PSMODE=ENABLED`

Pubblica

Invia una richiesta HTTP **POST** con l'URI:

```
http://hostname:port/context_root/msg/topic/topicString
```

Il contenuto del messaggio viene pubblicato utilizzando la stringa di argomenti *topicString*.

Sottoscrivi

Invia una richiesta HTTP **DELETE** con l'URI:

```
http://hostname:port/context_root/msg/topic/topicString
```

Il bridge WebSphere MQ per HTTP crea una sottoscrizione non durevole gestita alla stringa di argomenti *topicString*. La sottoscrizione viene eliminata non appena viene restituita una pubblicazione o fino alla scadenza dell'intervallo di attesa impostato dall'intestazione entità personalizzata, `x - msg - wait`.

Esecuzione del bridge WebSphere MQ per esempi HTTP

Il bridge WebSphere MQ per gli esempi HTTP è disponibile solo per il sistema Windows . Gli esempi mostrano come inoltrare i comandi HTTP **POST** e HTTP **DELETE** al bridge WebSphere MQ per HTTP dai programmi Java.

Prima di iniziare

Verificare il bridge WebSphere MQ per l'installazione HTTP eseguendo il passo "7" a pagina 1029 in "Installazione, configurazione e verifica del bridge WebSphere MQ per HTTP" a pagina 1028.

Gli esempi HTTP sono installati nelle directory mostrate in Tabella 156 a pagina 1033. In ogni caso, il codice sorgente è installato nella sottodirectory `/src` .

Tabella 156. Ubicazione degli esempi di HTTP

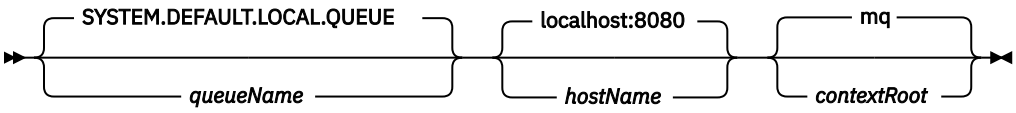
Piattaforma	**Ubicazione**
Finestre	*MQ_INSTALLATION_PATH*/tools/http/samples
Tutte le altre piattaforme	*MQ_INSTALLATION_PATH*/samp/http
MQ_INSTALLATION_PATH rappresenta la directory in cui è installato WebSphere MQ .	

Informazioni su questa attività

Gli esempi simulano le applicazioni di esempio WebSphere MQ AMQSPUT e AMQSGET. Illustrano le funzioni riportate di seguito in un ambiente di messaggistica point-to-point:

- **HTTPPOST** - Invia le richieste HTTP **POST** in una applicazione Java per inserire i messaggi in una coda WebSphere MQ , utilizzando il bridge WebSphere MQ per HTTP e gestisce le risposte.
- **HTTPDELETE** - Invia le richieste HTTP **DELETE** in una applicazione Java per ottenere i messaggi da una coda WebSphere MQ , utilizzando il bridge WebSphere MQ per HTTP e gestisce le risposte contenenti il messaggio WebSphere MQ .

Parametri per HTTPPOST e HTTPDELETE



Per eseguire l'esempio **HTTPPOST** , completare la seguente procedura:

Procedura

1. In una finestra comandi, passare alla directory degli esempi HTTP.
2. Eseguire l'esempio **HTTPPOST** .

   ```
   java -classpath . HTTPPOST [parameters]
   ```

   Quando l'esempio **HTTPPOST** viene avviato, viene visualizzato il seguente output:

```
HTTP POST Sample start
Target server is 'hostName'
Target queue is 'queueName'
Target context-root is 'contextRoot'
```

3. Nel prompt dei comandi, immettere il testo che si desidera per formare il corpo del messaggio.
4. Premere Invio per inviare il messaggio alla coda WebSphere MQ .
   a) Se si desidera inviare un altro messaggio, immettere un altro testo.
      Il testo forma il corpo di un secondo messaggio WebSphere MQ .
   b) Premere Invio per inviare il messaggio alla coda WebSphere MQ .
5. Premere due volte Invio per terminare **HTTPPOST**.
   Viene visualizzato il seguente output:

```
HTTP POST Sample end
```

Operazioni successive

L'esempio **HTTPDELETE** esegue una ricezione distruttiva di tutti i messaggi inseriti nella coda WebSphere MQ .

Eseguire l'esempio **HTTPDELETE** completando la seguente procedura:

1. In una finestra di comandi, passare a *MQ_INSTALLATION_PATH*/tools/samples.
   MQ_INSTALLATION_PATH rappresenta la directory in cui è installato WebSphere MQ .
2. Eseguire l'esempio **HTTPDELETE** .

```
java -classpath . HTTPPOST [parameters]
```

   Quando l'esempio **HTTPDELETE** viene avviato, viene visualizzato il seguente output:

```
HTTP DELETE Sample start
Target server is 'host:port'
Target queue is 'your queue name'
Target context-root is 'your context-root'
message
message
...
```

Considerazioni sulla sicurezza per il bridge WebSphere per HTTP

Le considerazioni sulla sicurezza Web standard si applicano all'autenticazione di un client browser Web. L'autorizzazione alle risorse WebSphere MQ è a livello dell'utente che esegue il servlet WebSphere Bridge per HTTP e non il client del browser Web individuale. La considerazione sulla sicurezza standard di WebSphere MQ si applica a WebSphere MQ.

I dati che fluiscono da un browser Web a un'applicazione WebSphere MQ utilizzando il bridge WebSphere per HTTP e indietro, impiegano tre passi:

Connessione client
Dal browser al WebSphere Bridge per HTTP su una connessione TCP/IP utilizzando HTTP.

Connessione dell'adattatore di risorse a WebSphere MQ
La connessione è da WebSphere Bridge for HTTP a WebSphere MQ gestore code. La connessione è una connessione client, su TCP/IP o una connessione bind WebSphere MQ locale. Una volta stabilita la connessione, la richiesta HTTP viene inserita in una coda locale standard o in una coda di trasmissione.

Dalla coda locale di WebSphere WebSphere MQ su uno o più canali, alla coda di destinazione.
Applicare tecniche standard per la protezione di code, argomenti, gestori code e canali.

La risposta esegue le operazioni in senso inverso.

Connessione client

Proteggere le connessioni tra i client HTTP e il server delle applicazioni utilizzando il contenitore Web. Utilizzare le tecniche del server HTTP standard, ad esempio l'utilizzo di HTTPS. Per informazioni, fare riferimento alla documentazione del server delle applicazioni.

Connessione dell'adattatore di risorse a WebSphere MQ

La connessione tra l'adattatore di risorse e il gestore code è autorizzata utilizzando un solo ID utente. Assegnare un singolo ID utente per identificare richieste da WebSphere Bridge for HTTP. L'ID utente deve avere autorizzazioni WebSphere MQ limitate solo per le risorse a cui gli utenti esterni devono avere accesso. È necessario autenticare il client effettivo separatamente e stabilire l'attendibilità per le successive interazioni con il client, utilizzando tecniche standard per la sicurezza Web.

Proteggere la connessione tra l'adattatore di risorse e il gestore code utilizzando l'unico ID utente. Limitare le autorizzazioni dell'ID utente a non più del necessario per leggere e scrivere messaggi in code e argomenti. Il WebSphere Bridge per HTTP è un punto di attacco tra Internet e la propria intranet.

Il modo in cui si protegge la connessione tra l'adattatore di risorse e WebSphere MQ dipende dall'adattatore di risorse specifico. Fare riferimento alla documentazione per l'adattatore di risorse.

Utilizzo di Component Object Model Interface (WebSphere MQ Automation Classes for ActiveX)

Le WebSphere MQ Automation Classes for ActiveX (MQAX) sono componenti ActiveX che forniscono classi che è possibile utilizzare nella propria applicazione per accedere a WebSphere MQ.

MQAX richiede un ambiente WebSphere MQ e un'applicazione WebSphere MQ corrispondente con cui comunicare.

Fornisce all'applicazione ActiveX la possibilità di eseguire transazioni e accedere ai dati su qualsiasi sistema aziendale a cui è possibile accedere tramite WebSphere MQ.

WebSphere MQ Automation Classes for ActiveX:

- Fornire l'accesso alle funzioni dell'API WebSphere MQ , consentendo l'interconnettività completa ad altre piattaforme WebSphere MQ .
- Conforme alle normali convenzioni previste per un componente ActiveX .
- Conforme al modello oggetto WebSphere MQ , disponibile anche per .NET, C + +, Java e LotusScript.

Vengono forniti esempi di starter MQAX. È possibile utilizzare questi esempi inizialmente per verificare che l'installazione di MQAX sia stata eseguita correttamente e che si disponga dell'ambiente WebSphere MQ di base. Gli esempi mostrano anche come utilizzare MQAX.

Script COM e ActiveX

Il COM (Component Object Model) è un modello di programmazione basato su oggetti definito da Microsoft. Specifica il modo in cui i componenti software possono essere forniti in un modo che consente loro di individuare e comunicare tra loro indipendentemente dal linguaggio del computer in cui sono scritti o dalla loro posizione.

ActiveX è una serie di tecnologie, basate su COM, che integra sviluppo di applicazioni, componenti riutilizzabili e tecnologie Internet sulle piattaforme Microsoft Windows . I componenti ActiveX forniscono interfacce a cui le applicazioni possono accedere dinamicamente. Un client di script ActiveX è un'applicazione, ad esempio un compilatore, che può creare o eseguire un programma o uno script che utilizza le interfacce fornite dai componenti ActiveX (o COM).

WebSphere MQ

WebSphere MQ Automation Classes for ActiveX può essere utilizzato solo con client di script **a 32 bit** ActiveX .

Il componente COM può essere utilizzato solo per applicazioni **a 32 bit** . Se si desidera scrivere l'applicazione COM a 64 bit, è possibile utilizzare l'interfaccia .NET.

Per eseguire MQAX in un ambiente server WebSphere MQ , è necessario che sul sistema sia installato Windows 2000 o versione successiva.

Per eseguire MQAX in un ambiente client WebSphere MQ MQI, è necessario che il client MQI WebSphere MQ su Windows 2000 o versioni successive sia installato sul sistema:

Il client WebSphere MQ MQI richiede l'accesso ad almeno un server WebSphere MQ . Quando il WebSphere MQ MQI client e il server WebSphere MQ sono installati sul proprio sistema, le applicazioni MQAX vengono sempre eseguite sul server. L'interfaccia ActiveX per MQAI è disponibile solo in ambienti server WebSphere MQ .

Progettazione e programmazione mediante WebSphere MQ Automation Classes for ActiveX

Progettazione di applicazioni MQAX che accedono ad applicazioni nonActiveX

Le classi di automazione WebSphere MQ forniscono accesso alle funzioni dell'API WebSphere MQ . È quindi possibile trarre vantaggio da tutti i vantaggi che l'uso di WebSphere MQ può apportare all'applicazione Windows .

La progettazione generale della propria applicazione è la stessa di qualsiasi applicazione WebSphere MQ , pertanto considerare tutti gli aspetti di progettazione descritti nella sezione “Sviluppo delle applicazioni” a pagina 7 .

Per utilizzare le classi di automazione WebSphere MQ , codificare i programmi Windows nell'applicazione utilizzando un linguaggio che supporta la creazione e l'utilizzo di oggetti COM. Ad esempio, Visual Basic, Java e altri client di script ActiveX . Le classi possono quindi essere facilmente integrate nell'applicazione perché gli oggetti WebSphere MQ necessari possono essere codificati utilizzando la sintassi nativa del linguaggio di implementazione.

Utilizzo di classi di automazione WebSphere MQ per ActiveX

Quando si progetta un'applicazione ActiveX che utilizza WebSphere MQ Automation Classes for ActiveX, l'elemento più importante di informazioni è il messaggio inviato o ricevuto dal sistema WebSphere MQ remoto. Pertanto, è necessario conoscere il formato degli elementi che vengono inseriti nel messaggio. Per uno script MQAX su un lavoro, sia esso che l'applicazione WebSphere MQ che preleva o invia il messaggio devono conoscere la struttura del messaggio.

Se si sta inviando un messaggio con un'applicazione MQAX e si desidera eseguire la conversione dei dati alla fine di MQAX, è necessario conoscere anche:

- La codepage utilizzata dal sistema remoto
- La codifica utilizzata dal sistema remoto

Per mantenere il tuo codice portatile, è buona norma impostare la codepage e la codifica, anche se attualmente sono uguali sia nel sistema di invio che in quello di ricezione.

Quando si considera come strutturare l'implementazione del sistema progettato, ricordare che gli script MQAX vengono eseguiti sulla stessa macchina su cui è installato il gestore code WebSphere MQ o il client WebSphere MQ .

Suggerimenti e suggerimenti per la programmazione

I seguenti suggerimenti e suggerimenti non sono in ordine significativo. Sono soggetti che, se rilevanti per il lavoro che stai facendo, potrebbero farti risparmiare tempo.

Proprietà descrittori messaggi

Se si gestiscono le proprietà del descrittore del messaggio in un programma, è preferibile utilizzare gli equivalenti esadecimali dei campi.

Le informazioni in questa sezione fanno riferimento alle proprietà seguenti:

- AccountingToken
- CorrelationId
- GroupId
- MessageId

Se un'applicazione WebSphere MQ è il creatore di un messaggio e WebSphere MQ genera tali proprietà, è preferibile utilizzare le proprietà esadecimale AccountingToken, CorrelationId, GroupIde MessageIdse si desidera visualizzarne i valori o modificarli in qualsiasi modo, includendo la loro trasmissione in un messaggio a WebSphere MQ. Il motivo di ciò è che i valori generati da WebSphere MQ sono stringhe di byte che hanno un valore compreso tra 0 e 255 inclusi, non sono stringhe di caratteri stampabili.

Quando lo script MQAX è il creatore di un messaggio, è possibile utilizzare le proprietà AccountingToken, CorrelationId, GroupIde MessageId o i loro equivalenti esadecimale.

Costanti WebSphere MQ

WebSphere Le costanti MQ vengono fornite come membri dell'enumerazione WebSphere MQ nella libreria MQAX200.

Costanti stringa WebSphere MQ

WebSphere MQ e le corrispondenti stringhe di caratteri.

Le costanti di stringa WebSphere MQ non sono disponibili quando si utilizza WebSphere MQ Automation Classes for ActiveX. È necessario utilizzare la stringa di caratteri esplicita per quelli mostrati nel seguente elenco e per tutti gli altri che potrebbero essere necessari. I comandi devono essere riempiti di otto caratteri utilizzando spazi:

MQFMT_NONE	" "
MMQFMT_ADMIN	"MQADMIN"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM"
MQFMT_CICS	"MQCICS"
MQFMT_COMMAND_1	"MQCMD1 "
MQFMT_COMMAND_2	"MQCMD2 "
MQFMT_DEAD_LETTER_HEADER	"MQDEAD"
INTESTAZIONE_DIST_MQFM	"DISTRIB.MQHT"
EVENTO MQFMT	"EVENTO"
IMS MQFMT	"MQIMS"
MQFMT_IMS_VAR_STRIsNG	"MQIMSVS"
MQFMT_MD_EXTENSIONE	"MQHMDE"
MQFMT_PCF	"MQPCF"
MQFMT_REF_MSG_HEADER	"MQHREF"
MQFMT_RF_HEADER	"MQHRF"
MQFMT_STRING	"MQSTR"

MQFMT_TRIGGER	"MQTRIG"
Intestazione MQFMT_WORK_INFO_HEADER	"MQHWIH"
MQFMT_XMIT_Q_HEADER	"MQXMIT"

Costanti stringa null

Le costanti WebSphere MQ , utilizzate per l'inizializzazione di quattro propriet ... MQMessage, MQMI_NONE (24 caratteri NULL), MQCI_NONE (24 caratteri NULL), MQGI_NONE (24 caratteri NULL) e MQACT_NONE (32 caratteri NULL), non sono supportate dalle WebSphere MQ classi di automazione per ActiveX. L'impostazione su stringhe vuote ha lo stesso effetto.

Ad esempio, per impostare i vari ID di un MQMessage su questi valori: *mymessage*.**MessageId** = "" *mymessage*.**CorrelationId** = "" *mymessage*.**AccountingToken** = ""

Ricezione di un messaggio da WebSphere MQ

Esistono diversi modi per ricevere un messaggio da WebSphere MQ:

- Eseguire il polling emettendo un GET seguito da un Wait, utilizzando la funzione Visual Basic TIMER.
- Immissione di un GET con l'opzione Attendi; specificare la durata dell'attesa impostando la proprietà WaitInterval . Considerare questa situazione quando, anche se si imposta il sistema per l'esecuzione in un ambiente a più thread, il software in esecuzione al momento potrebbe eseguire solo un singolo thread. In questo modo si evita il blocco indefinito del sistema.

  Gli altri thread non vengono influenzati. Tuttavia, se gli altri thread richiedono l'accesso a WebSphere MQ, è necessaria una seconda connessione a WebSphere MQ utilizzando ulteriori oggetti coda e gestore code MQAX.

L'emissione di un GET con l'opzione Attendi e l'impostazione di WaitInterval su MQWI_UNLIMITED determina il blocco del sistema fino al completamento della chiamata GET, se il processo è a thread singolo.

Utilizzo della conversione dati

Due forme di conversione dati sono supportate dalle classi di automazione WebSphere MQ per ActiveX - codifica numerica e conversione della serie di caratteri.

Codifica numerica

Se si imposta la proprietà MQMessage Encoding, i seguenti metodi vengono convertiti tra diversi sistemi di codifica numerica:

- metodo ReadDecimal2
- metodo ReadDecimal4
- metodo ReadDouble
- metodo ReadDouble4
- metodo ReadFloat
- metodo ReadInt2
- metodo ReadInt4
- metodo ReadLong
- metodo ReadShort
- metodo ReadUInt2
- metodo WriteDecimal2
- metodo WriteDecimal4
- metodo WriteDouble

- metodo WriteDouble4
- metodo WriteFloat
- metodo WriteInt2
- metodo WriteInt4
- metodo WriteLong
- Metodo WriteShort
- metodo WriteUInt2

La proprietà Codifica può essere impostata e interpretata utilizzando le costanti WebSphere MQ fornite. Figura 216 a pagina 1039 mostra un esempio di:

```
/* 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
```

Figura 216. Costanti WebSphere MQ fornite per la codifica

Ad esempio, per inviare un numero intero da un sistema Intel ad un sistema operativo System/390 nella codifica 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
```

Conversione serie di caratteri

La conversione della serie di caratteri è necessaria quando si invia un messaggio da un sistema ad un altro in cui le codepage sono diverse. La conversione della codepage è utilizzata da:

- metodo ReadString
- metodo ReadNullTerminatedString
- metodo WriteString
- metodo WriteNullTerminatedString
- Proprietà MessageData

È necessario impostare la proprietà MQMessage CharacterSet su un CCSID (character set value) supportato.

WebSphere MQ Automation Classes for ActiveX utilizza le tabelle di conversione per eseguire la conversione della serie di caratteri.

Ad esempio, per convertire automaticamente le stringhe nella codepage 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
```

Il metodo WriteString riceve i dati stringa ("Una stringa di caratteri" nell'esempio) come stringa Unicode. Converte quindi questi dati da Unicode nella codepage 437 utilizzando la tabella di conversione 34B001B5.TBL.

I caratteri nella stringa Unicode che non sono supportati dalla codepage 437 ricevono il carattere di sostituzione standard dalla codepage 437.

Allo stesso modo, quando si utilizza il metodo ReadString , il messaggio in entrata ha una serie di caratteri stabilita dal valore WebSphere MQ Message Descriptor (MQMD) ed è presente una conversione da questa codepage in Unicode prima che venga trasmesso nuovamente al proprio linguaggio di script.

Thread

WebSphere MQ Automation Classes for ActiveX implementa un modello a thread libero in cui gli oggetti possono essere utilizzati tra i thread.

Mentre MQAX consente l'utilizzo di oggetti MQQueue e MQQueueManager , WebSphere MQ non consente attualmente la condivisione di handle tra thread differenti.

I tentativi di utilizzarle su un altro thread generano un errore e WebSphere MQ restituisce un codice di ritorno di MQRC_HCONN_ERROR.

Nota: Esiste un solo oggetto MQSession per processo. L'utilizzo di MQSession CompletionCode e ReasonCode non è consigliato in ambienti a più thread. I valori di errore MQSession potrebbero essere sovrascritti da un secondo thread tra un errore generato e il controllo sul primo thread. I thread vengono serializzati per la durata di ogni chiamata del metodo o accesso alla proprietà. Quindi, l'immissione di Get con l'opzione Wait fa sì che altri thread che accedono agli oggetti MQAX vengano sospesi fino al termine dell'operazione.

Gestione degli errori

Queste informazioni descrivono le proprietà dell'oggetto MQAX, come funziona la gestione degli errori, le regole che descrivono come vengono gestite le eccezioni e come richiamare una proprietà.

Ogni oggetto MQAX include proprietà per conservare le informazioni sull'errore e un metodo per reimpostarle o cancellarle. Le proprietà sono:

- CompletionCode
- ReasonCode
- ReasonName

Il metodo è:

- Codici ClearError

Come funziona la gestione degli errori

Lo script MQAX o l'applicazione richiamano un metodo dell'oggetto MQAX oppure accedono o aggiornano una proprietà dell'oggetto MQAX:

1. ReasonCode e CompletionCode nell'oggetto interessato vengono aggiornati.
2. ReasonCode e CompletionCode nell'oggetto MQSession vengono aggiornati con le stesse informazioni.

   Nota: Consultare “Thread” a pagina 1040 per le limitazioni relative all'utilizzo dei codici di errore MQSession nelle applicazioni con thread.

   Se il CompletionCode è maggiore o uguale alla proprietà ExceptionThreshold di MQSession, MQAX genera un'eccezione (numero 32000). Utilizzarlo all'interno dello script utilizzando l'istruzione On Error (o equivalente) per elaborarlo.

3. Utilizzare la funzione Errore per richiamare la stringa di errore associata, che ha il formato:

```
MQAX: CompletionCode=xxx, ReasonCode=xxx, ReasonName=xxx
```

   Per ulteriori informazioni su come utilizzare le istruzioni On Error, consultare la documentazione per il linguaggio di script ActiveX .

   L'utilizzo di CompletionCode e ReasonCode nell'oggetto MQSession è utile per semplici gestori di errori.

   La proprietà ReasonName restituisce il nome simbolico WebSphere MQ per il valore corrente di ReasonCode.

Generazione di eccezioni

Le seguenti regole descrivono come vengono gestite le eccezioni:

- Ogni volta che una proprietà o un metodo imposta il codice di completamento su un valore maggiore o uguale alla soglia di eccezione (generalmente impostato su 2) viene generata un'eccezione.
- Tutte le chiamate di metodo e gli insiemi di proprietà impostano il codice di completamento.

Acquisizione di una proprietà

Questo è un caso particolare perché CompletionCode e ReasonCode non sono sempre aggiornati:

- Se una proprietà ha esito positivo, l'oggetto e l'oggetto MQSession ReasonCode e CompletionCode rimangono invariati.
- Se una proprietà ha esito negativo con un CompletionCode di avvertenza, ReasonCode e CompletionCode rimangono invariati.
- Se un richiamo di proprietà ha esito negativo con un CompletionCode di errore, ReasonCode e CompletionCode vengono aggiornati per riflettere i valori reali e l'elaborazione degli errori procede come descritto.

La classe MQSession ha un metodo *ReasonCodeName* che può essere utilizzato per sostituire un codice motivo WebSphere MQ con un nome simbolico. Ciò è particolarmente utile durante lo sviluppo di programmi in cui potrebbero verificarsi errori imprevisti. Tuttavia, il nome non è ideale per la presentazione agli utenti.

Ogni classe ha anche una proprietà *ReasonName*, che restituisce il nome simbolico del codice motivo corrente per tale classe.

WebSphere MQ Automation Classes for ActiveX reference

Questa sezione descrive le classi delle classi di automazione WebSphere MQ per ActiveX (MQAX), sviluppate per ActiveX. Le classi consentono di scrivere applicazioni ActiveX che possono accedere ad altre applicazioni in esecuzione negli ambienti nonActiveX , utilizzando WebSphere MQ.

WebSphere MQ Automation Classes per l'interfaccia ActiveX

WebSphere MQ Automation Classes for ActiveX fornisce costanti ActiveX numeriche predefinite (come MQMT_REQUEST) necessarie per utilizzare le classi.

Le classi di automazione ActiveX sono costituite da:

- “Classe MQSession” a pagina 1043
- “Classe MQQueueManager” a pagina 1046
- “Classe MQQueue” a pagina 1057
- “Classe MQMessage” a pagina 1072
- “classe di opzioni MQPutMessage” a pagina 1093

- “classe di opzioni MQGetMessage” a pagina 1095
- “Classe MQDistributionList” a pagina 1097
- “Classe di elementi MQDistributionList” a pagina 1102

Inoltre, WebSphere MQ Automation Classes for ActiveX fornisce costanti ActiveX numeriche predefinite (come MQMT_REQUEST) necessarie per utilizzare le classi. Questi sono forniti nell'enumerazione MQ nella libreria MQAX200. Le costanti sono un sottoinsieme di quelle definite nei file di intestazione C WebSphere MQ (cmqc * .h) con alcune classi di automazione WebSphere MQ aggiuntive per codici di errore ActiveX .

Informazioni sulle classi WebSphere MQ Automation per classi ActiveX

Leggere queste informazioni insieme agli argomenti di riferimento in Sviluppo di applicazioni di riferimento.

Per informazioni importanti, consultare Funzioni che possono essere utilizzate solo con l'installazione primaria su Windows .

La classe MQSession fornisce un oggetto root che contiene lo stato dell'ultima azione eseguita su uno qualsiasi degli oggetti MQAX. Per ulteriori informazioni, fare riferimento a “Gestione degli errori” a pagina 1040.

Le classi MQQueueManager e MQQueue forniscono accesso agli oggetti WebSphere MQ sottostanti. Gli accessi ai metodi o alle proprietà per queste classi in generale comportano chiamate effettuate in WebSphere MQ MQI.

Le classi MQMessage, MQPutMessageOptions e MQGetMessageOptions incapsulano le strutture di dati MQMD, MQPMO e MQGMO e vengono utilizzate per inviare i messaggi alle code e richiamare i messaggi da esse.

La classe MQDistributionList incapsula una raccolta di code - locali, remote o alias per l'output. La classe di elementi MQDistributionListincapsula le strutture MQOR, MQRR e MQPMR e le associa a un elenco di distribuzione proprietario.

Passaggio parametro

I parametri sui richiami del metodo vengono tutti passati per valore, tranne dove quel parametro è un oggetto, nel qual caso è un riferimento che viene passato.

Le definizioni di classe fornite elencano il tipo di dati per ogni parametro o proprietà. Per molti client ActiveX , come Visual Basic, se la variabile utilizzata non è del tipo richiesto, il valore viene automaticamente convertito in o dal tipo richiesto, a condizione che tale conversione sia possibile. Ciò segue le regole standard del client; MQAX non fornisce tale conversione.

Molti dei metodi utilizzano parametri di stringa a lunghezza fissa o restituiscono una stringa di caratteri a lunghezza fissa. Le regole di conversione sono le seguenti:

- Se l'utente fornisce una stringa a lunghezza fissa di lunghezza errata, come parametro di input o come valore di ritorno, il valore viene troncato o riempito con spazi finali come richiesto.
- Se l'utente fornisce una stringa a lunghezza variabile di lunghezza non corretta come parametro di input, il valore viene troncato o riempito con spazi finali.
- Se l'utente fornisce una stringa a lunghezza variabile di lunghezza non corretta come valore di ritorno, la stringa viene regolata sulla lunghezza richiesta (poiché la restituzione di un valore distrugge comunque il valore precedente nella stringa).
- Le stringhe fornite come parametri di input possono contenere valori null incorporati.

Queste classi possono essere trovate nella libreria MQAX200 .

Metodi di accesso agli oggetti

Questi metodi non si riferiscono direttamente a una singola chiamata WebSphere MQ . Ciascuno di questi metodi crea un oggetto in cui vengono conservate le informazioni di riferimento, seguito dalla connessione o dall'apertura di un oggetto WebSphere MQ :

Quando viene effettuata una connessione a un gestore code, contiene l'attributo 'connection handle' generato da WebSphere MQ.

Quando una coda viene aperta, contiene l'attributo 'object handle' generato da WebSphere MQ.

Questi WebSphere MQ attributi non sono direttamente disponibili per il programma MQAX.

Errori

Gli errori sintattici sul passaggio dei parametri possono essere rilevati al momento della compilazione e al runtime dal client ActiveX . Gli errori possono essere rilevati utilizzando On Error in Visual Basic.

Le classi WebSphere MQ ActiveX contengono tutte due proprietà speciali di sola lettura: ReasonCode e CompletionCode. Queste proprietà possono essere lette in qualsiasi momento.

Un tentativo di accedere a qualsiasi altra proprietà o di emettere una chiamata di metodo potrebbe generare un errore da WebSphere MQ.

Se una serie di proprietà o un richiamo del metodo ha esito positivo, il ReasonCode dell'oggetto proprietario è impostato su MQRC_NONE e CompletionCode è impostato su MQCC_OK.

Se l'accesso alla proprietà o il richiamo del metodo non riesce, i codici motivo e di completamento vengono impostati in questi campi.

Classe MQSession

Questa è la classe root per WebSphere MQ Automation Classes for ActiveX.

È sempre presente un solo oggetto MQSession per processo client ActiveX . Un tentativo di creare un secondo oggetto crea un secondo riferimento all'oggetto originale.

Creazione

Nuovo crea un nuovo oggetto MQSession.

Sintassi

Dim *mqsess* **As New MQSession Set** *mqsess* = **New MQSession**

Proprietà

- “proprietà CompletionCode” a pagina 1044.
- “proprietà ExceptionThreshold” a pagina 1044.
- “proprietà ReasonCode” a pagina 1044.
- “proprietà ReasonName” a pagina 1045.

Metodo

- “Metodo AccessGetMessageOptions” a pagina 1045.
- “metodo AccessMessage” a pagina 1045.
- “metodo AccessPutMessageOptions” a pagina 1045.
- “metodo Gestore AccessQueue” a pagina 1045.
- “Metodo di codici ClearError” a pagina 1046.

- “metodo Nome ReasonCode” a pagina 1046.

proprietà CompletionCode

Sola lettura. Restituisce il codice di completamento WebSphere MQ impostato dal metodo o dalla serie di proprietà più recenti emessi rispetto a qualsiasi oggetto WebSphere MQ .

Viene reimpostato su MQCC_OK quando un metodo o una serie di proprietà viene richiamata correttamente rispetto a qualsiasi oggetto MQAX.

Un gestore eventi di errore può esaminare questa proprietà per diagnosticare l'errore, senza dover conoscere quale oggetto è stato coinvolto.

L'utilizzo di CompletionCode e ReasonCode nell'oggetto MQSession è molto conveniente per semplici gestori di errori.

Nota: Consultare “Thread” a pagina 1040 per le limitazioni sull'utilizzo dei codici di errore MQSession nelle applicazioni con thread.

Definito in: classe MQSession

Tipo di dati: Long

Valori:

- MQCC_OK
- MQCC_AVVERTENZA
- MQCC_NON RIUSCITO

Sintassi:

Per ottenere: *completioncode & = MQSession*.CompletionCode

proprietà ExceptionThreshold

Lettura - scrittura. Definisce il livello di errore WebSphere MQ per cui MQAX genererà un'eccezione. Il valore predefinito è MQCC_FAILED. Un valore maggiore di MQCC_FAILED impedisce effettivamente l'elaborazione delle eccezioni, lasciando che il programmatore esegua i controlli su CompletionCode e ReasonCode.

Definito in: classe MQSession

Tipo di dati: Long

Valori:

- Qualsiasi, ma considerare MQCC_WARNING, MQCC_FAILED o superiore.

Sintassi:

Per ottenere: *ExceptionThreshold& = MQSession*.**ExceptionThreshold**

Per impostare: *MQSession*.**ExceptionThreshold** = *ExceptionThreshold$*

proprietà ReasonCode

Sola lettura. Restituisce il codice motivo impostato dal metodo più recente o dalla serie di proprietà emessa rispetto a qualsiasi oggetto WebSphere MQ .

Un gestore eventi di errore può esaminare questa proprietà per diagnosticare l'errore, senza dover conoscere quale oggetto è stato coinvolto.

L'utilizzo di CompletionCode e ReasonCode nell'oggetto MQSession è molto conveniente per semplici gestori di errori.

Nota: Consultare “Thread” a pagina 1040 per le limitazioni sull'utilizzo dei codici di errore MQSession nelle applicazioni con thread.

Definito in: classe MQSession

Tipo di dati: Long

Valori:

- Consultare Motivo (MQLONG) e i valori MQAX aggiuntivi elencati in “Codici di origine” a pagina 1109.

Sintassi: per ottenere *reasoncode & = MQSession*.**ReasonCode**

proprietà ReasonName

Sola lettura. Restituisce il nome simbolico dell'ultimo codice motivo. Ad esempio, "MQRC_QMGR_NOT_AVAILABLE".

Nota: Consultare “Thread” a pagina 1040 per le limitazioni sull'utilizzo dei codici di errore MQSession nelle applicazioni con thread.

Definito in: classe MQSession

Tipo di dati: stringa

Valori:

- Vedere codici di errore API.

Sintassi: per ottenere: *reasonname $= MQSession*.**ReasonName**

Metodo AccessGetMessageOptions

Crea un nuovo oggetto MQGetMessageOptions.

Definito in: classe MQSession

Sintassi: *gmo = MQSession*.**AccessGetMessageOptions()**

metodo AccessMessage

Crea un nuovo oggetto MQMessage.

Definito in: classe MQSession

Sintassi: *msg = MQSession*.**AccessMessage()**

metodo AccessPutMessageOptions

Crea un nuovo oggetto MQPutMessageOptions.

Definito in: classe MQSession

Sintassi: *pmo = MQSession*.**AccessPutMessageOptions()**

metodo Gestore AccessQueue

Crea un nuovo oggetto MQQueueManager e lo connette a un gestore code reale mediante il server WebSphere MQ o il client WebSphere MQ MQI. Oltre all'esecuzione di una connessione, questo metodo esegue anche un'apertura per l'oggetto gestore code.

Quando il client WebSphere MQ MQI e il server WebSphere MQ sono installati sul sistema, per impostazione predefinita le applicazioni MQAX verranno eseguite sul server. Per eseguire MQAX sul client, la libreria di bind client deve essere specificata nella variabile di ambiente `GMQ_MQ_LIB` , ad esempio, impostare `GMQ_MQ_LIB=mqic.dll`.

Per un'installazione solo client, non è necessario impostare la variabile di ambiente `GMQ_MQ_LIB` . Quando questa variabile non è impostata, WebSphere MQ tenta di caricare amqzst.dll. Se questa DLL non è presente (come nel caso di un'installazione solo client), WebSphere MQ tenta di caricare mqic.dll.

Se ha esito positivo, imposta ConnectionStatus di MQQueueManagersu TRUE.

Un gestore code può essere connesso al massimo a un oggetto MQQueueManager per istanza ActiveX .

Se la connessione al gestore code ha esito negativo, viene generato un evento di errore e vengono impostati ReasonCode e CompletionCode dell'oggetto MQSession.

Definito in: classe MQSession

Sintassi: *set qm = MQSession*.**AccessQueueManager** (*Nome$*)

Parametro:*Nome$* Stringa. Nome del gestore code a cui connettersi.

Metodo di codici ClearError

Reimposta CompletionCode su MQCC_OK e ReasonCode su MQRC_NONE.

Definito in: classe MQSession

Sintassi: Call *MQSession*.**ClearErrorCodes ()**

metodo Nome ReasonCode

Restituisce il nome del codice motivo con il valore numerico fornito. È utile fornire agli utenti indicazioni più chiare circa le condizioni di errore. Il nome è ancora un po' criptico (ad esempio, ReasonCodeName (2059) is **MQRC_Q_MGR_NOT_AVAILABLE**), quindi dove possibili errori devono essere rilevati e sostituiti con testo descrittivo appropriato per l'applicazione.

Definito in: classe MQSession

Sintassi: *errname $= MQSession*.**ReasonCodeName**(*reasonCode&*)

Parametro*reasoncode &* Long. Il codice di errore per cui è richiesto il nome simbolico.

Classe MQQueueManager

Questa classe rappresenta una connessione a un gestore code. Il gestore code può essere in esecuzione localmente (un WebSphere MQ server) o in remoto con l'accesso fornito dal client WebSphere MQ . Un'applicazione deve creare un oggetto di questa classe e connetterlo a un gestore code. Quando un oggetto di questa classe viene eliminato, viene automaticamente disconnesso dal gestore code.

Contenimento

Gli oggetti della classe MQQueue sono associati a questa classe.

Nuovo crea un nuovo oggetto di MQQueueManager e imposta tutte le proprietà sui valori iniziali. In alternativa, utilizzare il metodo AccessQueueManager della classe MQSession.

Creazione

Nuovo crea un oggetto **nuovo** MQQueueManager e imposta tutte le proprietà sui valori iniziali. In alternativa, utilizzare il metodo AccessQueueManager della classe MQSession.

Sintassi

Dim *mgr* **As New MQQueueManager set** *mgr* = **New MQQueueManager**

Proprietà

- “proprietà ID AlternateUser” a pagina 1048.
- “proprietà AuthorityEvent” a pagina 1048.
- “proprietà BeginOptions” a pagina 1048.
- “proprietà Definizione ChannelAuto” a pagina 1049.
- “proprietà ChannelAutoDefinitionEvent” a pagina 1049.
- “proprietà ChannelAutoDefinitionExit” a pagina 1049.

- “proprietà CharacterSet” a pagina 1049.
- “proprietà CloseOptions” a pagina 1049.
- “proprietà CommandInputQueueName” a pagina 1050.
- “proprietà CommandLevel” a pagina 1050.
- “proprietà CompletionCode” a pagina 1050.
- “proprietà ConnectionHandle” a pagina 1050.
- “proprietà ConnectionStatus” a pagina 1050.
- “proprietà ConnectOptions” a pagina 1051.
- “proprietà DeadLetterQueueName” a pagina 1051.
- “Proprietà DefaultTransmissionQueueName” a pagina 1051.
- “proprietà Descrizione” a pagina 1051.
- “proprietà DistributionLists” a pagina 1051.
- “proprietà InhibitEvent” a pagina 1052.
- “proprietà IsConnected” a pagina 1052.
- “proprietà IsOpen” a pagina 1052.
- “proprietà LocalEvent” a pagina 1052.
- “proprietà MaximumHandles” a pagina 1052.
- “proprietà Lunghezza MaximumMessage” a pagina 1053.
- “proprietà MaximumPriority” a pagina 1053.
- “proprietà Messaggi MaximumUncommitted” a pagina 1053.
- “proprietà Nome” a pagina 1053.
- “proprietà ObjectHandle” a pagina 1053.
- “proprietà PerformanceEvent” a pagina 1053.
- “Proprietà della piattaforma” a pagina 1054.
- “proprietà ReasonCode” a pagina 1054.
- “proprietà ReasonName” a pagina 1054.
- “proprietà RemoteEvent” a pagina 1054.
- “proprietà Evento StartStop” a pagina 1054.
- “proprietà Disponibilità SyncPoint” a pagina 1055.
- “proprietà TriggerInterval” a pagina 1055.

Metodi

- “Metodo AccessQueue” a pagina 1055.
- “metodo AddDistributionList” a pagina 1056.
- “Metodo backout” a pagina 1056.
- “Metodo di inizio” a pagina 1056.
- “Metodo di codici ClearError” a pagina 1056.
- “Metodo commit” a pagina 1056.
- “Metodo CONNECT” a pagina 1057.
- “Metodo di disconnessione” a pagina 1057.

Accesso proprietà

È possibile accedere alle seguenti proprietà in qualsiasi momento.

- “proprietà ID AlternateUser” a pagina 1048.
- “proprietà CompletionCode” a pagina 1050.
- “proprietà ConnectionStatus” a pagina 1050.
- “proprietà ReasonCode” a pagina 1054.

È possibile accedere alle proprietà rimanenti solo se l'oggetto è connesso a un gestore code e l'ID utente è autorizzato a interrogare tale gestore code. Se un ID utente alternativo è impostato e l'ID utente corrente è autorizzato a utilizzarlo, l'ID utente alternativo viene controllato per l'autorizzazione per l'interrogazione

Se queste condizioni non si applicano, WebSphere MQ Automation Classes for ActiveX tenta di connettersi al gestore code e di aprirlo per l'interrogazione automatica. Se l'operazione ha esito negativo, la chiamata imposta un CompletionCode di MQCC_FAILED e uno dei seguenti ReasonCodes:

- MQRC_CONNECTION_BROKEN
- MQRC_NOT_AUTHORIZED
- ERRORE MQRC_Q_MGR_NAME_
- MQRC_Q_MGR_NOT_AVAILABLE

proprietà ID AlternateUser

Lettura - scrittura. L'ID dell'utente alternativo da utilizzare per convalidare l'accesso agli attributi del gestore code.

Questa proprietà non deve essere impostata se IsConnected è TRUE.

Questa proprietà non può essere impostata mentre l'oggetto è aperto.

classe **Defined in:** MQQueueManager

Data Type: Stringa di 12 caratteri

Syntax: Per ottenere: *altuser $= MQQueueManager*.**AlternateUserID** Per impostare: *MQQueueManager*.**AlternateUserID** = *altuser $*

proprietà AuthorityEvent

Sola lettura. L'attributo AuthorityEvent MQI.

Definito in:
Classe MQQueueManager

Tipo di dati:
Lungo

Valori:

- DISABILITAZIONE_MQEVR
- MQEVR_ENABLED

Sintassi: per ottenere: *authevent = MQQueueManager*.**AuthorityEvent**

proprietà BeginOptions

Lettura - scrittura. Queste sono le opzioni che si applicano al metodo Begin. Inizialmente MQBO_NONE.

Definito in:
Classe MQQueueManager

Tipo di dati:
Lungo

Valori:

- MQBO_NONE

Sintassi: per ottenere: *beginoptions & =MQQueueManager*.**BeginOptions**

Per impostare: *MQQueueManager***.BeginOptions**=*beginoptions &*

proprietà Definizione ChannelAuto

Sola lettura. Controlla se è consentita la definizione automatica del canale.

Definito in:
Classe MQQueueManager

Tipo di dati:
Lungo

Valori:

- DISABLE_MQCHAD
- ENABLE_MQCHAD

Sintassi: Per ottenere *channelautodef & = MQQueueManager*.**ChannelAutoDefinition**

proprietà ChannelAutoDefinitionEvent

Sola lettura. Controlla se vengono generati eventi di definizione di canale automatici.

Definito in:
Classe MQQueueManager

Tipo di dati:
Lungo

Valori:

- DISABILITAZIONE_MQEVR
- MQEVR_ENABLED

Sintassi: Per ottenere: *channelautodefevent & =MQQueueManager*.**ChannelAutoDefinitionEvent**

proprietà ChannelAutoDefinitionExit

Sola lettura. Il nome dell'uscita utente utilizzato per la definizione di canale automatica.

Definito in:
Classe MQQueueManager

Tipo di dati:
Stringa

Sintassi: Per ottenere *channelautodefexit$= MQQueueManager*.**ChannelAutoDefinitionExit**

proprietà CharacterSet

Sola lettura. L'attributo MQI CodedCharSetId .

Definito in: classe MQQueueManager

Tipo di dati: Long

Sintassi: per richiamare: *characterset & = MQQueueManager***.CharacterSet**

proprietà CloseOptions

Lettura - scrittura. Opzioni utilizzate per controllare cosa accade quando il gestore code viene chiuso. Il valore iniziale è MQCO_NONE.

Definito in:
Classe MQQueueManager

Tipo di dati:
Lungo

Valori:

- MQCO_NONE

Sintassi: per ottenere: *closeopt & = MQQueueManager*.**CloseOptions**

Per impostare: *MQQueueManager*.**CloseOptions** =*closeopt &*

proprietà CommandInputQueueName

Sola lettura. L'attributo MQI CommandInputQName.

Definito in: classe MQQueueManager

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere *commandinputqname $= MQQueueManager*.**CommandInputQueueName**

proprietà CommandLevel

Sola lettura. Restituisce la versione e il livello dell'implementazione del gestore code WebSphere MQ (attributo MQI CommandLevel)

Definito in: classe MQQueueManager

Tipo di dati: Long

Sintassi: per ottenere: *level & = MQQueueManager*.**CommandLevel**

proprietà CompletionCode

Sola lettura. Restituisce il codice di completamento impostato dall'ultimo metodo o accesso alla proprietà emesso rispetto all'oggetto.

Definito in: classe MQQueueManager

Tipo di dati: Long

Valori:

- MQCC_OK
- MQCC_AVVERTENZA
- MQCC_NON RIUSCITO

Sintassi: per ottenere: *completioncode & = MQQueueManager*.**CompletionCode**

proprietà ConnectionHandle

Sola lettura. L'handle di connessione per l'oggetto gestore code WebSphere MQ .

Definito in:
Classe MQQueueManager

Tipo di dati:
Lungo

Sintassi: per ottenere: *hconn & = MQQueueManager.***ConnectionHandle**

proprietà ConnectionStatus

Sola lettura. Indica se l'oggetto è connesso o meno al relativo gestore code.

Definito in: classe MQQueueManager

Tipo di dati : booleano

Valori:

- VERO (-1)
- FALSE (0)

Sintassi: per ottenere *status = MQQueueManager*.**ConnectionStatus**

proprietà ConnectOptions

Lettura - scrittura. Queste opzioni si applicano al metodo Connect. Inizialmente MQCNO_NONE.

Definito in:
Classe MQQueueManager

Tipo di dati:
Lungo

Valori:

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING
- MQCNO_NONE

Sintassi: per ottenere: *connectoptions & =MQQueueManager*.**ConnectOptions**

Per impostare: *MQQueueManager*.**ConnectOptions**=*connectoptions &*

proprietà DeadLetterQueueName

Sola lettura. L'attributo MQI DeadLetterQName.

Definito in: classe MQQueueManager

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere: *dlqname $= MQQueueManager*.**DeadLetterQueueName**

Proprietà DefaultTransmissionQueueName

Sola lettura. L'attributo MQI DefXmitQName.

Definito in: classe MQQueueManager

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere: *defxmitqname $= MQQueueManager*.**DefaultTransmissionQueueName**

proprietà Descrizione

Sola lettura. L'attributo QMgrDesc MQI.

Definito in: classe MQQueueManager

Tipo di dati: stringa di 64 caratteri

Sintassi: per ottenere *description $= MQQueueManager*.**Descrizione**

proprietà DistributionLists

Sola lettura. Questa è la funzionalità del gestore code per supportare gli elenchi di distribuzione.

Definito in:
Classe MQQueueManager

Tipo di dati:
Booleano

Valori:

- VERO (-1)
- FALSE (0)

Sintassi: Per ottenere *distributionlists= MQQueueManager*.**DistributionLists**

proprietà InhibitEvent

Sola lettura. L'attributo InhibitEvent MQI.

Definito in: classe MQQueueManager

Tipo di dati: Long

Valori:

- DISABILITAZIONE_MQEVR
- MQEVR_ENABLED

Sintassi: per ottenere: *inievent & = MQQueueManager*.**InhibitEvent**

proprietà IsConnected

Un valore che indica se il gestore code è attualmente connesso.

Sola lettura.

Definito in: classe MQQueueManager

Tipo di dati : booleano

Valori:

- VERO (-1)
- FALSE (0)

Sintassi: per ottenere *isconnected = MQQueueManager*.**IsConnected**

proprietà IsOpen

Un valore che indica se il gestore code è attualmente aperto per l'interrogazione.

Sola lettura.

Definito in:
Classe MQQueueManager

Tipo di dati:
Booleano

Valori:

- VERO (-1)
- FALSE (0)

Sintassi : *IsOpen = MQQueueManager*.**IsOpen**

proprietà LocalEvent

Sola lettura. L'attributo LocalEvent MQI.

Definito in: classe MQQueueManager

Tipo di dati: Long

Valori:

- DISABILITAZIONE_MQEVR
- MQEVR_ENABLED

Sintassi: per ottenere *localevent & = MQQueueManager*.**LocalEvent**

proprietà MaximumHandles

Sola lettura. L'attributo MQI MaxHandles .

Definito in: classe MQQueueManager

Tipo di dati: Long

Sintassi: Per ottenere *maxhandle & = MQQueueManager*.**MaximumHandles**

proprietà Lunghezza MaximumMessage

Sola lettura. L'attributo MQI MaxMsgLength Queue Manager.

Definito in: classe MQQueueManager

Tipo di dati: Long

Sintassi: per ottenere: *maxmessagelength & = MQQueueManager*.**MaximumMessageLength**

proprietà MaximumPriority

Sola lettura. L'attributo MaxPriority MQI.

Definito in: classe MQQueueManager

Tipo di dati: Long

Sintassi: per ottenere: *maxpriority & = MQQueueManager*.**MaximumPriority**

proprietà Messaggi MaximumUncommitted

Sola lettura. L'attributo MQI MaxUncommittedMsgs.

Definito in: classe MQQueueManager

Tipo di dati: Long

Sintassi: per ottenere: *maxuncommitted & = MQQueueManager*.**MaximumUncommittedMessages**

proprietà Nome

Lettura - scrittura. L'attributo QMgrName MQI. Questa proprietà non può essere scritta una volta che MQQueueManager è connesso.

Definito in: classe MQQueueManager

Tipo di dati: stringa di 48 caratteri

Sintassi: per richiamare: *name $= MQQueueManager*.**name**

Per impostare: *MQQueueManager*.**name** = *name $*

Nota: Visual Basic riserva la proprietà "Nome" per l'utilizzo nell'interfaccia visiva. Pertanto, quando si utilizza Visual Basic, utilizzare le lettere minuscole, ossia "name".

proprietà ObjectHandle

Sola lettura. L'handle di oggetti per l'oggetto gestore code WebSphere MQ .

Definito in:
Classe MQQueueManager

Tipo dati
Lungo

Sintassi: per ottenere: *hobj & = MQQueueManager.***ObjectHandle**

proprietà PerformanceEvent

Sola lettura. L'attributo PerformanceEvent MQI.

Definito in: classe MQQueueManager

Tipo di dati: Long

Valori:

- DISABILITAZIONE_MQEVR
- MQEVR_ENABLED

Sintassi: per ottenere: *perfevent & = MQQueueManager*.PerformanceEvent

Proprietà della piattaforma

Sola lettura. L'attributo MQI Platform.

Definito in: classe MQQueueManager

Tipo di dati: Long

Valori:

- MQPL_WINDOWS_NT
- WINDOWS MQPL

Sintassi: per ottenere *platform & = MQQueueManager***.Piattaforma**

proprietà ReasonCode

Sola lettura. Restituisce il codice motivo impostato dall'ultimo metodo o accesso alla proprietà emesso per l'oggetto.

Definito in: classe MQQueueManager

Tipo di dati: Long

Valori:

- Vedere codici di errore API.

Sintassi: Per ottenere: *reasoncode & = MQQueueManager***.ReasonCode**

proprietà ReasonName

Sola lettura. Restituisce il nome simbolico dell'ultimo codice motivo. Ad esempio, "MQRC_QMGR_NOT_AVAILABLE".

Definito in: classe MQQueueManager

Tipo di dati: stringa

Valori:

- Vedere codici di errore API.

Sintassi: per richiamare *reasonname $= MQQueueManager***.ReasonName**

proprietà RemoteEvent

Sola lettura. L'attributo RemoteEvent MQI.

Definito in: classe MQQueueManager

Tipo di dati: Long

Valori:

- DISABILITAZIONE_MQEVR
- MQEVR_ENABLED

Sintassi: per ottenere: *remoteevent & = MQQueueManager***.RemoteEvent**

proprietà Evento StartStop

Sola lettura. L'attributo evento StartStopMQI.

Definito in: classe MQQueueManager

Tipo di dati: Long

Valori:

- DISABILITAZIONE_MQEVR
- MQEVR_ENABLED

Sintassi: per ottenere: *strstpevent & = MQQueueManager*.**StartStopEvent**

proprietà Disponibilità SyncPoint

Sola lettura. L'attributo SyncPoint MQI.

Definito in: classe MQQueueManager

Tipo di dati: Long

Valori:

- MQSP_DISPONIBILE
- MQSP_NON_DISPONIBILE

Sintassi: per ottenere: *syncpointavailability & = MQQueueManager*.**SyncPointAvailability**

proprietà TriggerInterval

Sola lettura. L'attributo TriggerInterval MQI.

Definito in: classe MQQueueManager

Tipo di dati: Long

Sintassi: Per ottenere: *trigint & = MQQueueManager*.**TriggerInterval**

Metodo AccessQueue

Crea un oggetto MQQueue e lo associa a questo oggetto MQQueueManager impostando la proprietà di riferimento della connessione della coda. Imposta le proprietà Name, OpenOptions, DynamicQueueName e AlternateUserId dell'oggetto MQQueue sui valori forniti e tenta di aprirlo.

Se l'apertura ha esito negativo, la chiamata ha esito negativo. Viene generato un evento di errore sull'oggetto. ReasonCode e CompletionCodee MQSession ReasonCode e CompletionCode dell'oggetto sono impostati.

I parametri DynamicQueueName, QueueManagerName e AlternateUserId sono facoltativi e per impostazione predefinita sono "".

È necessario specificare OpenOption MQOO_INQUIRE in aggiunta ad altre opzioni se le proprietà della coda devono essere lette.

Non impostare il nome QueueManagero impostarlo su "" se la coda da aprire è locale. Altrimenti, impostarlo sul nome del gestore code remoto che possiede la coda e si tenta di aprire una definizione locale della coda remota. Per ulteriori informazioni sulla risoluzione del nome della coda remota e sull'alias del gestore code, consultare Cosa sono gli alias? .

Se la proprietà Nome è impostata su un nome coda modello, specificare il nome della coda dinamica da creare nel parametro DynamicQueueName$. Se il valore fornito nel parametro DynamicQueueName$ è "", il valore impostato nell'oggetto coda e utilizzato nella chiamata aperta è "AMQ. *". Consultare “Creazione di code dinamiche” a pagina 224 per ulteriori informazioni sulla denominazione delle code dinamiche.

Definizione

Definita in: classe MQQueueManager .

Sintassi

Sintassi: set queue = MQQueueManager.**AccessQueue**(Name$, OpenOptions&, QueueManagerName$, DynamicQueueName$, AlternateUserId$)

Parametri

Stringa *Name$* . Nome della coda WebSphere MQ .

OpenOptions: lungo. Opzioni da utilizzare quando la coda è aperta. Vedere OpenOptions (MQLONG).

Stringa *QueueManagerName$* . Nome del gestore code proprietario della coda da aprire. Il valore "" implica che il gestore code è locale.

Stringa *DynamicQueueName$* . Il nome assegnato alla coda dinamica nel momento in cui la coda viene aperta quando il parametro Nome $specifica una coda modello.

Stringa *AlternateUserId$* . L'ID utente alternativo utilizzato per convalidare l'accesso quando si apre la coda.

metodo AddDistributionList

Crea un nuovo oggetto MQDistributionList e imposta il suo riferimento di connessione sul gestore code proprietario.

Definito in:
Classe MQQueueManager

Sintassi: *set distributionlist* = **MQQueueManager.***AddDistributionList*

Metodo backout

Esegue il backout di eventuali inserimenti e richiami di messaggi non sottoposti a commit che si sono verificati come parte di un'unità di lavoro dall'ultimo punto di sincronizzazione.

Definito in: classe MQQueueManager

Sintassi: Call *MQQueueManager.***Backout**

Metodo di inizio

Avvia un'unità di lavoro coordinata dal gestore code. Le opzioni di inizio influiscono sul comportamento di questo metodo.

Definito in:
Classe MQQueueManager

Sintassi: Call *MQQueueManager*.**Begin ()**

Metodo di codici ClearError

Reimposta CompletionCode su MQCC_OK e ReasonCode su MQRC_NONE per la classe MQQueueManager e la classe MQSession.

Definito in: classe MQQueueManager

Sintassi: Call *MQQueueManager*.**ClearErrorCodes ()**

Metodo commit

Esegue il commit di eventuali inserimenti e richiami di messaggi che si sono verificati come parte di un'unità di lavoro dall'ultimo punto di sincronizzazione.

Definito in: classe MQQueueManager

Sintassi: Call *MQQueueManager*.**Commit ()**

Metodo CONNECT

Connette l'oggetto MQQueueManager a un gestore code reale tramite il client o il server WebSphere MQ MQI. Oltre a stabilire la connessione, questo metodo apre anche l'oggetto gestore code in modo che possa essere sottoposto a query.

Imposta IsConnected su TRUE.

A un massimo di un oggetto MQQueueManager per istanza ActiveX è consentito connettersi a un gestore code.

Definito in: classe MQQueueManager

Sintassi: Call *MQQueueManager*.**Connetti ()**

Metodo di disconnessione

Disconnette l'oggetto MQQueueManager dal gestore code.

Imposta IsConnected su FALSE.

Tutti gli oggetti coda associati all'oggetto MQQueueManager vengono resi inutilizzabili e non possono essere riaperti.

Tutte le modifiche non sottoposte a commit (inserimenti e richiami di messaggi) vengono sottoposte a commit.

Definito in: classe MQQueueManager

Sintassi: Call *MQQueueManager*.**Disconnetti ()**

Classe MQQueue

Questa classe rappresenta l'accesso a una coda WebSphere MQ . Questa connessione viene fornita da un oggetto MQQueueManager . Quando un oggetto di questa classe viene eliminato, viene chiuso automaticamente.

Contenimento

La classe MQQueue è contenuta nella classe MQQueueManager .

Creazione

New crea un nuovo oggetto MQQueue e imposta tutte le proprietà sui valori iniziali. In alternativa, utilizzare il metodo AccessQueue della classe MQQueueManager .

Sintassi

```
Dim que As New MQQueue Set que  = New MQQueue
```

Proprietà

- “proprietà ID AlternateUser” a pagina 1060.
- “proprietà Nome BackoutRequeue” a pagina 1060.
- “proprietà BackoutThreshold” a pagina 1060.
- “proprietà Nome BaseQueue” a pagina 1060.
- “proprietà CloseOptions” a pagina 1060.
- “proprietà CompletionCode” a pagina 1061.
- “proprietà ConnectionReference” a pagina 1061.
- “proprietà CreationDate” a pagina 1061.

- “proprietà CurrentDepth” a pagina 1061.
- “proprietà DefaultInputOpenOption” a pagina 1061.
- “proprietà DefaultPersistence” a pagina 1062.
- “proprietà DefaultPriority” a pagina 1062.
- “proprietà DefinitionType” a pagina 1062.
- “proprietà Evento DepthHigh” a pagina 1062.
- “proprietà Limite DepthHigh” a pagina 1062.
- “proprietà Evento DepthLow” a pagina 1063.
- “proprietà Limite DepthLow” a pagina 1063.
- “proprietà Evento DepthMaximum” a pagina 1063.
- “proprietà Evento DepthHigh” a pagina 1062.
- “proprietà Limite DepthHigh” a pagina 1062.
- “proprietà Evento DepthLow” a pagina 1063.
- “proprietà Limite DepthLow” a pagina 1063.
- “proprietà Evento DepthMaximum” a pagina 1063.
- “proprietà Descrizione” a pagina 1063.
- “proprietà Nome DynamicQueue” a pagina 1063.
- “proprietà Backout HardenGet” a pagina 1064.
- “proprietà InhibitGet” a pagina 1064.
- “proprietà InhibitPut” a pagina 1064.
- “proprietà Nome InitiationQueue” a pagina 1064.
- “proprietà IsOpen” a pagina 1064.
- “proprietà MaximumDepth” a pagina 1065.
- “proprietà Lunghezza MaximumMessage” a pagina 1065.
- “proprietà Sequenza MessageDelivery” a pagina 1065.
- “proprietà ObjectHandle” a pagina 1065.
- “proprietà Conteggio OpenInput” a pagina 1066.
- “proprietà OpenOptions” a pagina 1066.
- “proprietà Conteggio OpenOutput” a pagina 1066.
- “proprietà OpenStatus” a pagina 1066.
- “proprietà ProcessName” a pagina 1066.
- “proprietà Nome QueueManager” a pagina 1066.
- “Proprietà QueueType” a pagina 1067.
- “proprietà ReasonCode” a pagina 1067.
- “proprietà ReasonName” a pagina 1067.
- “Proprietà RemoteQueueManagerName” a pagina 1067.
- “proprietà RemoteQueueNome” a pagina 1067.
- “proprietà ResolvedQueueManagerName” a pagina 1068.
- “proprietà Nome ResolvedQueue” a pagina 1068.
- “proprietà RetentionInterval” a pagina 1068.
- “proprietà Ambito” a pagina 1068.
- “proprietà ServiceInterval” a pagina 1068.
- “Proprietà evento ServiceInterval” a pagina 1068.

- “proprietà Condividibilità” a pagina 1069.
- “proprietà Nome TransmissionQueue” a pagina 1069.
- “proprietà TriggerControl” a pagina 1069.
- “proprietà TriggerData” a pagina 1069.
- “proprietà TriggerDepth” a pagina 1069.
- “proprietà Priorità TriggerMessage” a pagina 1069.
- “proprietà TriggerType” a pagina 1070.
- “proprietà Utilizzo” a pagina 1070.

Metodi

- “Metodo di codici ClearError” a pagina 1070
- “Metodo di chiusura” a pagina 1070
- “Metodo GET” a pagina 1070
- “Apri metodo” a pagina 1071
- “Metodo PUT” a pagina 1071

Accesso proprietà

Se l'oggetto della coda non è connesso a un gestore code, è possibile leggere le seguenti proprietà:

- “proprietà CompletionCode” a pagina 1061
- “proprietà OpenStatus” a pagina 1066
- “proprietà ReasonCode” a pagina 1067

e si può leggere e scrivere a:

- “proprietà ID AlternateUser” a pagina 1060
- “proprietà CloseOptions” a pagina 1060
- “proprietà ConnectionReference” a pagina 1061
- “proprietà Nome” a pagina 1065
- “proprietà OpenOptions” a pagina 1066

Se l'oggetto coda è connesso a un gestore code, è possibile leggere tutte le proprietà.

Proprietà attributo coda

Le propriet ... non elencate nella sezione precedente sono tutti attributi della coda WebSphere MQ sottostante. È possibile accedervi solo se l'oggetto è connesso a un gestore code e l'ID utente dell'utente è autorizzato per l'interrogazione o l'impostazione rispetto a tale coda. Se un ID utente alternativo è impostato e l'ID utente corrente è autorizzato ad utilizzarlo, l'ID utente alternativo viene controllato per l'autorizzazione.

La proprietà deve essere una proprietà appropriata per il QueueTypefornito. Per ulteriori informazioni, consultare Attributi per le code .

Se queste condizioni non si applicano, l'accesso alla proprietà imposterà un CompletionCode di MQCC_FAILED e uno dei seguenti ReasonCodes:

- MQRC_CONNECTION_BROKEN
- MQRC_NOT_AUTHORIZED
- ERRORE MQRC_Q_MGR_NAME_
- MQRC_Q_MGR_NOT_CONNECTED
- MQRC_SELECTOR_NOT_FOR_TYPE (CompletionCode è MQCC_WARNING)

Apertura di una coda

L'unico modo per creare un oggetto MQQueue è utilizzando il metodo MQQueueManager AccessQueue o Nuovo. Un oggetto MQQueue aperto rimane aperto (OpenStatus= TRUE) fino a quando non viene chiuso o eliminato o fino a quando l'oggetto gestore code di creazione non viene eliminato o la connessione al gestore code non viene persa. Il valore della proprietà MQQueue CloseOptions controlla il funzionamento dell'operazione di chiusura che si verifica quando l'oggetto MQQueue viene eliminato.

Il metodo MQQueueManager AccessQueue apre la coda utilizzando il parametro OpenOptions . Il metodo MQQueue.Open apre la coda utilizzando la proprietà OpenOptions . WebSphere MQ convalida OpenOptions rispetto all'autorizzazione utente come parte del processo di apertura della coda.

proprietà ID AlternateUser

Lettura - scrittura. L'ID utente alternativo utilizzato per convalidare l'accesso alla coda quando viene aperto.

Questa proprietà non può essere impostata mentre l'oggetto è aperto (ovvero, quando IsOpen è TRUE).

Definito in: classe MQQueue

Tipo di dati: stringa di 12 caratteri

Sintassi: per ottenere *altuser $= MQQueue*.**AlternateUserId**

Per impostare: *MQQueue.***AlternateUserId** = *altuser $*

proprietà Nome BackoutRequeue

Sola lettura. L'attributo MQI BackOutRequeueQName .

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere *backoutrequeuename $= MQQueue*.**BackoutRequeueNome**

proprietà BackoutThreshold

Sola lettura. L'attributo BackoutThreshold MQI.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- Consultare BackoutThreshold (MQLONG).

Sintassi: per richiamare *backoutthreshold & = MQQueue.***BackoutThreshold**

proprietà Nome BaseQueue

Sola lettura. Il nome della coda in cui viene risolto l'alias.

Valido solo per le code alias.

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere *baseqname $= MQQueue*.**BaseQueueName**

proprietà CloseOptions

Lettura - scrittura. Opzioni utilizzate per controllare cosa accade quando la coda viene chiusa.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQCO_NONE
- MQCO_DELETE
- MQCO_DELETE_PURGE

MQCO_DELETE e MQCO_DELETE_PURGE sono validi solo per code dinamiche.

Sintassi: per ottenere *closeopt & = MQQueue***.CloseOptions**

Per impostare: *MQQueue***.CloseOptions** = *closeopt &*

proprietà CompletionCode

Sola lettura. Restituisce il codice di completamento impostato dall'ultimo metodo o accesso alla proprietà emesso rispetto all'oggetto.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQCC_OK
- MQCC_AVVERTENZA
- MQCC_NON RIUSCITO

Sintassi: per ottenere: *completioncode & = MQQueue***.CompletionCode**

proprietà ConnectionReference

Lettura - scrittura. Definisce l'oggetto gestore code a cui appartiene un oggetto coda. Non è possibile scrivere il riferimento di connessione mentre una coda è aperta.

Definito in: classe MQQueue

Tipo di dati: MQQueueManager

Valori:

- Un riferimento a un oggetto gestore code WebSphere MQ attivo

Sintassi: per impostare: *set MQQueue***.ConnectionReference** = *ConnectionReference*

Per ottenere: *set ConnectionReference = MQQueue***.ConnectionReference**

proprietà CreationDate

Sola lettura. Data e ora in cui è stata creata questa coda.

Definito in: classe MQQueue

Tipo di dati: variante di tipo 7 (data/ora) o EMPTY

Sintassi: per ottenere *datetime = MQQueue***.CreationDateTime**

proprietà CurrentDepth

Sola lettura. Il numero di messaggi nella coda.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere *currentdepth & = MQQueue***.CurrentDepth**

proprietà DefaultInputOpenOption

Sola lettura. Controlla il modo in cui la coda viene aperta se OpenOptions specifica MQOO_INPUT_AS_Q_DEF.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_SHARED

Sintassi: per ottenere *defaultinop & = MQQueue*.**DefaultInputOpenOption**

proprietà DefaultPersistence

Sola lettura. La persistenza predefinita per i messaggi su una coda.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere *defpersistence & = MQQueue*.**DefaultPersistence**

proprietà DefaultPriority

Sola lettura. La priorità predefinita per i messaggi su una coda.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere *defpriority & = MQQueue*.**DefaultPriority**

proprietà DefinitionType

Sola lettura. Il tipo di definizione della coda.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQQDT_PREDEFINED
- MQQDT_PERMANENT_DYNAMIC
- MQQDT_TEMPORARY_DYNAMIC

Sintassi: per ottenere: *deftype & = MQQueue*.**DefinitionType**

proprietà Evento DepthHigh

Sola lettura. L'attributo evento QDepthHighMQI.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- DISABILITAZIONE_MQEVR
- MQEVR_ENABLED

Sintassi: Per ottenere: *depthhighevent & = MQQueue.***DepthHighEvent**

proprietà Limite DepthHigh

Sola lettura. Attributo Limite MQI QDepthHigh.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere *depthhighlimit & = MQQueue.***DepthHighLimit**

proprietà Evento DepthLow

Sola lettura. L'attributo evento QDepthLowMQI.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- DISABILITAZIONE_MQEVR
- MQEVR_ENABLED

Sintassi: Per ottenere *depthlowevent & = MQQueue.***DepthLowEvent**

proprietà Limite DepthLow

Sola lettura. L'attributo QDepthLowLimite MQI.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere: *depthlowlimit & = MQQueue.***DepthLowLimite**

proprietà Evento DepthMaximum

Sola lettura. L'attributo evento QDepthMaxMQI.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- DISABILITAZIONE_MQEVR
- MQEVR_ENABLED

Sintassi: per ottenere *depthmaximumevent & = MQQueue.***DepthMaximumEvent**

proprietà Descrizione

Sola lettura. Una descrizione della coda.

Definito in: classe MQQueue

Tipo di dati: stringa di 64 caratteri

Sintassi: per ottenere *description $= MQQueue***.Descrizione**

proprietà Nome DynamicQueue

Lettura - scrittura, sola lettura quando la coda è aperta.

Controlla il nome della coda dinamica utilizzato quando viene aperta una coda modello. Può essere impostato con un carattere jolly dall'utente come un insieme di proprietà (solo quando la coda è chiusa) o come un parametro per MQQueueManager.AccessQueue().

Il nome effettivo della coda dinamica viene trovato interrogando QueueName.

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Valori:

- Qualsiasi nome coda WebSphere MQ valido.

Sintassi: Per impostare: *MQQueue***.DynamicQueueName** = *dynamicqueuename $*

Per richiamare: *dynamicqueuename $* = *MQQueue***.DynamicQueueName**

proprietà Backout HardenGet

Sola lettura. Indica se mantenere un conteggio di backout accurato.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQQA_BACKOUT_HARDENED
- MQQA_BACKOUT_NOT HARDENED

Sintassi: per ottenere: *hardengetback & = MQQueue***.HardenGetBackout**

proprietà InhibitGet

Lettura - scrittura. L'attributo InhibitGet MQI.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQQA_GET_INIBITO
- MQQA_GET_ALLOWED

Sintassi: per ottenere *getstatus & = MQQueue***.InhibitGet**

Per impostare: *MQQueue***.InhibitGet** = *getstatus &*

proprietà InhibitPut

Lettura - scrittura. L'attributo InhibitPut MQI.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQQA_PUT_INIBITO
- MQQA_PUT_CONSENTITO

Sintassi: per ottenere *putstatus & = MQQueue***.InhibitPut**

Per impostare: *MQQueue***.InhibitPut** = *putstatus &*

proprietà Nome InitiationQueue

Sola lettura. Nome della coda di iniziazione.

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere *initqname $= MQQueue***.InitiationQueueName**

proprietà IsOpen

Restituisce se la coda è aperta.

Sola lettura.

Definito in: classe MQQueue

Tipo di dati : booleano

Valori:

- VERO (-1)
- FALSE (0)

Sintassi: Per ottenere *open = MQQueue***.IsOpen**

proprietà MaximumDepth

Sola lettura. Profondità massima della coda.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere: *maxdepth & = MQQueue***.MaximumDepth**

proprietà Lunghezza MaximumMessage

Sola lettura. Lunghezza massima consentita del messaggio in byte per questa coda.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere *maxmlength & = MQQueue***.MaximumMessageLength**

proprietà Sequenza MessageDelivery

Sola lettura. La sequenza di distribuzione dei messaggi.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- PRIORITà_MQMDS
- FIFO MQMDS

Sintassi: Per ottenere *messdelseq & = MQQueue***.MessageDeliverySequence**

proprietà Nome

Lettura - scrittura. L'attributo MQI Queue. Questa proprietà non può essere scritta dopo l'apertura di MQQueue.

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Sintassi: per richiamare: *name $= MQQueue***.name**

Per impostare: *MQQueue***.name** = *name $*

Nota: Visual Basic riserva la proprietà "Nome" per l'utilizzo nell'interfaccia visiva. Pertanto, quando si utilizza Visual Basic, utilizzare il carattere minuscolo, ossia "name".

proprietà ObjectHandle

Sola lettura. L'handle di oggetto per l'oggetto coda WebSphere MQ .

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere *hobj & = MQQueue.***ObjectHandle**

proprietà Conteggio OpenInput

Sola lettura. Numero di aperture per l'input.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere: *openincount & = MQQueue*.**OpenInputCount**

proprietà OpenOptions

Lettura - scrittura. Opzioni da utilizzare per aprire la coda.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- Vedere OpenOptions (MQLONG).

Sintassi: Per ottenere: *openopt & = MQQueue*.**OpenOptions**

Per impostare: *MQQueue.***OpenOptions** = *openopt &*

proprietà Conteggio OpenOutput

Sola lettura. Numero di aperture per l'output.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere *openoutcount & = MQQueue*.**OpenOutputCount**

proprietà OpenStatus

Sola lettura. Indica se la coda è aperta o meno. Il valore iniziale è TRUE dopo il metodo AccessQueue o FALSE dopo New.

Definito in: classe MQQueue

Tipo di dati : booleano

Valori:

- VERO (-1)
- FALSE (0)

Sintassi Per ottenere: *status & = MQQueue*.**OpenStatus**

proprietà ProcessName

Sola lettura. L'attributo ProcessName MQI.

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere: *procname $ =MQQueue*.**ProcessName**

proprietà Nome QueueManager

Lettura - scrittura. Il nome del gestore code WebSphere MQ .

Definito in: classe MQQueue

Tipo di dati: stringa

Sintassi: per richiamare: *QueueManagerName$ = MQQueue*.**QueueManagerName**

Per impostare: *MQQueue*.**QueueManagerNome** = *QueueManagerName$*

Proprietà QueueType

Sola lettura. L'attributo QType MQI.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- ALIAS MQQT
- LOCALE MQQT
- MODEL MQQT
- REMOTE MQQT

Sintassi: per ottenere: *queuetype & = MQQueue*.**QueueType**

proprietà ReasonCode

Sola lettura. Restituisce il codice motivo impostato dall'ultimo metodo o accesso alla proprietà emesso per l'oggetto.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- Vedere codici di errore API.

Sintassi: Per ottenere *reasoncode & = MQQueue*.**ReasonCode**

proprietà ReasonName

Sola lettura. Restituisce il nome simbolico dell'ultimo codice motivo. Ad esempio, "MQRC_QMGR_NOT_AVAILABLE".

Definito in: classe MQQueue

Tipo di dati: stringa

Valori:

- Vedere codici di errore API.

Sintassi: per richiamare *reasonname $= MQQueue*.**ReasonName**

Proprietà RemoteQueueManagerName

Sola lettura. Il nome del gestore code remoto. Valido solo per le code remote.

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere: *remqmanname $= MQQueue*.**RemoteQueueManagerName**

proprietà RemoteQueueNome

Sola lettura. Il nome della coda come è noto sul gestore code remoto. Valido solo per le code remote.

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Sintassi: per richiamare *remqname $= MQQueue*.**RemoteQueueName**

proprietà ResolvedQueueManagerName

Sola lettura. Il nome del gestore code di destinazione finale come riconosciuto dal gestore code locale.

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere *resqmanname $= MQQueue*.**ResolvedQueueManagerName**

proprietà Nome ResolvedQueue

Sola lettura. Il nome della coda di destinazione finale noto al gestore code locale.

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere: *resqname $= MQQueue*.**ResolvedQueueName**

proprietà RetentionInterval

Sola lettura. Il periodo di tempo per cui la coda deve essere mantenuta.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per richiamare *retinterval & = MQQueue*.**RetentionInterval**

proprietà Ambito

Sola lettura. Controlla se una voce per questa coda esiste anche in una directory della cella.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MGR coda MQSCO
- CCELL MQSCO

Sintassi: per ottenere: *scope & = MQQueue*.**Scope**

proprietà ServiceInterval

Sola lettura. L'attributo QServiceInterval MQI.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere *serviceinterval & = MQQueue.***ServiceInterval**

Proprietà evento ServiceInterval

Sola lettura. L'attributo evento QServiceIntervalMQI.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQQSIE_HIGH
- MQQSIE_OK
- MQQSIE_NONE

Sintassi: Per ottenere *serviceintervalevent & = MQQueue.***ServiceIntervalEvent**

proprietà Condividibilità

Sola lettura. Condividibilità coda.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQQA_XX_ENCODE_CASE_ONE abilitazione
- MQQA_NOT_SHAREABLE

Sintassi: per ottenere *shareability & = MQQueue***.Shareability**

proprietà Nome TransmissionQueue

Sola lettura. Il nome della coda di trasmissione. Valido solo per le code remote.

Definito in: classe MQQueue

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere *transqname $= MQQueue***.TransmissionQueueName**

proprietà TriggerControl

Lettura - scrittura. Controllo trigger.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQT_DISATTIVO
- MQT_ATTIVO

Sintassi: per ottenere *trigcontrol & = MQQueue***.TriggerControl**

Per impostare: *MQQueue***.TriggerControl** = *trigcontrol &*

proprietà TriggerData

Lettura - scrittura. I dati del trigger.

Definito in: classe MQQueue

Tipo di dati: stringa di 64 caratteri

Sintassi: per ottenere: *trigdata $= MQQueue***.TriggerData**

Per impostare: *MQQueue***.TriggerData** = *trigdata $*

proprietà TriggerDepth

Lettura - scrittura. Il numero di messaggi che devono essere sulla coda prima che venga scritto un messaggio trigger.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per ottenere: *trigdepth & = MQQueue***.TriggerDepth**

Per impostare: *MQQueue***.TriggerDepth** = *trigdepth &*

proprietà Priorità TriggerMessage

Lettura - scrittura. La priorità dei messaggi per i trigger.

Definito in: classe MQQueue

Tipo di dati: Long

Sintassi: per richiamare: *trigmesspriority & = MQQueue*.**TriggerMessagePriority**

Per impostare: *MQQueue*.**TriggerMessagePriority** = *trigmesspriority &*

proprietà TriggerType

Lettura - scrittura. Il tipo di trigger.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQTT_NONE
- MQTT_FIRST
- MQTT_EVERY
- DEPTH MQT

Sintassi: per ottenere: *trigtype & = MQQueue*.**TriggerType**

Per impostare: *MQQueue*.**TriggerType** = *Trigtype &*

proprietà Utilizzo

Sola lettura. Indica per cosa viene utilizzata la coda.

Definito in: classe MQQueue

Tipo di dati: Long

Valori:

- MQUS_NORMALE
- MQUS_TRASMISSIONE

Sintassi: per ottenere *usage & = MQQueue*.**Utilizzo**

Metodo di codici ClearError

Reimposta CompletionCode su MQCC_OK e ReasonCode su MQRC_NONE per le classi MQQueue e MQSession.

Definito in: classe MQQueue

Sintassi: Call *MQQueue*.**ClearErrorCodes ()**

Metodo di chiusura

Chiude una coda utilizzando i valori correnti di CloseOptions.

Definito in: classe MQQueue

Sintassi: richiamare *MQQueue*.**Chiudi ()**

Metodo GET

Richiama un messaggio dalla coda.

Questo metodo utilizza un oggetto MQMessage come parametro, utilizzando alcuni dei campi in MQMD dell'oggetto come parametri di input. In particolare, vengono utilizzati i campi MessageId e CorrelId , quindi è importante assicurarsi che questi campi siano impostati come richiesto. Per ulteriori informazioni relative a questi campi, consultare MsgId (MQBYTE24) e CorrelId (MQBYTE24) .

Se il metodo ha esito negativo, l'oggetto MQMessage non viene modificato. Se ha esito positivo, le parti MQMD e Message Data dell'oggetto MQMessage vengono sostituite da MQMD e Message Data del messaggio in entrata. Le proprietà di controllo MQMessage sono impostate come segue

- **MessageLength** è impostato sulla lunghezza del messaggio WebSphere MQ
- **DataLength** è impostato sulla lunghezza del messaggio WebSphere MQ
- **DataOffset** è impostato a zero

Definito in:
Classe MQQueue

Sintassi: richiamare *MQQueue*.**Richiama**(*Message, GetMsgOptions, GetMsgLength*)

Parametri
Messaggio

Oggetto MQMessage che rappresenta il messaggio da richiamare.

Opzioni GetMsg

Oggetto facoltativo MQGetMessageOptions per controllare l'operazione get. Se questo parametro non è specificato, vengono utilizzate le opzioni MQGetMessagepredefinite.

GetMsgLunghezza:

Il valore facoltativo di lunghezza di 2 o 4 byte per controllare la lunghezza massima del messaggio WebSphere MQ richiamato dalla coda.

Se viene specificata l'opzione MQGMO_ACCEPT_TRUNCATED_MSG, GET ha esito positivo con un codice di completamento MQCC_WARNING e un codice motivo MQRC_TRUNCATED_MSG_ACCEPTED se la dimensione del messaggio supera la lunghezza specificata.

MessageData contiene i primi GetMsgbyte di lunghezza dei dati.

Se MQGMO_ACCEPT_TRUNCATED_MSG ***non viene specificato*** e la dimensione del messaggio supera la lunghezza specificata, viene restituito il codice di completamento MQCC_FAILED insieme al codice motivo MQRC_TRUNCATED_MESSAGE_FAILED.

Se il contenuto del buffer di messaggi non è definito, la lunghezza totale del messaggio viene impostata sulla lunghezza completa del messaggio che sarebbe stato richiamato.

Se il parametro della lunghezza del messaggio non è specificato, la lunghezza del buffer del messaggio viene automaticamente regolata almeno alla dimensione del messaggio in arrivo.

Apri metodo

Apre una coda utilizzando i valori correnti di:

1. QueueName
2. QueueManagerName
3. AlternateUserid
4. Nome DynamicQueue

Definito in:
Classe MQQueue

Sintassi: richiamare *MQQueue*.**Apri ()**

Metodo PUT

Inserisce un messaggio nella coda.

Questo metodo utilizza un oggetto MQMessage come parametro. Le proprietà MQMD (Message Descriptor) di questo oggetto potrebbero essere modificate come risultato di questo metodo. I valori che hanno immediatamente dopo l'esecuzione di questo metodo sono i valori che sono stati inseriti in WebSphere MQ.

Le modifiche all'oggetto MQMessage dopo il completamento dell'inserimento non influiscono sul messaggio effettivo nella coda WebSphere MQ .

Definito in:
Classe MQQueue

Sintassi: richiamare *MQQueue*.**Put**(*Messaggio, PutMsgOpzioni)*

Parametri
Messaggio

Oggetto MQMessage che rappresenta il messaggio da inserire.

Opzioni PutMsg

MQPutMessageOggetto opzioni contenente opzioni per controllare l'operazione di inserimento. Se non vengono specificate, vengono utilizzate le opzioni PutMessagepredefinite.

Classe MQMessage

Questa classe rappresenta un messaggio WebSphere MQ . Include le proprietà per incapsulare il descrittore di messaggi WebSphere MQ (MQMD) e fornisce un buffer per contenere i dati del messaggio definito dall'applicazione.

La classe include i metodi di scrittura per copiare i dati da un'applicazione ActiveX in un oggetto MQMessage. Allo stesso modo, la classe include i metodi di lettura per copiare i dati da un oggetto MQMessage a un'applicazione ActiveX . La classe gestisce l'assegnazione e la deallocazione della memoria per il buffer automaticamente. L'applicazione non deve dichiarare la dimensione del buffer quando viene creato l'oggetto MQMessage perché il buffer cresce per contenere i dati scritti.

Non è possibile inserire un messaggio in una coda WebSphere MQ se la dimensione del buffer supera la proprietà MaximumMessageLength di tale coda.

Dopo che è stato creato, un oggetto MQMessage può essere inserito in una coda WebSphere MQ utilizzando il metodo MQQueue.Put . Questo metodo prende una copia delle parti MQMD e dei dati del messaggio dell'oggetto e inserisce la copia nella coda. L'applicazione può quindi modificare o eliminare un oggetto MQMessage dopo l'inserimento, senza influire sul messaggio nella coda WebSphere MQ . Il gestore code può modificare alcuni campi in MQMD quando copia il messaggio nella coda WebSphere MQ .

È possibile leggere un messaggio in entrata in un oggetto MQMessage utilizzando il metodo MQQueue.Get . Ciò sostituisce qualsiasi MQMD o dato del messaggio che potrebbe essere già stato nell'oggetto MQMessage con i valori del messaggio in entrata. Regola la dimensione del buffer di dati dell'oggetto MQMessage in modo che corrisponda alla dimensione dei dati del messaggio in entrata.

Contenimento

I messaggi sono contenuti nella classe MQSession.

Creazione

Nuovo crea un oggetto MQMessage. Le proprietà del descrittore messaggi sono inizialmente impostate sui valori predefiniti e il relativo buffer di dati dei messaggi è vuoto.

Sintassi

```
Dim msg As New MQMessage or Set msg = New MQMessage
```

Proprietà

Le proprietà di controllo sono:

Le proprietà del descrittore messaggi sono:

Metodi

- “Metodo di lettura” a pagina 1085
- “metodo ReadBoolean” a pagina 1085
- “metodo ReadByte” a pagina 1085
- “metodo ReadDecimal2” a pagina 1086
- “metodo ReadDecimal4” a pagina 1086
- “metodo ReadDouble” a pagina 1086
- “metodo ReadDouble4” a pagina 1086
- “metodo ReadFloat” a pagina 1086
- “metodo ReadInt2” a pagina 1087
- “metodo ReadInt4” a pagina 1087
- “metodo ReadLong” a pagina 1087
- “metodo ReadNullTerminatedString” a pagina 1087
- “metodo ReadShort” a pagina 1087
- “metodo ReadString” a pagina 1088
- “metodo ReadUInt2” a pagina 1088
- “metodo ReadUnsignedByte” a pagina 1088
- “metodo ReadUTF” a pagina 1088
- “Metodo ResizeBuffer” a pagina 1089
- “Metodo di scrittura” a pagina 1089
- “metodo WriteBoolean” a pagina 1089
- “metodo WriteByte” a pagina 1089
- “metodo WriteDecimal2” a pagina 1090
- “metodo WriteDecimal4” a pagina 1090
- “metodo WriteDouble” a pagina 1090
- “metodo WriteDouble4” a pagina 1090
- “metodo WriteFloat” a pagina 1091
- “metodo WriteInt2” a pagina 1091
- “metodo WriteInt4” a pagina 1091
- “metodo WriteLong” a pagina 1091
- “metodo WriteNullTerminatedString” a pagina 1091
- “Metodo WriteShort” a pagina 1092
- “metodo WriteString” a pagina 1092
- “metodo WriteUInt2” a pagina 1092
- “Metodo byte WriteUnsigned” a pagina 1092
- “Metodo WriteUTF” a pagina 1093

Accesso proprietà

Tutte le proprietà possono essere lette in qualsiasi momento.

Le proprietà di controllo sono di sola lettura, ad eccezione di DataOffset che è lettura - scrittura.
Le proprietà Descrittore messaggio sono tutte di lettura - scrittura, tranne BackoutCount e TotalMessageLength che sono entrambe di sola lettura.

Notare, tuttavia, che alcune delle proprietà MQMD potrebbero essere modificate dal gestore code quando il messaggio viene inserito in una coda WebSphere MQ . Consultare i campi in MQMD per i dettagli su come potrebbero essere modificati.

Conversione dati

È possibile passare dati binari a un messaggio WebSphere MQ impostando la proprietà CharacterSet su Coded Character Set Identifier del gestore code (MQCCSI_Q_MGR) e inviando una stringa. È possibile utilizzare la funzione chr $per impostare dati non carattere nella stringa.

I metodi di lettura e scrittura eseguono la conversione dei dati. Vengono convertiti tra i formati interni ActiveX e i formati del messaggio WebSphere MQ come definito dalle proprietà Codifica e CharacterSet dal descrittore del messaggio. Quando si scrive un messaggio, impostare i valori in Codifica e CharacterSet che corrispondono alle caratteristiche del destinatario del messaggio prima di emettere un metodo di scrittura. Durante la lettura di un messaggio, questo passo non è normalmente richiesto perché questi valori saranno stati impostati da quelli in MQMD in entrata.

Questo è un passo di conversione dati aggiuntivo che si verifica dopo qualsiasi conversione eseguita dal metodo MQQueue.Get .

proprietà CompletionCode

Sola lettura. Restituisce il codice di completamento WebSphere MQ impostato dal metodo più recente o dall'accesso alla proprietà emesso per questo oggetto.

Definito in: classe MQMessage

Tipo di dati: Long

Valori:

- MQCC_OK
- MQCC_AVVERTENZA
- MQCC_NON RIUSCITO

Sintassi: Per ottenere *completioncode & = MQMessage***.CompletionCode**

proprietà DataLength

Sola lettura. Questa proprietà restituisce il valore:

```
MQMessage.MessageLength - MQMessage.DataOffset
```

Può essere utilizzato prima di un metodo di lettura, per controllare che il numero previsto di caratteri sia effettivamente presente nel buffer.

Il valore iniziale è zero.

Definito in: classe MQMessage

Tipo di dati: Long

Sintassi: Per ottenere *bytesleft & = MQMessage***.DataLength**

proprietà DataOffset

Lettura - scrittura. La posizione corrente all'interno della parte Dati messaggio dell'oggetto messaggio.

Il valore è espresso come offset di byte dall'inizio del buffer di dati del messaggio; il primo carattere nel buffer corrisponde a un valore DataOffset pari a zero.

Un metodo di lettura o scrittura inizia la sua operazione sul carattere a cui fa riferimento DataOffset. Questi metodi elaborano i dati nel buffer in modo sequenziale da questa posizione e aggiornano DataOffset in modo che puntino al byte (se presente) immediatamente dopo l'ultimo byte elaborato.

DataOffset può assumere solo valori compresi nell'intervallo da zero a MessageLength incluso. Quando DataOffset = MessageLength punta alla parte finale, che è il primo carattere non valido del buffer. I metodi di scrittura sono consentiti in questa situazione - estendono i dati nel buffer e aumentano MessageLength del numero di byte aggiunti. La lettura oltre la fine del buffer non è valida.

Il valore iniziale è zero.

Definito in: classe MQMessage

Tipo di dati: Long

Sintassi: per ottenere *currpos & = MQMessage*.**DataOffset**

Per impostare: *MQMessage*.**DataOffset** = *currpos &*

proprietà MessageLength

Sola lettura. Restituisce la lunghezza totale della parte Dati messaggio dell'oggetto messaggio in caratteri, indipendentemente dal valore di DataOffset.

Il valore iniziale è zero. È impostato sulla lunghezza del messaggio in entrata dopo un richiamo del metodo Get che fa riferimento a questo oggetto messaggio. Viene incrementato se l'applicazione utilizza un metodo di scrittura per aggiungere dati all'oggetto. Non è influenzato dai metodi Read.

Definito in: classe MQMessage

Tipo di dati: Long

Sintassi: per ottenere *msglength & = MQMessage*.**MessageLength**

proprietà ReasonCode

Sola lettura. Restituisce il codice di errore impostato dal metodo più recente o l'accesso alla proprietà emesso per questo oggetto.

Definito in: classe MQMessage

Tipo di dati: Long

Valori:

- Vedere codici di errore API.

Sintassi: per ottenere *reasoncode & = MQMessage*.**ReasonCode**

proprietà ReasonName

Sola lettura. Restituisce il nome simbolico dell'ultimo codice motivo. Ad esempio, "MQRC_QMGR_NOT_AVAILABLE". **Definito in:** classe MQMessage

Tipo di dati: stringa

Valori:

- Vedere codici di errore API.

Sintassi: per ottenere: *reasonname $= MQMessage*.**ReasonName**

proprietà AccountingToken

Lettura - scrittura. MQMD AccountingToken - parte del contesto di identità del messaggio.

Il valore iniziale è tutti null.

Definito in: classe MQMessage

Tipo di dati: stringa di 32 caratteri

Sintassi: per ottenere: *actoken $= MQMessage*.**AccountingToken**

Per impostare: *MQMessage.***AccountingToken** = *actoken $*

Per ulteriori informazioni su quando è necessario utilizzare l'esadecimale AccountingTokenal posto della proprietà AccountingToken , consultare “Proprietà descrittori messaggi” a pagina 1037 .

proprietà Hex AccountingToken

Lettura - scrittura. MQMD AccountingToken - parte del contesto di identità del messaggio.

Ogni due caratteri rappresentano l'equivalente esadecimale di un singolo carattere ASCII. Ad esempio, la coppia di caratteri "6" e "1" rappresentano il singolo carattere "A", la coppia di caratteri "6" e "2" rappresentano il singolo carattere "B" e così via.

È necessario fornire 64 caratteri esadecimali validi.

Il suo valore iniziale è "0 ... 0"

Definito in: classe MQMessage

Tipo di dati: stringa di 64 caratteri esadecimali che rappresenta 32 caratteri ASCII

Sintassi: per ottenere: *actokenh $= MQMessage*.**AccountingTokenHex**

Per impostare: *MQMessage.***AccountingTokenHex** = *actokenh $*

Per ulteriori informazioni su quando è necessario utilizzare l'esadecimale AccountingTokenal posto della proprietà AccountingToken , consultare “Proprietà descrittori messaggi” a pagina 1037 .

proprietà Dati ApplicationId

Lettura - scrittura. I dati MQMD ApplIdentity- parte del contesto di identità del messaggio.

Il valore iniziale è costituito da tutti spazi.

Definito in: classe MQMessage

Tipo di dati: stringa di 32 caratteri

Sintassi: per richiamare: *applid $= MQMessage*.**ApplicationIdData**

Per impostare: *MQMessage*.**ApplicationIdData** = *applid $*

proprietà Dati ApplicationOrigin

Lettura - scrittura. MQMD ApplOriginData - parte del contesto di origine del messaggio.

Il valore iniziale è costituito da tutti spazi.

Definito in: classe MQMessage

Tipo di dati: stringa di 4 caratteri

Sintassi: per ottenere: *applor $= MQMessage*.**ApplicationOriginData**

Per impostare: *MQMessage*.**ApplicationOriginData** = *applor $*

proprietà BackoutCount

Sola lettura. MQMD BackoutCount.

Il suo valore iniziale è 0

Definito in: classe MQMessage

Tipo di dati: Long

Sintassi: per ottenere *backoutct & = MQMessage*.**BackoutCount**

proprietà CharacterSet

Lettura - scrittura. MQMD CodedCharSetId.

Il valore iniziale è il valore speciale **MQCCSI_Q_MGR** .

Se CharacterSet è impostato su **MQCCSI_Q_MGR**, la codepage per la locale corrente viene utilizzata per la conversione dei caratteri nel metodo WriteString . Per le applicazioni server, la codepage utilizzata è la codepage del gestore code. Per le applicazioni client, è la codepage locale corrente predefinita.

Ad esempio:

```
msg.CharacterSet = MQCCSI_Q_MGR
msg.WriteString(chr$(n))
```

dove ' n' è maggiore o uguale a zero e minore o uguale a 255, risulta in un singolo byte del valore di 'n' scritto nel buffer.

Definito in: classe MQMessage

Tipo di dati: Long

Sintassi: Per ottenere: *:30ccid& = MQMessage* **.CharacterSet**

Per impostare: *MQMessage***.CharacterSet**= *ccid &*

Esempio

Se si desidera che la stringa venga scritta nella codepage 437, immettere:

```
Message.CharacterSet = 437
Message.WriteString ("string to be written")
```

Impostare il valore desiderato in CharacterSet prima di emettere qualsiasi chiamata WriteString .

proprietà CorrelationId

Lettura - scrittura. Il CorrelationId da includere nell'MQMD di un messaggio quando viene inserito in una coda. Anche l'ID a cui far corrispondere quando si ottiene un messaggio da una coda.

Il valore iniziale è null.

Definito in: classe MQMessage

Tipo di dati: stringa di 24 caratteri

Sintassi: per ottenere *correlid $= MQMessage***.CorrelationId** Per impostare: *MQMessage***.CorrelationId** = *correlid $*

Consultare “Proprietà descrittori messaggi” a pagina 1037 per ulteriori informazioni su quando è necessario utilizzare l'esadecimale CorrelationIdal posto della proprietà CorrelationId .

proprietà Hex CorrelationId

Lettura - scrittura. Il CorrelationId da includere nell'MQMD di un messaggio quando viene inserito in una coda. Anche il CorrelationId rispetto al quale eseguire la corrispondenza quando si ottiene un messaggio da una coda.

Ogni due caratteri della stringa rappresentano l'equivalente esadecimali di un singolo carattere ASCII. Ad esempio, la coppia di caratteri "6" e "1" rappresentano il singolo carattere "A", la coppia di caratteri "6" e "2" rappresentano il singolo carattere "B" e così via.

È necessario fornire 48 caratteri esadecimali validi.

Il valore iniziale è "0 ... 0".

Definito in: classe MQMessage

Tipo di dati: stringa di 48 caratteri esadecimali che rappresentano 24 caratteri ASCII

Sintassi: per ottenere *correlidh $= MQMessage***.CorrelationIdHex**

Per impostare: *MQMessage***.CorrelationIdHex** = *correlidh $*

Consultare “Proprietà descrittori messaggi” a pagina 1037 per una discussione su quando è necessario utilizzare l'esadecimale CorrelationIdal posto della proprietà CorrelationId .

proprietà Codifica

Lettura - scrittura. Il campo MQMD che identifica la rappresentazione utilizzata per i valori numerici nei dati del messaggio dell'applicazione.

Il valore iniziale è il valore speciale MQENC_NATIVE, che varia in base alla piattaforma.

Questa proprietà viene utilizzata dai metodi seguenti:

- metodo ReadDecimal2
- metodo ReadDecimal4
- metodo ReadDouble
- metodo ReadDouble4
- metodo ReadFloat
- metodo ReadInt2
- metodo ReadInt4
- metodo ReadLong
- metodo ReadShort
- metodo ReadUInt2
- metodo WriteDecimal2
- metodo WriteDecimal4
- metodo WriteDouble
- metodo WriteDouble4
- metodo WriteFloat
- metodo WriteInt2
- metodo WriteInt4
- metodo WriteLong
- Metodo WriteShort
- metodo WriteUInt2

Definito in: classe MQMessage

Tipo di dati: Long

Sintassi: per ottenere *encoding & = MQMessage***.Codifica** Per impostare: *MQMessage***.Codifica** = *codifica &*

Se ci si sta preparando a scrivere i dati nel buffer di messaggi, è necessario impostare questo campo in modo che corrisponda alle caratteristiche della piattaforma del gestore code di ricezione se il gestore code di ricezione non è in grado di eseguire la propria conversione dati.

proprietà Scadenza

Lettura - scrittura. Il campo dell'ora di scadenza MQMD, previsto in decimi di secondo.

Il valore iniziale è il valore speciale MQEI_UNLIMITED

Definito in: classe MQMessage

Tipo di dati: Long

Sintassi : per ottenere *scadenza & = MQMessage***.Scadenza**

Per impostare: *MQMessage***.Scadenza** = *scadenza &*

proprietà Feedback

Lettura - scrittura. Il campo di feedback MQMD.

Il valore iniziale è il valore speciale MQFB_NONE.

Definito in: classe MQMessage

Tipo di dati: Long

Valori:

- Vedere Feedback.

Sintassi: Per ottenere: *feedback & = MQMessage*.**Feedback**

Per impostare: *MQMessage*.**Feedback** = *feedback &*

proprietà Formato

Lettura - scrittura. Il campo formato MQMD. Fornisce il nome di un formato integrato o definito dall'utente che descrive la natura dei dati del messaggio.

Il valore iniziale è il valore speciale MQFMT_NONE.

Definito in: classe MQMessage

Tipo di dati: stringa di 8 caratteri

Sintassi: per ottenere *format $= MQMessage*.**Formatta**

Per impostare: *MQMessage*.**Formato** = *formato $*

proprietà GroupId

Lettura - scrittura. L' GroupId da includere in MQPMR di un messaggio quando viene inserito su una coda. Anche l'ID a cui far corrispondere quando si ottiene un messaggio da una coda. Il valore iniziale è tutti null.

Definito in:
Classe MQMessage

Tipo di dati:
Stringa di 24 caratteri

Sintassi: per ottenere: *groupid $= MQMessage*.**GroupId**

Per impostare: *MQMessage*.**GroupId** = *groupid $*

Consultare “Proprietà descrittori messaggi” a pagina 1037 per ulteriori informazioni su quando è necessario utilizzare l'esadecimale GroupIdal posto della proprietà GroupId .

proprietà Hex GroupId

Lettura - scrittura. L' GroupId da includere in MQPMR di un messaggio quando viene inserito su una coda. Anche l'ID a cui far corrispondere quando si ottiene un messaggio da una coda.

Ogni due caratteri della stringa rappresentano l'equivalente esadecimali di un singolo carattere ASCII. Ad esempio, la coppia di caratteri "6" e "1" rappresentano il singolo carattere "A", la coppia di caratteri "6" e "2" rappresentano il singolo carattere "B" e così via.

È necessario fornire 48 caratteri esadecimali validi.

Il valore iniziale è "0 ... 0".

Definito in:
Classe MQMessage

Tipo di dati:
Stringa di 48 caratteri esadecimali che rappresentano 24 caratteri ASCII.

Sintassi: per ottenere *groupidh $= MQMessage*.**GroupIdHex**

Per impostare: *MQMessage*.**GroupIdHex** = *groupidh $*

Consultare “Proprietà descrittori messaggi” a pagina 1037 per ulteriori informazioni su quando è necessario utilizzare l'esadecimale GroupIdal posto della proprietà GroupId .

proprietà MessageData

Lettura - scrittura. Richiama o imposta l'intero contenuto di un messaggio come stringa di caratteri.

Definito in: classe MQMessage

Tipo di dati: variante

Nota: Il tipo di dati utilizzato da questa proprietà è Variant ma MQAX prevede che si tratta di un tipo di stringa variante. Se si passa una variante diversa da questo tipo, verrà restituito l'errore MQRC_OBJECT_TYPE_ERROR.

Sintassi: per ottenere *String$ = MQMessage*.**MessageData**

Per impostare: *MQMessage*.**MessageData** = *String$*

proprietà MessageFlags

Lettura - scrittura. Indicatori di messaggio che specificano le informazioni di controllo della segmentazione. Il valore iniziale è 0.

Definito in:
Classe MQMessage

Tipo di dati:
Lungo

Valori:
Consultare MsgFlags (MQLONG).

Sintassi: per richiamare *messageflags & = MQMessage*.**MessageFlags**

Per impostare: *MQMessage*.**MessageFlags** = *messageflags &*

proprietà MessageId

Lettura - scrittura. Il MessageId da includere nell'MQMD di un messaggio quando viene inserito su una coda. Anche l'ID a cui far corrispondere quando si ottiene un messaggio da una coda.

Il valore iniziale è tutti null.

Definito in: classe MQMessage

Tipo di dati: stringa di 24 caratteri

Sintassi: Per ottenere: *messageid $= MQMessage*.**MessageId**

Per impostare: *MQMessage*.**MessageId** = *messageid $*

Consultare “Proprietà descrittori messaggi” a pagina 1037 per ulteriori informazioni su quando è necessario utilizzare MessageIdHex al posto della proprietà MessageId .

proprietà Hex MessageId

Lettura - scrittura. Il MessageId da includere nell'MQMD di un messaggio quando viene inserito su una coda. Inoltre, l' MessageId a cui far corrispondere quando si riceve un messaggio da una coda.

Ogni due caratteri della stringa rappresentano l'equivalente esadecimali di un singolo carattere ASCII. Ad esempio, la coppia di caratteri "6" e "1" rappresentano il singolo carattere "A", la coppia di caratteri "6" e "2" rappresentano il singolo carattere "B" e così via.

È necessario fornire 48 caratteri esadecimali validi.

Il valore iniziale è "0 ... 0".

Definito in: classe MQMessage

Tipo di dati: stringa di 48 caratteri esadecimali che rappresentano 24 caratteri ASCII

Sintassi: per ottenere *messageidh $= MQMessage*.**MessageIdHex**

Per impostare: *MQMessage*.**MessageIdHex** = *messageidh $*

Consultare “Proprietà descrittori messaggi” a pagina 1037 per ulteriori informazioni su quando è necessario utilizzare MessageIdHex al posto della proprietà MessageId .

proprietà Numero MessageSequence

Lettura - scrittura. Informazioni di sequenza che identificano un messaggio all'interno di un gruppo. Il valore iniziale è 1.

Definito in:
Classe MQMessage

Tipo di dati:
Lungo

Valori:
Vedere MsgSeqNumber (MQLONG).

Sintassi: per ottenere *sequencenumber & = MQMessage*.**SequenceNumber**

Per impostare: *MQMessage*.**SequenceNumber** = *sequencenumber &*

proprietà MessageType

Lettura - scrittura. Il campo MQMD MsgType .

Il valore iniziale è MQMT_DATAGRAM.

Definito in: classe MQMessage

Tipo di dati: Long

Valori:

- Vedere MsgType (MQLONG).

Sintassi: per ottenere: *msgtype & = MQMessage*.**MessageType**

Per impostare: *MQMessage*.**MessageType** = *msgtype &*

proprietà Offset

Lettura - scrittura. L'offset in un messaggio segmentato. Il valore iniziale è 0.

Definito in:
Classe MQMessage

Tipo di dati:
Lungo

Valori:
Vedere Offset (MQLONG).

Sintassi: per ottenere *offset & = MQMessage*.**Offset**

Per impostare: *MQMessage*.**Offset** = *offset &*

proprietà OriginalLength

Lettura - scrittura. La lunghezza originale di un messaggio segmentato. Il valore iniziale è MQOL_UNDEFINED.

Definito in:
Classe MQMessage

Tipo di dati:
Lungo

Valori:
Vedere OriginalLength (MQLONG).

Sintassi: per ottenere *originallength & = MQMessage*.**OriginalLength**

Per impostare: *MQMessage*.**OriginalLength** = *originallength &*

proprietà Persistenza

Lettura - scrittura. L'impostazione di persistenza del messaggio.

Il valore iniziale è MQPER_PERSISTENCE_AS_Q_DEF.

Definito in: classe MQMessage

Tipo di dati: Long

Sintassi: per ottenere *persist & = MQMessage***.Persistenza**

Per impostare: *MQMessage***.Persistenza** = *persist &*

proprietà Priorità

Lettura - scrittura. La priorità del messaggio.

Il valore iniziale è il valore speciale MQPRI_PRIORITY_AS_Q_DEF

Definito in: classe MQMessage

Tipo di dati: Long

Sintassi: per richiamare *priority & = MQMessage***.Priorità**

Per impostare: *MQMessage***.Priorità** = *priorità &*

proprietà Nome PutApplication

Lettura - scrittura. Il nome MQMD PutAppl- parte del contesto di origine del messaggio.

Il valore iniziale è costituito da tutti spazi.

Definito in: classe MQMessage

Tipo di dati: stringa di 28 caratteri

Sintassi: per richiamare: *putapplnm $= MQMessage***.PutApplicationName**

Per impostare *MQMessage***.PutApplicationNome** = *putapplnm $*

proprietà Tipo PutApplication

Lettura - scrittura. Il tipo MQMD PutAppl- parte del contesto di origine del messaggio.

Il valore iniziale è MQAT_NO_CONTEXT

Definito in: classe MQMessage

Tipo di dati: Long

Valori:

- Vedere PutAppl(MQLONG).

Sintassi: per ottenere *putappltp & = MQMessage***.PutApplicationTipo**

Per impostare: *MQMessage***.PutApplicationTipo** = *putappltp &*

proprietà PutDate

Lettura / scrittura. Questa proprietà combina i campi MQMD PutDate e PutTime . Queste sono parti del contesto Origine messaggio che indicano quando è stato inserito il messaggio.

L'estensione ActiveX converte il formato data / ora ActiveX e i formati Data e ora utilizzati in WebSphere MQ MQMD. Se viene ricevuto un messaggio che ha un valore PutDate o PutTimenon valido, la proprietà PutDateTime dopo il metodo get è impostata su EMPTY.

Il valore iniziale è EMPTY.

Definito in: classe MQMessage

Tipo di dati: variante di tipo 7 (data/ora) o EMPTY.

Sintassi: per ottenere *datetime = MQMessage*.**PutDateTime**

Per impostare: *MQMessage*.**PutDateTime** = *datetime*

Proprietà Nome ReplyToQueueManager

Lettura - scrittura. Il campo Gestore code MQMD ReplyTo.

Il suo valore iniziale è costituito da tutti spazi

Definito in: classe MQMessage

Tipo di dati: stringa di 48 caratteri

Sintassi: Per ottenere: *replytoqmgr $= MQMessage*.**ReplyToQueueManagerName**

Per impostare: *MQMessage*.**ReplyToQueueManagerName** = *replytoqmgr $*

Proprietà ReplyToQueueName

Lettura - scrittura. Il campo MQMD ReplyToQ.

Il suo valore iniziale è costituito da tutti spazi

Definito in: classe MQMessage

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere *replytoq $= MQMessage*.**ReplyToQueueName**

Per impostare: *MQMessage*.**ReplyToQueueName** = *replytoq $*

proprietà Report

Lettura - scrittura. Le opzioni Report del messaggio.

Il valore iniziale è MQRO_NONE.

Definito in: classe MQMessage

Tipo di dati: Long

Valori:

- Vedere Report.

Sintassi: per ottenere *report & = MQMessage*.**Report**

Per impostare: *MQMessage*.**Report** = *report &*

proprietà Lunghezza TotalMessage

Sola lettura. Richiama la lunghezza dell'ultimo messaggio ricevuto da MQGET. Se il messaggio non è stato troncato, questo valore è uguale al valore della proprietà MessageLength .

Definito in: classe MQMessage

Tipo di dati: Long

Sintassi: per ottenere *totalmessagelength & = MQMessage*.**TotalMessageLength**

Proprietà UserID

Lettura - scrittura. MQMD UserIdentifier - parte del contesto di identità del messaggio.

Il valore iniziale è costituito da tutti spazi.

Definito in: classe MQMessage

Tipo di dati: stringa di 12 caratteri

Sintassi: per ottenere *userid $= MQMessage*.**UserId**

Per impostare: *MQMessage*.**UserId** = *userid $*

Metodo di codici ClearError

Reimposta CompletionCode su MQCC_OK e ReasonCode su MQRC_NONE per le classi MQMessage e MQSession.

Definito in: classe MQMessage

Sintassi: richiamare *MQMessage*.**ClearErrorCodes ()**

Metodo ClearMessage

Questo metodo cancella la parte del buffer di dati dell'oggetto MQMessage. Qualsiasi dato del messaggio nel buffer di dati viene perso, perché MessageLength, DataLengthe DataOffset sono tutti impostati a zero.

La parte MQMD (Message Descriptor) non viene influenzata; è possibile che un'applicazione debba modificare alcuni campi MQMD prima di riutilizzare l'oggetto MQMessage. Per impostare nuovamente i campi MQMD, utilizzare Nuovo per sostituire l'oggetto con una nuova istanza.

Definito in: classe MQMessage

Sintassi: Call *MQMessage*.**ClearMessage()**

Metodo di lettura

Legge una sequenza di byte dal buffer di messaggi in una schiera di byte. DataOffset viene incrementato e la lunghezza dei dati decrementata dal numero di byte letti.

Definito in:
Classe MQMessage

Sintassi Data = MQMessage.**Read**(len &)

Parametri:
len &: Lungo. Lunghezza dei dati in byte da leggere.

metodo ReadBoolean

Legge un valore booleano a 1 byte dalla posizione corrente nel buffer di messaggi e restituisce un valore booleano a 2 - byte TRUE (-1) /FALSE (0). DataOffset viene incrementato di uno e la lunghezza dei dati viene ridotta di uno.

Definito in:
Classe MQMessage

Sintassi: *value = MQMessage*.**ReadBoolean**

metodo ReadByte

Questo metodo legge 1 byte dal buffer di dati del messaggio, iniziando con il carattere a cui fa riferimento DataOffset e lo restituisce come un valore intero (con segno a 2 byte) compreso nell'intervallo tra -128 e 127.

Il metodo non riesce se MQMessage.DataLength è minore di 1 quando viene emesso.

DataOffset viene incrementato di 1 e DataLength viene decrementato di 1 se il metodo riesce.

Si presuppone che il byte dei dati del messaggio sia un numero intero binario con segno.

Definito in: classe MQMessage

Sintassi: *integerv% = MQMessage*.**ReadByte**

metodo ReadDecimal2

Legge un numero decimale compresso a 2 byte e lo restituisce come un valore intero a 2 byte con segno. DataOffset viene incrementato di due e la lunghezza dei dati viene ridotta di due.

Definito in:
Classe MQMessage

Sintassi: *value% = MQMessage*.**ReadDecimal2**

metodo ReadDecimal4

Legge un numero decimale compresso a 4 byte e lo restituisce come valore intero a 4 byte con segno. DataOffset viene incrementata di quattro e la Lunghezza dati viene ridotta di quattro.

Definito in:
Classe MQMessage

Sintassi: valore *chiamata & = MQMessage*.**ReadDecimal4**

metodo ReadDouble

Questo metodo legge 8 byte dal buffer di dati dei messaggi, iniziando con il byte a cui fa riferimento DataOffset e lo restituisce come valore a virgola mobile doppio (con segno a 8 byte).

Il metodo ha esito negativo se MQMessage.DataLength è inferiore a 8 quando viene emesso.

DataOffset viene incrementato di 8 e DataLength viene decrementato di 8 se il metodo riesce.

Si presume che gli 8 caratteri dei dati del messaggio siano un numero a virgola mobile binario. La codifica è specificata dalla proprietà MQMessage.Encoding . Notare che la conversione dal formato System/360 non è supportata.

Definito in: classe MQMessage

Sintassi: *doublev# = MQMessage*.**ReadDouble**

metodo ReadDouble4

I metodi ReadDouble4 e WriteDouble4 sono alternative a ReadFloat e WriteFloat. Questo perché supportano valori di messaggi a virgola mobile System/390 a 4 byte troppo grandi per essere convertiti in formato a virgola mobile IEEE a 4 byte.

Questo metodo legge 4 byte dal buffer di dati del messaggio, iniziando con il byte a cui fa riferimento DataOffset e lo restituisce come valore a virgola mobile Double (con segno a 8 byte).

Il metodo ha esito negativo se MQMessage.DataLength è inferiore a 4 quando viene emesso.

DataOffset viene incrementato di 4 e DataLength viene ridotto di 4 se il metodo riesce.

Si presume che i 4 caratteri dei dati del messaggio siano un numero a virgola mobile binario. La codifica è specificata dalla proprietà MQMessage.Encoding . Notare che la conversione dal formato System/360 non è supportata.

Definito in: classe MQMessage

Sintassi: *doublev# = MQMessage.***ReadDouble4**

metodo ReadFloat

Questo metodo legge 4 byte dal buffer di dati dei messaggi, iniziando con il byte a cui fa riferimento DataOffset e lo restituisce come un valore a virgola mobile singolo (a 4 byte con segno).

Il metodo ha esito negativo se MQMessage.DataLength è inferiore a 4 quando viene emesso.

DataOffset viene incrementato di 4 e DataLength viene ridotto di 4 se il metodo riesce.

Si presume che i 4 caratteri dei dati del messaggio siano un numero a virgola mobile. La codifica è specificata dalla proprietà MQMessage.Encoding . Notare che la conversione dal formato System/360 non è supportata.

Definito in: classe MQMessage

Sintassi: *singlev! = MQMessage*.**ReadFloat**

metodo ReadInt2

Il metodo è identico al metodo ReadShort .

Sintassi: *integerv% = MQMessage*.**Lettura2**

metodo ReadInt4

Questo metodo è identico al metodo ReadLong .

Sintassi: *bigint & = MQMessage*.**Lettura4**

metodo ReadLong

Questo metodo legge 4 byte dal buffer di dati del messaggio, iniziando con il byte a cui fa riferimento DataOffset e lo restituisce come valore intero Long (con segno a 4 byte).

Il metodo ha esito negativo se MQMessage.DataLength è inferiore a 4 quando viene emesso.

DataOffset viene incrementato di 4 e DataLength viene ridotto di 4 se il metodo riesce.

Si presume che i 4 caratteri dei dati del messaggio siano un numero intero binario. La codifica è specificata dalla proprietà MQMessage.Encoding .

Definito in: classe MQMessage

Sintassi: *bigint & = MQMessage*.**ReadLong**

metodo ReadNullTerminatedString

Questo metodo deve essere utilizzato al posto di ReadString se la stringa potrebbe contenere caratteri null incorporati.

Questo metodo legge il numero specificato di byte dal buffer di dati del messaggio a partire dal byte a cui fa riferimento DataOffset e lo restituisce come stringa ActiveX . Se la stringa contiene un valore null incorporato prima della fine, la lunghezza della stringa restituita viene ridotta per riflettere solo quei caratteri prima del valore null.

DataOffset viene incrementato e DataLength viene ridotto dal valore specificato indipendentemente dal fatto che la stringa contenga caratteri null incorporati.

I caratteri nei dati del messaggio vengono considerati come una stringa nella codepage specificata dalla proprietà MQMessage.CharacterSet . La conversione in rappresentazione ActiveX viene eseguita per l'applicazione.

Definito in:
Classe MQMessage

Sintassi: *string $ = MQMessage*.**ReadNullTerminatedString(length &)**

Parametri:
lunghezza & *Lungo.* Lunghezza del campo stringa in byte.

metodo ReadShort

Questo metodo legge 2 byte dal buffer di dati del messaggio, iniziando con il byte a cui fa riferimento DataOffset e lo restituisce come valore intero (con segno a 2 byte).

Il metodo ha esito negativo se MQMessage.DataLength è inferiore a 2 quando viene emesso.

DataOffset viene incrementato di 2 e DataLength viene decrementato di 2 se il metodo riesce.

Si presume che i 2 caratteri dei dati del messaggio siano un numero intero binario. La codifica è specificata dalla proprietà MQMessage.Encoding .

Definito in: classe MQMessage

Sintassi: *integerv% = MQMessage*.**ReadShort**

metodo ReadString

Questo metodo legge n byte dal buffer di dati dei messaggi iniziando con il byte a cui fa riferimento DataOffset e lo restituisce come stringa ActiveX .

Il metodo ha esito negativo se MQMessage.DataLength è inferiore a n quando viene emesso.

DataOffset viene incrementato di n e DataLength viene decrementato di n se il metodo riesce.

Si presume che gli n caratteri dei dati del messaggio siano una stringa nella codepage specificata dalla proprietà MQMessage.CharacterSet . La conversione in rappresentazione ActiveX viene eseguita per l'applicazione.

Definito in: classe MQMessage

Sintassi: *stringv $= MQMessage*.**ReadString**(*lunghezza &*)

Parametro
lunghezza & Long. Lunghezza del campo stringa in byte.

metodo ReadUInt2

Questo metodo legge 2 byte dal buffer di dati del messaggio, iniziando con il byte indicato da DataOffset e restituendolo come un valore intero Long (con segno a 4 byte).

Il metodo ha esito negativo se MQMessage.DataLength è inferiore a 2 quando viene emesso.

DataOffset viene incrementato di 2 e DataLength viene decrementato di 2 se il metodo riesce.

Si presuppone che i 2 byte dei dati del messaggio siano un numero intero binario senza segno. La codifica è specificata dalla proprietà MQMessage.Encoding .

Definito in: classe MQMessage

Sintassi: *bigint & = MQMessage*.**ReadUInt2**

metodo ReadUnsignedByte

Questo metodo legge 1 byte dal buffer di dati del messaggio, iniziando con il byte a cui fa riferimento DataOffset e lo restituisce come un valore intero (con segno a 2 byte) compreso nell'intervallo tra 0 e 255.

Il metodo non riesce se MQMessage.DataLength è minore di 1 quando viene emesso.

DataOffset viene incrementato di 1 e DataLength viene decrementato di 1 se il metodo riesce.

Si presuppone che 1 carattere dei dati del messaggio sia un numero intero binario senza segno.

Definito in: classe MQMessage

Sintassi: *integerv% = MQMessage*.**ReadUnsignedByte**

metodo ReadUTF

Questo metodo legge una stringa di formato UTF dal messaggio che inizia con il byte a cui fa riferimento DataOffset e la restituisce come stringa ActiveX . La stringa nel messaggio è composta da una lunghezza di 2 byte seguita dai dati carattere.

Il metodo ha esito negativo se MQMessage.DataLength è inferiore alla lunghezza della stringa quando viene emesso.

DataOffset viene incrementato dalla lunghezza della stringa e DataLength viene ridotto dalla lunghezza della stringa se il metodo riesce.

Definito in:
Classe MQMessage

Sintassi: *value $= MQMessage*.**ReadUTF**

Metodo ResizeBuffer

Questo metodo modifica la quantità di memoria attualmente assegnata internamente per contenere il buffer di dati del messaggio. Fornisce all'applicazione un certo controllo sulla gestione del buffer automatico, in quanto se l'applicazione sa che sta per gestire un messaggio di grandi dimensioni, può garantire che venga assegnato un buffer sufficientemente grande. L'applicazione non ha bisogno di utilizzare questa chiamata - in caso contrario, il codice di gestione del buffer automatico aumenterà la dimensione del buffer per adattarla.

Se si ridimensiona il buffer in modo che sia più piccolo rispetto alla MessageLengthcorrente, si rischia di perdere i dati. Se si perdono i dati, il metodo restituisce un CompletionCode di MQCC_WARNING e un ReasonCode di MQRC_DATA_TRUNCATED.

Se si ridimensiona il buffer in modo che sia inferiore al valore della proprietà **DataOffset** :

- La proprietà **DataOffset** viene modificata per puntare alla fine del nuovo buffer
- La proprietà **DataLength** è impostata su zero
- La proprietà **MessageLength** viene modificata nella nuova dimensione del buffer

Definito in:
Classe MQMessage

Sintassi: *MQMessage*.**ResizeBuffer**(*Length &*)

Parametro:
Lunghezza & Lunga. Dimensione richiesta in caratteri.

Metodo di scrittura

Scrive una sequenza di byte nel buffer di messaggi da una schiera di byte nella posizione a cui fa riferimento l'offset di dati. Se necessario, la lunghezza del buffer (MQMessage.MQMessageLength) viene estesa per contenere l'intera lunghezza della matrice di byte. DataOffset viene incrementato del numero di byte scritti se il metodo ha esito positivo.

Definito in:
Classe MQMessage

Sintassi: Call *MQMessage*.**Write**(valore)

Parametri:
data: un array di byte o un riferimento variante a un array di byte

metodo WriteBoolean

Scrive un valore booleano a 1 byte nella posizione corrente nel buffer di messaggi da un valore booleano a 2 byte. DataOffset viene incrementato di uno.

Definito in:
Classe MQMessage

Sintassi: Call *MQMessage*.**WriteBoolean**(valore)

Parametro:
valore: booleano (2-byte). Valore da scrivere.

metodo WriteByte

Questo metodo acquisisce un valore intero a 2 byte con segno e lo scrive nel buffer dei dati di messaggio come un numero binario a 1 byte nella posizione indicata da DataOffset. Sostituisce tutti i dati già presenti nella posizione nel buffer ed estende la lunghezza del buffer (MQMessage.MessageLength), se necessario.

DataOffset viene incrementato di uno se il metodo ha esito positivo.

Il valore specificato deve essere compreso tra -128 e 127. In caso contrario, il metodo restituisce CompletionCode MQCC_FAILED e ReasonCode MQRC_WRITE_VALUE_ERROR.

Definito in: classe MQMessage

Sintassi: Call *MQMessage*.**WriteByte**(*value%*)

Parametro *value%* Numero intero. Valore da scrivere.

metodo WriteDecimal2

Scrive un numero intero a 2 byte con segno come numero decimale compresso a 2 byte. DataOffset viene incrementata di due.

Definito in:
Classe MQMessage

Sintassi: Call *MQMessage*.**WriteDecimal2**(valore%)

Parametro:
valore% intero. Valore da scrivere.

metodo WriteDecimal4

Scrive un numero intero a 4 byte con segno come numero decimale compresso a 4 byte. DataOffset viene incrementata di quattro.

Definito in:
Classe MQMessage

Sintassi: chiamare *MQMessage*.**WritedDecimal4**(valore &)

Parametro:
valore & Lungo. Valore da scrivere.

metodo WriteDouble

Questo metodo utilizza un valore a virgola mobile di 8 byte con segno e lo scrive nel buffer dei dati dei messaggi come un numero a virgola mobile di 8 byte a partire dalla posizione indicata da DataOffset. Sostituisce tutti i dati già presenti in queste posizioni nel buffer ed estende la lunghezza del buffer (MQMessage.MessageLength) se necessario.

DataOffset viene incrementato di 8 se il metodo ha esito positivo.

Il metodo viene convertito nella rappresentazione a virgola mobile specificata dalla proprietà MQMessage.Encoding . *Conversione nel formato System/360 non supportata.*

Definito in: classe MQMessage

Sintassi: Call *MQMessage*.**WriteDouble**(*value#*)

Parametro:
valore# Doppio. Valore da scrivere.

metodo WriteDouble4

Consultare “metodo ReadDouble4” a pagina 1086 per una descrizione di quando ReadDouble4 e WriteDouble4 devono essere utilizzati al posto di ReadFloat e WriteFloat.

Questo metodo utilizza un valore a virgola mobile a 8 byte con segno e lo scrive nel buffer dei dati dei messaggi come un numero mobile a 4 byte a partire dalla posizione a cui fa riferimento DataOffset.

DataOffset viene incrementato di 4 se il metodo ha esito positivo.

Sostituisce tutti i dati già presenti in queste posizioni nel buffer ed estende la lunghezza del buffer (MQMessage.MessageLength) se necessario.

Il metodo viene convertito nella rappresentazione a virgola mobile specificata dalla proprietà MQMessage.Encoding . *Conversione nel formato System/360 non supportata.*

Definito in: classe MQMessage

Sintassi: Call *MQMessage*.**WriteDouble4 (value#)**

Parametro *value#* Double. Valore da scrivere.

metodo WriteFloat

Questo metodo utilizza un valore a virgola mobile a 4 byte con segno e lo scrive nel buffer dei dati di messaggio come un numero a virgola mobile a 4 byte a partire dal carattere a cui fa riferimento DataOffset. Sostituisce tutti i dati già presenti in queste posizioni nel buffer ed estende la lunghezza del buffer (MQMessage.MessageLength) se necessario.

DataOffset viene incrementato di 4 se il metodo ha esito positivo.

Il metodo si converte nella rappresentazione binaria specificata dalla proprietà MQMessage.Encoding . *Conversione nel formato System/360 non supportata.*

Definito in: classe MQMessage

Sintassi: richiamare *MQMessage*.**WriteFloat**(valore*!*)

Parametro *valore!* Mobile. Valore da scrivere.

metodo WriteInt2

Questo metodo è identico al metodo WriteShort .

Sintassi: richiamare *MQMessage*.**WriteInt2**(*valore%*)

Parametro *valore%* Numero intero. Valore da scrivere.

metodo WriteInt4

Questo metodo è identico al metodo WriteLong .

Sintassi: richiamare *MQMessage*.**WriteInt4**(valore*&*)

Parametro *valore &* Long. Valore da scrivere.

metodo WriteLong

Questo metodo acquisisce un valore intero a 4 byte con segno e lo scrive nel buffer di dati del messaggio come un numero binario a 4 byte a partire dal byte a cui fa riferimento DataOffset. Sostituisce tutti i dati già presenti in queste posizioni nel buffer ed estende la lunghezza del buffer (MQMessage.MessageLength) se necessario.

DataOffset viene incrementato di 4 se il metodo ha esito positivo.

Il metodo si converte nella rappresentazione binaria specificata dalla proprietà MQMessage.Encoding .

Definito in: classe MQMessage

Sintassi: chiamare *MQMessage*.**WriteLong**(*value &*)

Parametro *valore &* Long. Valore da scrivere.

metodo WriteNullTerminatedString

Questo metodo esegue una normale WriteString ed esegue il riempimento dei byte rimanenti fino alla lunghezza specificata con un valore null. Se il numero di byte scritti dalla stringa di scrittura iniziale è

uguale alla lunghezza specificata, non vengono scritti valori null. Se il numero di byte supera la lunghezza specificata, viene impostato un errore (codice motivo MQRC_WRITE_VALUE_ERROR).

DataOffset viene incrementato della lunghezza specificata se il metodo ha esito positivo.

Definito in: classe MQMessage

Sintassi: Call *MQMessage*.**WriteNullTerminatedString***(value$, length &)*

Parametri:
valore $String. Valore da scrivere.

lunghezza & Lungo. Lunghezza del campo stringa in byte.

Metodo WriteShort

Questo metodo acquisisce un valore intero a 2 byte con segno e lo scrive nel buffer dei dati dei messaggi come un numero binario a 2 byte a partire dal byte a cui fa riferimento DataOffset. Sostituisce tutti i dati già presenti in queste posizioni nel buffer e, se necessario, estenderà la lunghezza del buffer (MQMessage.MessageLength).

DataOffset viene incrementato di 2 se il metodo ha esito positivo.

Il metodo si converte nella rappresentazione binaria specificata dalla proprietà MQMessage.Encoding .

Definito in: classe MQMessage

Sintassi: Richiamare *MQMessage***.WriteShort**(*value%*)

Parametro *valore%* Numero intero. Valore da scrivere.

metodo WriteString

Questo metodo prende una stringa ActiveX e la scrive nel buffer di dati del messaggio a partire dal byte a cui fa riferimento DataOffset. Sostituisce tutti i dati già presenti in queste posizioni nel buffer e, se necessario, estenderà la lunghezza del buffer (MQMessage.MessageLength).

DataOffset viene incrementato della lunghezza della stringa in byte se il metodo ha esito positivo.

Il metodo converte i caratteri nella codepage specificata dalla proprietà MQMessage.CharacterSet .

Definito in: classe MQMessage

Sintassi: richiamare *MQMessage*.**WriteString**(*value $*)

Parametro *Valore $* Stringa. Valore da scrivere.

metodo WriteUInt2

Questo metodo acquisisce un valore intero a 4 byte con segno e lo scrive nel buffer di dati del messaggio come un numero binario senza segno a 2 byte a partire dal byte a cui fa riferimento DataOffset. Sostituisce tutti i dati già presenti in queste posizioni nel buffer ed estende la lunghezza del buffer (MQMessage.MessageLength) se necessario.

DataOffset viene incrementato di 2 se il metodo ha esito positivo.

Il metodo si converte nella rappresentazione binaria specificata dalla proprietà MQMessage.Encoding . Il valore specificato deve essere compreso tra 0 e 2 * *16-1. Se non è il metodo viene restituito con CompletionCode MQCC_FAILED e ReasonCode MQRC_WRITE_VALUE_ERROR.

Definito in: classe MQMessage

Sintassi: richiamare *MQMessage***.WriteUInt2**(valore*&*)

Parametro *valore &* Long. Valore da scrivere.

Metodo byte WriteUnsigned

Questo metodo acquisisce un valore intero a 2 byte con segno e lo scrive nel buffer di dati del messaggio come un numero binario senza segno a 1 byte a partire dal carattere a cui fa riferimento DataOffset. Sostituisce tutti i dati già presenti in queste posizioni nel buffer ed estende la lunghezza del buffer (MQMessage.MessageLength) se necessario.

DataOffset viene incrementato di 1 se il metodo ha esito positivo.

Il valore specificato deve essere compreso tra 0 e 255. Se non è il metodo viene restituito con CompletionCode MQCC_FAILED e ReasonCode MQRC_WRITE_VALUE_ERROR.

Definito in:
Classe MQMessage

Sintassi: richiamare *MQMessage*.**WriteUnsignedByte**(*value%*)

Parametro *valore%* Numero intero. Valore da scrivere.

Metodo WriteUTF

Questo metodo prende una stringa ActiveX e la scrive nel buffer di dati del messaggio nella posizione corrente in formato UTF. I dati scritti sono composti da una lunghezza di 2 byte seguita dai dati carattere. DataOffset viene incrementato della lunghezza della stringa se il metodo ha esito positivo.

Definito in:
Classe MQMessage

Sintassi: Call *MQMessage*.**WriteUTF**(value$)

Parametro:
valore $String. Valore da scrivere.

classe di opzioni MQPutMessage

Questa classe racchiude le varie opzioni che controllano l'azione di inserimento di un messaggio in una coda WebSphere MQ .

Contenimento

La classe Opzioni MQPutMessageè contenuta nella classe MQSession.

Creazione

Nuovo crea un nuovo oggetto MQPutMessageOptions e imposta le relative proprietà sui valori iniziali.

In alternativa, utilizzare il metodo AccessPutMessageOptions della classe MQSession.

Sintassi

Dim *pmo* **Come Nuovo MQPutMessageOpzioni** o

Set *pmo* = **Nuovo MQPutMessageOpzioni**

Proprietà

- “proprietà CompletionCode” a pagina 1094.
- “proprietà Opzioni” a pagina 1094.
- “proprietà ReasonCode” a pagina 1094.
- “proprietà ReasonName” a pagina 1094.
- “proprietà RecordFields” a pagina 1094.
- “proprietà ResolvedQueueManagerName” a pagina 1095.
- “proprietà Nome ResolvedQueue” a pagina 1095.

Metodi

- “Metodo di codici ClearError” a pagina 1095.

proprietà CompletionCode

Sola lettura. Restituisce il codice di completamento impostato dall'ultimo metodo o accesso alla proprietà emesso rispetto all'oggetto.

Definito in: MQPutMessageclasse Options

Tipo di dati: Long

Valori:

- MQCC_OK
- MQCC_AVVERTENZA
- MQCC_NON RIUSCITO

Sintassi: per ottenere: *completioncode & = PutOpts*.**CompletionCode**

proprietà Opzioni

Lettura - scrittura. Il campo Opzioni MQPM. Il valore iniziale di questo campo è MQPMO_NONE. Per ulteriori informazioni, consultare Opzioni MQPMO.

Definito in: MQPutMessageOptions Class.

Tipo di dati: Long

Sintassi: per richiamare: *options & = PutOpts*.**Opzioni**

Per impostare: *PutOpts*.**Opzioni** = *opzioni &*

Le opzioni MQPMO_PASS_IDENTITY_CONTEXT e MQPMO_PASS_ALL_CONTEXT non sono supportate.

proprietà ReasonCode

Sola lettura. Restituisce il codice motivo impostato dall'ultimo metodo o accesso alla proprietà emesso per l'oggetto.

Definito in: MQPutMessageclasse Options

Tipo di dati: Long

Valori:

- Vedere codici di errore API.

Sintassi: per ottenere: *reasoncode & = PutOpts*.**ReasonCode**

proprietà ReasonName

Sola lettura. Restituisce il nome simbolico dell'ultimo codice motivo. Ad esempio, "MQRC_QMGR_NOT_AVAILABLE".

Definito in: MQPutMessageclasse Options

Tipo di dati: stringa

Valori:

- Vedere codici di errore API.

Sintassi: Per richiamare: *reasonname $= PutOpts*.**ReasonName**

proprietà RecordFields

Lettura - scrittura. Indicatori che indicano quali campi devono essere personalizzati in base alla coda durante l'inserimento di un messaggio in un elenco di distribuzione. Il valore iniziale è zero.

Questa proprietà corrisponde agli indicatoriRecFields PutMsgnella struttura MQI MQPMO. In MQI, questi indicatori controllano quali campi (nella struttura MQPMR) sono presenti e utilizzati da MQPUT. In un oggetto Opzioni MQPutMessage, questi campi sono sempre presenti e gli indicatori, pertanto, influenzano solo i campi utilizzati da Put. Per ulteriori dettagli, consultare il manuale *WebSphere MQ Application Programming Reference* .

Definito in:
classe di opzioni MQPutMessage

Tipo di dati:
Lungo

Sintassi: per ottenere *recordfields & = PutOpts***.RecordFields**

Per impostare: *PutOpts*.**RecordFields** = *recordfields &*

proprietà ResolvedQueueManagerName

Sola lettura. Il campo Nome MQPMO ResolvedQMgr. Consultare ResolvedQMgrName (MQCHAR48) per i dettagli. Il valore iniziale è costituito da tutti spazi.

Definito in: MQPutMessageclasse Options

Tipo di dati: stringa di 48 caratteri

Sintassi : *qmgr $= PutOpts***.ResolvedQueueManagerName**

proprietà Nome ResolvedQueue

Sola lettura. Il campo MQPMO ResolvedQName . Per i dettagli, consultare ResolvedQName (MQCHAR48) . Il valore iniziale è costituito da tutti spazi.

Definito in: MQPutMessageclasse Options

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere *qname $= PutOpts***.ResolvedQueueName**

Metodo di codici ClearError

Reimposta il CompletionCode su MQCC_OK e il ReasonCode su MQRC_NONE per la classe Opzioni MQPutMessagee la classe MQSession.

Definito in: MQPutMessageclasse Options

Sintassi: Call *PutOpts***.ClearErrorCodes ()**

classe di opzioni MQGetMessage

Questa classe incapsula le varie opzioni che controllano l'azione di richiamo di un messaggio da una coda WebSphere MQ .

Contenimento

La classe di opzioni MQGetMessageè contenuta nella classe MQSession.

Creazione

Nuovo crea un nuovo oggetto Opzioni MQGetMessagee imposta tutte le proprietà sui valori iniziali.

In alternativa, utilizzare il metodo AccessGetMessageOptions della classe MQSession.

Proprietà

Metodi

Sintassi

Dim *gmo* **As New MQGetMessageOptions** o

Set *gmo* = **New MQGetMessageOptions**

proprietà CompletionCode

Sola lettura. Restituisce il codice di completamento impostato dall'ultimo metodo o accesso alla proprietà emesso rispetto all'oggetto.

Definito in MQGetMessageOptions Class.

Tipo di dati: Long

Valori:

- MQCC_OK
- MQCC_AVVERTENZA
- MQCC_NON RIUSCITO

Sintassi: per ottenere *completioncode & = GetOpts*.**CompletionCode**

proprietà MatchOptions

Lettura - scrittura. Opzioni che controllano i criteri di selezione utilizzati per MQGET. Il valore iniziale è MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID.

Definito in:
classe di opzioni MQGetMessage

Tipo di dati:
Lungo

Valori:
Vedere MatchOptions (MQLONG).

Sintassi: per ottenere *matchoptions & = GetOpts*.**MatchOptions**

Per impostare: *GetOpts*.**MatchOptions** =*matchoptions &*

proprietà Opzioni

Lettura - scrittura. Il campo Opzioni MQGMO. Consultare Opzioni per i dettagli. Il valore iniziale è MQGMO_NO_WAIT.

Definito in MQGetMessageOptions Class.

Tipo di dati: Long

Sintassi Per ottenere: *options & = GetOpts*.**Opzioni** Per impostare: *GetOpts*.**Opzioni** = *opzioni &*

proprietà ReasonCode

Sola lettura. Restituisce il codice motivo impostato dall'ultimo metodo o accesso alla proprietà emesso per l'oggetto.

Definito in: classe di opzioni MQGetMessage

Tipo di dati: Long

Valori:

- Vedere codici di errore API.

Sintassi: per ottenere *reasoncode & = GetOpts*.**ReasonCode**

proprietà ReasonName

Sola lettura. Restituisce il nome simbolico dell'ultimo codice motivo. Ad esempio, "MQRC_QMGR_NOT_AVAILABLE". **Definito in:** classe di opzioni MQGetMessage

Tipo di dati: stringa

Valori:

- Vedere codici di errore API.

Sintassi: per richiamare *reasonname $= MQGetMessageOptions*.**ReasonName**

proprietà Nome ResolvedQueue

Sola lettura. Il campo MQGMO ResolvedQName . Per i dettagli, consultare ResolvedQName (MQCHAR48) . Il valore iniziale è costituito da tutti spazi.

Definito in: classe di opzioni MQGetMessage

Tipo di dati: stringa di 48 caratteri

Sintassi: per ottenere *qname $= GetOpts*.**ResolvedQueueName**

proprietà WaitInterval

Lettura / scrittura. Il campo MQGMO WaitInterval . Il tempo massimo, in millisecondi, in cui Get attende l'arrivo di un messaggio adatto - se l'azione di attesa è stata richiesta dalla proprietà Options. Questo campo ha un valore iniziale di 0. Per dettagli sulle opzioni MQGMO, consultare MQGMO.

Definito in: classe di opzioni MQGetMessage

Tipo di dati: Long

Sintassi: per ottenere *wait & = GetOpts*.**WaitInterval**

Per impostare: *GetOpts*.**WaitInterval** = wait &

Metodo di codici ClearError

Reimposta il CompletionCode su MQCC_OK e il ReasonCode su MQRC_NONE sia per la classe di opzioni MQGetMessageche per la classe MQSession.

Definito in: classe di opzioni MQGetMessage

Sintassi: Call *GetOpts*.**ClearErrorCodes ()**

Classe MQDistributionList

Questa classe incapsula una raccolta di code - locali, remote o alias per l'emissione.

Creazione

new crea un nuovo oggetto MQDistributionList .

In alternativa, utilizzare il metodo AddDistributionList della classe MQQueueManager

Proprietà

Metodo

Sintassi

Dim *distlist.***As New MQDistributionList** o **Set** *distlist* = **New MQDistributionList**

proprietà ID AlternateUser

Lettura - scrittura. L'ID utente alternativo utilizzato per convalidare l'accesso all'elenco di code quando vengono aperte.

Definito in:
Classe MQDistributionList

Tipo di dati:
Stringa di 12 caratteri

Sintassi: per ottenere *altuser $= MQDistributionList*.**AlternateUserId**

Per impostare: *MQDistributionList*.**AlternateUserId** = *altuser $*

proprietà CloseOptions

Lettura - scrittura. Opzioni utilizzate per controllare cosa accade quando l'elenco di distribuzione viene chiuso. Il valore iniziale è MQCO_NONE.

Definito in:
Classe MQDistributionList

Tipo di dati:
Lungo

Valori:

- MQCO_NONE
- MQCO_DELETE

- MQCO_DELETE_PURGE

Sintassi: per ottenere: *closeopt & = MQDistributionList*.**CloseOptions**

Per impostare: *MQDistributionList*.**CloseOptions** = *closeopt &*

proprietà CompletionCode

Sola lettura. Il codice di completamento impostato dall'ultimo accesso metodo o proprietà emesso rispetto all'oggetto.

Definito in:
Classe MQDistributionList

Tipo di dati:
Lungo

Valori:

- MQCC_OK
- MQCC_AVVERTENZA
- MQCC_NON RIUSCITO

Sintassi Per ottenere: *completioncode & = MQDistributionList*.**CompletionCode**

proprietà ConnectionReference

Lettura - scrittura. Il gestore code a cui appartiene l'elenco di distribuzione.

Definito in:
Classe MQDistributionList

Tipo di dati:
MQQueueManager

Sintassi: per ottenere *set queuemanager = MQDistributionList*.**ConnectionReference**

Per impostare: *impostare MQDistributionList*. **ConnectionReference** = *queuemanager*

proprietà FirstDistributionListItem

Sola lettura. Il primo oggetto dell'elenco di distribuzione associato all'elenco di distribuzione.

Definito in:
Classe MQDistributionList

Tipo di dati:
MQDistributionListElemento

Valori:

Sintassi: per ottenere *set distributionlistitem = MQDistributionList*.**FirstDistributionListItem**

proprietà IsOpen

Sola lettura.

Definito in:
Classe MQDistributionList

Tipo di dati:
Booleano

Valori:

- VERO (-1)
- FALSE (0)

Sintassi : *IsOpen = MQDistributionList*.**IsOpen**

proprietà OpenOptions

Lettura - scrittura. Opzioni da utilizzare quando viene aperto l'elenco di distribuzione.

Definito in:
Classe MQDistributionList

Tipo di dati:
Lungo

Valori:
Vedere Opzioni MQPMO.

Sintassi: Per ottenere: *openopt & = MQDistributionList*.**OpenOptions**

Per impostare: *MQDistributionList.***OpenOptions** = openopt &

proprietà ReasonCode

Sola lettura. Il codice di errore impostato dall'ultimo accesso metodo o proprietà emesso rispetto all'oggetto.

Definito in:
Classe MQDistributionList

Tipo di dati:
Lungo

Valori:
Vedere codici di errore API.

Sintassi: per ottenere *reasoncode & = MQDistributionList*.**ReasonCode**

proprietà ReasonName

Sola lettura. Il nome simbolico per ReasonCode. Ad esempio, "MQRC_QMGR_NOT_AVAILABLE".

Definito in:
Classe MQDistributionList

Tipo di dati:
Stringa

Valori:
Vedere codici di errore API.

Sintassi: per richiamare *reasonname $= MQDistributionList*.**ReasonName**

metodo AddDistributionListItem

Crea un nuovo oggetto elemento MQDistributionListe lo associa con l'oggetto elenco di distribuzione. Il parametro nome coda è obbligatorio.

La proprietà DistributionList dell'elenco di distribuzione è impostata sull'elenco di distribuzione proprietario e la proprietà FirstDistributionListItem dell'elenco di distribuzione è impostata per fare riferimento a questo nuovo elemento dell'elenco di distribuzione.

Per il nuovo elemento dell'elenco di distribuzione, la proprietà PreviousDistributionListItem non è impostata su nulla e la proprietà NextDistributionListItem è impostata per fare riferimento a qualsiasi elemento dell'elenco di distribuzione che era stato prima o a nulla se non era presente in precedenza (ossia, il nuovo elemento è inserito davanti a quelli già esistenti).

Ciò restituirà un errore se l'elenco di distribuzione è aperto.

Definito in:
Classe MQDistributionList

Sintassi: set distributionlistitem = *MQDistributionList***.AddDistributionListItem** (QName$, QMgrName$)

Parametri:

Stringa *QName$* . Nome della coda WebSphere MQ di .

QMgrName$ Stringa. Nome del gestore code WebSphere MQ .

Metodo di codici ClearError

Reimposta CompletionCode su MQCC_OK e ReasonCode su MQRC_NONE per la classe MQDistributionList e la classe MQSession.

Definito in:
Classe MQDistributionList

Sintassi: Call *MQDistributionList*.**ClearErrorCodes**()

Metodo di chiusura

Chiude un elenco di distribuzione utilizzando il valore corrente delle opzioni di chiusura.

Definito in:
Classe MQDistributionList

Sintassi: Call *MQDistributionList*.**Close**()

Apri metodo

Apre ciascuna delle code specificate dalle proprietà QueueName e QueueManagerName delle voci dell'elenco di distribuzione associate all'oggetto corrente utilizzando il valore corrente di AlternateUserId.

Definito in:
Classe MQDistributionList

Sintassi: Call *MQDistributionList*.**Open**()

Metodo PUT

Inserisce un messaggio in ciascuna delle code identificate dalle voci dell'elenco di distribuzione associate all'elenco di distribuzione.

Definito in:

Classe MQDistributionList

Sintassi

Chiamare MQDistributionList.**Put**(Message, PutMsgOptions &)

Parametri

Messaggio Oggetto MQMessage che rappresenta il messaggio da inserire.

PutMsgOpzioni MQPutMessageOggetto opzioni contenente le opzioni per controllare l'operazione di inserimento. Se non viene specificato, vengono utilizzate le opzioni PutMessagepredefinite.

Questo metodo utilizza un oggetto MQMessage come parametro. Le seguenti proprietà dell'elemento dell'elenco di distribuzione possono essere modificate come risultato di questo metodo:

- CompletionCode
- ReasonCode
- ReasonName
- MessageId
- MessageIdesadecimale
- CorrelationId

- CorrelationIdesadecimale
- GroupId
- GroupIdesadecimale
- Feedback
- AccountingToken
- AccountingTokenesadecimale

Classe di elementi MQDistributionList

Questa classe incapsula le strutture MQOR, MQRR e MQPMR e le associa a un elenco di distribuzione proprietario.

Creazione

Utilizzare il metodo AddDistributionListItem della classe MQDistributionList

Proprietà

Metodi

- "proprietà AccountingToken" a pagina 1103.
- "proprietà Hex AccountingToken" a pagina 1103.
- "proprietà CompletionCode" a pagina 1103.
- "proprietà CorrelationId" a pagina 1104.
- "proprietà Hex CorrelationId" a pagina 1104.
- "proprietà DistributionList" a pagina 1104.
- "proprietà Feedback" a pagina 1104.
- "proprietà GroupId" a pagina 1105.
- "proprietà Hex GroupId" a pagina 1105.
- "proprietà MessageId" a pagina 1105.
- "proprietà Hex MessageId" a pagina 1105.
- "proprietà NextDistributionListItem" a pagina 1106.
- "proprietà PreviousDistributionListItem" a pagina 1106.
- "proprietà Nome QueueManager" a pagina 1106.
- "proprietà QueueName" a pagina 1106.
- "proprietà ReasonCode" a pagina 1106.
- "proprietà ReasonName" a pagina 1107.
- "Metodo di codici ClearError" a pagina 1107.

Proprietà:

- proprietà AccountingToken
- proprietà Hex AccountingToken
- proprietà CompletionCode
- proprietà CorrelationId
- proprietà Hex CorrelationId
- proprietà DistributionList
- proprietà Feedback

- proprietà GroupId
- proprietà Hex GroupId
- proprietà MessageId
- proprietà Hex MessageId
- proprietà NextDistributionListItem
- proprietà PreviousDistributionListItem
- proprietà Nome QueueManager
- proprietà QueueName
- proprietà ReasonCode
- proprietà ReasonName

Metodi:

- Metodo di codici ClearError

Creazione:

Utilizzare il metodo AddDistributionListItem della classe MQDistributionList

proprietà AccountingToken

Lettura - scrittura. Il AccountingToken da includere nel MQPMR di un messaggio quando viene inserito in una coda. Il valore iniziale è tutti null.

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa di 32 caratteri

Sintassi: per ottenere: *accountingtoken $= MQDistributionListItem*.**AccountingToken**

Per impostare: *MQDistributionListItem*.**AccountingToken** = *accountingtoken $*

proprietà Hex AccountingToken

Lettura - scrittura. Il AccountingToken da includere nel MQPMR di un messaggio quando viene inserito in una coda.

Ogni due caratteri della stringa rappresentano l'equivalente esadecimali di un singolo carattere ASCII. Ad esempio, la coppia di caratteri "6" e "1" rappresentano il singolo carattere "A", la coppia di caratteri "6" e "2" rappresentano il singolo carattere "B" e così via.

È necessario fornire 64 caratteri esadecimali validi.

Il valore iniziale è "0 ... 0".

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa di 64 caratteri esadecimali che rappresentano 32 caratteri ASCII.

Sintassi: per ottenere: *accountingtokenh $= MQDistributionListItem*.**AccountingTokenHex**

Per impostare: *MQDistributionListItem*.**AccountingTokenHex** = *accountingtokenh $*

proprietà CompletionCode

Sola lettura. Il codice di completamento impostato dall'ultima richiesta di apertura o di inserimento emessa rispetto all'oggetto dell'elenco di distribuzione proprietario.

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Lungo

Valori:

- MQCC_OK
- MQCC_AVVERTENZA
- MQCC_NON RIUSCITO

Sintassi: Per ottenere: *completioncode $= MQDistributionListItem*.**CompletionCode**

proprietà CorrelationId

Lettura - scrittura. Il CorrelId da includere in MQPMR di un messaggio quando viene inserito su una coda. Il valore iniziale è tutti null.

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa di 24 caratteri

Sintassi: per ottenere: *correlid $= MQDistributionListItem*.**CorrelationId**

Per impostare *MQDistributionListItem*.**CorrelationId** = *correlid $*

proprietà Hex CorrelationId

Lettura - scrittura. Il CorrelId da includere in MQPMR di un messaggio quando viene inserito su una coda.

Ogni due caratteri della stringa rappresentano l'equivalente esadecimali di un singolo carattere ASCII. Ad esempio, la coppia di caratteri "6" e "1" rappresentano il singolo carattere "A", la coppia di caratteri "6" e "2" rappresentano il singolo carattere "B" e così via.

È necessario fornire 48 caratteri esadecimali validi.

Il valore iniziale è "0 .. 0".

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa di 48 caratteri esadecimali che rappresentano 24 caratteri ASCII.

Sintassi: per ottenere: *correlidh $= MQDistributionListItem*.**CorrelationIdHex**

Per impostare: *MQDistributionListItem*.**CorrelationIdHex** = *correlidh $*

proprietà DistributionList

Sola lettura. L'elenco di distribuzione a cui è associata questa voce dell'elenco di distribuzione.

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
MQDistributionList

Sintassi: Per ottenere: *set distributionlist = MQDistributionListItem***DistributionList**

proprietà Feedback

Lettura - scrittura. Il valore Feedback da includere nel MQPMR di un messaggio quando viene inserito in una coda.

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Lungo

Valori:
Vedi Feedback (MQLONG).

Sintassi Per ottenere: *feedback & = MQDistributionListItem*.**Feedback**

Per impostare: *MQDistributionListItem*.**Feedback** = *feedback &*

proprietà GroupId

Lettura - scrittura. L' GroupId da includere in MQPMR di un messaggio quando viene inserito su una coda. Il valore iniziale è tutti null.

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa di 24 caratteri

Sintassi: per ottenere: *groupid $= MQDistributionListItem*.**GroupId**

Per impostare: *MQDistributionListItem*.**GroupId** = *groupid $*

proprietà Hex GroupId

Lettura - scrittura. L' GroupId da includere in MQPMR di un messaggio quando viene inserito su una coda.

Ogni due caratteri della stringa rappresentano l'equivalente esadecimali di un singolo carattere ASCII. Ad esempio, la coppia di caratteri "6" e "1" rappresentano il singolo carattere "A", la coppia di caratteri "6" e "2" rappresentano il singolo carattere "B" e così via.

È necessario fornire 48 caratteri esadecimali validi.

Il valore iniziale è "0 .. 0".

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa di 48 caratteri esadecimali che rappresentano 24 caratteri ASCII.

Sintassi: per ottenere: *groupidh $= MQDistributionListItem*.**GroupIdHex**

Per impostare: *MQDistributionListItem*.**GroupIdHex** = *groupidh $*

proprietà MessageId

Lettura - scrittura. Il MessageId da includere in MQPMR di un messaggio quando viene inserito su una coda. Il valore iniziale è tutti null.

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa di 24 caratteri

Sintassi: per ottenere *messageid $= MQDistributionListItem*.**MessageId**

Per impostare: *MQDistributionListItem*.**MessageId** = *message id $*

proprietà Hex MessageId

Lettura - scrittura. Il MessageId da includere in MQPMR di un messaggio quando viene inserito su una coda.

Ogni due caratteri della stringa rappresentano l'equivalente esadecimali di un singolo carattere ASCII. Ad esempio, la coppia di caratteri "6" e "1" rappresentano il singolo carattere "A", la coppia di caratteri "6" e "2" rappresentano il singolo carattere "B" e così via.

È necessario fornire 48 caratteri esadecimali validi.

Il valore iniziale è "0 .. 0".

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa di 48 caratteri esadecimali che rappresentano 24 caratteri ASCII.

Sintassi: per ottenere: *messageidh $= MQDistributionListItem*.**MessageIdHex**

Per impostare: *MQDistributionListItem*.**MessageIdHex** = *messageidh $*

proprietà NextDistributionListItem

Sola lettura. Il successivo oggetto elemento dell'elenco di distribuzione associato allo stesso elenco di distribuzione.

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
MQDistributionListElemento

Sintassi: per ottenere *set distributionlistitem = MQDistributionListItem.***NextDistributionListItem**

proprietà PreviousDistributionListItem

Sola lettura. L'oggetto elemento dell'elenco di distribuzione precedente associato allo stesso elenco di distribuzione.

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
MQDistributionListElemento

Sintassi: per ottenere *set distributionlistitem = MQDistributionListItem*.**PreviousDistributionListItem**

proprietà Nome QueueManager

Lettura - scrittura. Il nome del gestore code WebSphere MQ .

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa di 48 caratteri.

Sintassi: per richiamare: *qmname $= MQDistributionListItem*.**QueueManagerName**

Per impostare: *MQDistributionListItem*.**QueueManagerName** = *qmname $*

proprietà QueueName

Lettura - scrittura. Il nome della coda WebSphere MQ .

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa di 48 caratteri.

Sintassi: per ottenere: *qname $= MQDistributionListItem*.**QueueName**

Per impostare: *MQDistributionListItem*.**QueueName** = *qname $*

proprietà ReasonCode

Sola lettura. Il codice di completamento impostato dall'ultima apertura o inserimento emesso nell'oggetto dell'elenco di distribuzione proprietario.

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Lungo

Valori:
Vedere codici di errore API.

- MQCC_OK
- MQCC_AVVERTENZA
- MQCC_NON RIUSCITO

Sintassi: per ottenere: *reasoncode & = MQDistributionListItem*.**ReasonCode**

proprietà ReasonName

Sola lettura. Il nome simbolico per ReasonCode. Ad esempio, "MQRC_QMGR_NOT_AVAILABLE".

Definito in:
Classe di elementi MQDistributionList

Tipo di dati:
Stringa

Valori:
Vedere codici di errore API.

Sintassi: per ottenere: *reasonname $= MQDistributionListItem*.**ReasonName**

Metodo di codici ClearError

Reimposta il CompletionCode su MQCC_OK e il ReasonCode su MQRC_NONE sia per la classe di elementi MQDistributionListche per la classe MQSession.

Definito in:
Classe di elementi MQDistributionList

Sintassi: Call *MQDistributionListItem*.**ClearErrorCodes**

Risoluzione dei problemi

Informazioni sulla funzione di traccia fornita, insidie comuni e aiuto su come evitarle.

Questa sezione spiega la funzione di traccia fornita e i dettagli delle insidie comuni, con un aiuto per evitarle:

Utilizzo della traccia

MQAX include una funzione di traccia che consente all'organizzazione del servizio di identificare cosa accade quando si verifica un problema. Mostra i percorsi intrapresi quando si esegue lo script MQAX. A meno che non si abbia un problema, eseguire l'esecuzione con la traccia disattivata per evitare un utilizzo non necessario delle risorse di sistema.

Esistono tre variabili di ambiente impostate per controllare la traccia:

- TRACCIA MQ_O

- PERCORSO_TRACCIA_OMQ
- LIVELLO_TRACCIA_OMQ_

Notare che specificando *qualsiasi* valore per OMQ_TRACE si attiva la funzione di traccia. Anche se si imposta OMQ_TRACE su OFF, la traccia è ancora attiva.

Per disattivare la traccia, non specificare un valore per OMQ_TRACE.

1. Fare clic su **Avvia**
2. Fare clic su **Pannello di controllo**
3. Fare doppio clic su **Sistema**
4. Fare clic su **Avanzate**
5. Fare clic su **Ambiente**
6. Nella sezione intitolata "Variabili utente per (nome utente)", fare clic su **Nuovo**
7. Immettere il nome della variabile e un valore valido nei campi appropriati e fare clic su **OK**
8. Fare clic su **OK** per chiudere la finestra Variabile di ambiente
9. Fare clic su **OK** per chiudere la finestra Proprietà del sistema
10. Chiude la finestra Pannello di controllo

Quando si decide dove si desidera che vengano scritti i file di traccia, assicurarsi di disporre dell'autorizzazione sufficiente per scrivere, non solo leggere dal disco.

Con la traccia attivata, rallenta l'esecuzione di MQAX, ma non influisce sulle prestazioni degli ambienti ActiveX o WebSphere MQ . Quando non è più necessario un file di traccia, è possibile eliminarlo.

È necessario arrestare l'esecuzione di MQAX per modificare lo stato della variabile OMQ_TRACE.

Nome file di traccia e directory

Il nome del file di traccia ha il formato OMQnnnnn.trc, dove nnnnn è l'id del processo di ActiveX in esecuzione al momento.

Tabella 157. Comandi e relativi effetti

Comando	Effetto
SET OMQ_TRACE_PATH = unità: \directory	Imposta la directory di traccia in cui verrà scritto il file di traccia.
IMPOSTA PERCORSO TRACCIA OMQ =	Rimuove la variabile di ambiente OMQ_PATH dalla directory di lavoro corrente (quando viene avviato ActiveX).
ECHO %OMQ_TRACE_PATH%	Visualizza l'impostazione corrente della directory di traccia su Windows.
SET TRACCIA OMQ = xxxxxxxx	Ciò imposta la traccia su ON. Si attiva la traccia inserendo uno o più caratteri dopo il segno '='. Ad esempio: SET OMQ_TRACE=yes SET OMQ_TRACE = no. In entrambi questi esempi, la traccia verrà impostata su ON. Questo è valido solo per una singola finestra / sessione
SET OMQ_TRACE=	Imposta la traccia su OFF
%OMQ_TRACE% ECHO	Visualizza il contenuto della variabile di ambiente in Windows.
SET	Visualizza il contenuto di tutte le variabili di ambiente in Windows.
IMPOSTAZIONE LIVELLO TRACCIA OMQ = 9	Imposta il livello di traccia su 9. I valori maggiori di 9 non producono ulteriori informazioni nel file di traccia.

Quando lo script WebSphere MQ Automation Classes for ActiveX non riesce

Se lo script WebSphere MQ Automation Classes for ActiveX ha esito negativo, esistono diverse fonti di informazioni.

Report dei sintomi del primo errore

Indipendentemente dalla funzione di traccia, per gli errori interni e imprevisti, potrebbe essere prodotto un report dei sintomi del primo errore.

Questo report si trova in un file denominato OMQnnnnn.fdc, dove nnnnn è il numero del processo ActiveX in esecuzione al momento. Questo file si trova nella directory di lavoro da cui è stato avviato ActiveX o nel percorso specificato nella variabile di ambiente OMQ_PATH.

Altre fonti di informazione

WebSphere MQ fornisce vari log di errori e informazioni di traccia, a seconda delle piattaforme coinvolte. Consultare il log eventi dell'applicazione Windows NT.

Codici di origine

I seguenti codici di errore possono verificarsi in aggiunta a quelli documentati per WebSphere MQ MQI. Per altri codici, fare riferimento alla registrazione eventi dell'applicazione WebSphere MQ .

Tabella 158. Codici di errore e loro significato

Codice di errore	Spiegazione
MQRC_LIBRARY_LOAD_ERROR (6000)	Non è stato possibile caricare una o più librerie WebSphere MQ . Verificare che tutte le librerie WebSphere MQ siano nel percorso di ricerca corretto sul sistema che si sta utilizzando. Ad esempio, assicurarsi che le directory contenenti le librerie WebSphere MQ siano in PATH.
ERRORE MQRC_CLASS_LIBRARY_ERROR (6001)	Una delle chiamate della libreria di classi WebSphere MQ ha restituito un ReasonCode/CompletionCodenon previsto. Per i dettagli, consultare il report sul sintomo del primo errore. Prendere nota dell'ultimo metodo / proprietà e della classe utilizzati e informare il supporto IBM del problema.
MQRC_STRING_LENGTH_TOO_BIG (6002)	È stato effettuato un tentativo di scrivere una stringa di formato UTF con una lunghezza superiore a 65.535 byte nel buffer dei messaggi.
ERRORE MQRC_WRITE_VALUE_(6003)	Viene utilizzato un valore non compreso nell'intervallo, ad esempio msg.WriteByte (240).
MQRC_PACKED_DECIMAL_ERROR (6004)	Si è tentato di leggere un numero decimale compresso dal buffer dei messaggi, ma i dati nel puntatore dati non sono in un formato dati compresso valido.
ERRORE MQRC_FLOAT_CONVERSION_ERROR (6005)	È stato effettuato un tentativo di leggere un numero a virgola mobile singolo o doppio dal buffer dei messaggi, ma i dati nel puntatore dati non sono in un formato a virgola mobile appropriato.
MQRC_REOPEN_EXCL_INPUT_ERROR (6100)	Un oggetto aperto non dispone delle **OpenOptions** corrette e richiede una o più opzioni aggiuntive. È richiesta una riapertura implicita, ma è stata impedita la chiusura. Impostare esplicitamente **OpenOptions**per coprire tutte le eventualità in modo che non sia richiesta la riapertura implicita. La chiusura è stata impedita perché la coda è aperta per l'input esclusivo e la chiusura rappresenterebbe una finestra di opportunità per altri potenzialmente per ottenere l'accesso alla coda.
MQRC_REOPEN_INQUIRE_ERROR (6101)	Un oggetto aperto non dispone delle OpenOptions corrette e richiede una o più opzioni aggiuntive. È richiesta una riapertura implicita, ma è stata impedita la chiusura. Impostare esplicitamente OpenOptions per includere MQOO_INQUIRE. La chiusura è stata impedita perché una o più caratteristiche dell'oggetto devono essere controllate dinamicamente prima della chiusura e le OpenOptions non includono già MQOO_INQUIRE.

Tabella 158. Codici di errore e loro significato (Continua)

Codice di errore	Spiegazione
MQRC_REOPEN_SAVED_CONTEXT_ERR (6102)	Un oggetto aperto non dispone delle OpenOptions corrette e richiede una o più opzioni aggiuntive. È richiesta una riapertura implicita, ma è stata impedita la chiusura. Impostare esplicitamente OpenOptions per coprire tutte le eventualità in modo che non sia richiesta la riapertura implicita. La chiusura è stata impedita perché la coda è aperta con MQOO_SAVE_ALL_CONTEXT e una ricezione distruttiva è stata eseguita in precedenza. Ciò ha causato l'associazione delle informazioni di stato conservate con la coda aperta e tali informazioni verrebbero eliminate dalla chiusura.
MQRC_REOPEN_TEMPORARY_Q_ERROR (6103)	Un oggetto aperto non dispone delle **OpenOptions** corrette e richiede una o più opzioni aggiuntive. È richiesta una riapertura implicita, ma è stata impedita la chiusura. Impostare esplicitamente **OpenOptions**per coprire tutte le eventualità in modo che non sia richiesta la riapertura implicita. La chiusura è stata impedita poiché la coda è una coda locale del tipo di definizione MQQDT_TEMPORARY_DYNAMIC, che verrebbe distrutta dalla chiusura.
MQRC_ATTRIBUTE_LOCKED (6104)	Si è tentato di modificare il valore o l'attributo di un oggetto mentre l'oggetto è aperto. Alcuni attributi, come **AlternateUserId**, non possono essere modificati mentre un oggetto è aperto.
MQRC_CURSOR_NOT_VALID (6105)	Il cursore di ricerca per una coda aperta è stato invalidato dall'ultimo utilizzo da parte di una riapertura implicita. Impostare esplicitamente OpenOptions per coprire tutte le eventualità in modo che non sia richiesta la riapertura implicita.
MQRC_ENCODING_ERROR (6106)	La codifica del successivo elemento del messaggio deve essere MQENC_NATIVE per la lettura.
ERRORE DI MQRC_STRUCID_(6107)	La struttura dell'ID per l'elemento del messaggio successivo, derivato dai 4 caratteri che iniziano con il puntatore dati, manca o è incongruente con il tipo di variabile in cui l'elemento viene letto.
MQRC_NULL_POINTER (6108)	È stato fornito un puntatore null dove è richiesto o implicito un puntatore non null. Ciò può essere causato dall'utilizzo di dichiarazioni esplicite per oggetti WebSphere MQ utilizzati da VBA come parametri per le chiamate (ad esempio dim msg as Object is ok, dim msg as MqMessage può causare problemi). Ad esempio, in Excel, con q definito e impostare dim msg come MqMessageq.put msg fornisce reasonCode MQRC_NULL_POINTER. Funziona correttamente da VisualBasic.
MQRC_NO_CONNECTION_REFERENCE (6109)	L'oggetto **MQQueue** ha perso la connessione a **MQQueueManager**. Ciò si verifica se **MQQueueManager**è disconnesso. Eliminare l'oggetto **MQQueue** .
MQRC_NO_BUFFER (6110)	Nessun buffer disponibile. Per un oggetto **MQMessage** , non è possibile assegnarne uno, ad indicare un'incongruenza interna nello stato dell'oggetto che non dovrebbe verificarsi.
MQRC_BINARY_DATA_LENGTH_ERROR (6111)	La lunghezza dei dati binari non è congruente con la lunghezza dell'attributo di destinazione. Zero è una lunghezza corretta per tutti gli attributi. 24 è una lunghezza corretta per un **CorrelationId** e per **MessageId** 32 è una lunghezza corretta per un **AccountingToken**
MQRC_BUFFER_NOT_AUTOMATIC (6112)	Un buffer gestito e definito dall'utente non può essere ridimensionato. Poiché i buffer dei messaggi sono gestiti dal sistema, ciò indica un'incoerenza interna.
MQRC_INSUFFICIENT_BUFFER (6113)	Lo spazio di buffer disponibile non è sufficiente dopo il puntatore dati per soddisfare la richiesta. Ciò potrebbe essere dovuto al fatto che il buffer non può essere ridimensionato.
MQRC_INSUFFICIENT_DATA (6114)	Dati insufficienti dopo il puntatore dati per soddisfare la richiesta di lettura. Ridurre il buffer alla dimensione corretta e leggere nuovamente i dati.

Tabella 158. Codici di errore e loro significato (Continua)

Codice di errore	Spiegazione
MQRC_DATA_TRUNCATED (6115)	I dati sono stati troncati durante la copia da un buffer ad un altro. Ciò potrebbe essere dovuto al fatto che il buffer di destinazione non può essere ridimensionato o perché si è verificato un problema con l'uno o l'altro buffer o perché un buffer viene ridimensionato con una sostituzione più piccola.
MQRC_ZERO_LENGTH (6116)	È stata fornita una lunghezza zero dove è richiesta o implicita una lunghezza positiva.
MQRC_NEGATIVE_LENGTH (6117)	È stata fornita una lunghezza negativa dove è richiesta una lunghezza zero o positiva.
MQRC_NEGATIVE_OFFSET (6118)	È stato fornito un offset negativo dove è richiesto un offset zero o positivo.
MQRC_INCONSISTENT_FORMAT (6119)	Il formato dell'elemento del messaggio successivo non è coerente con il tipo di variabile in cui viene letto l'elemento.
MQRC_INCONSISTENT_OBJECT_STATE (6120)	Esiste un'incongruenza tra questo oggetto, che è aperto, e l'oggetto MQQueueManager di riferimento, che non è connesso.
MQRC_CONTEXT_OBJECT_NOT_VALID (6121)	Il riferimento al contesto delle opzioni MQPutMessagenon fa riferimento a un oggetto MQQueue valido. L'oggetto è stato precedentemente distrutto.
MQRC_CONTEXT_OPEN_ERROR (6122)	Il riferimento al contesto delle opzioni MQPutMessagefa riferimento a un oggetto MQQueue che non è stato possibile aprire per stabilire un contesto. Ciò potrebbe essere dovuto al fatto che l'oggetto MQQueue ha opzioni di apertura non appropriate. Esaminare il codice motivo dell'oggetto di riferimento per stabilire la causa.
MQRC_STRUC_LENGTH_ERROR (6123)	La lunghezza di una struttura dati interna non è congruente con il contenuto. Per un MQRMH, la lunghezza non è sufficiente per contenere i campi fissi e tutti i dati di offset.
MQRC_NOT_CONNECTED (6124)	Un metodo non è riuscito perché una connessione richiesta a un gestore code non era disponibile e una connessione non può essere stabilita implicitamente.
MQRC_NOT_OPEN (6125)	Un metodo non è riuscito perché un oggetto WebSphere MQ non era aperto e l'apertura non può essere eseguita implicitamente.
MQRC_DISTRIBUTION_LIST_EMPTY (6126)	Non è stato possibile aprire un MQDistributionList poiché non vi sono oggetti elemento MQDistributionList nell'elenco di distribuzione. Azione correttiva: aggiungere almeno un oggetto MQDistributionListItem all'elenco di distribuzione.
MQRC_INCONSISTENT_OPEN_OPTIONS (6127)	Un metodo non è riuscito perché l'oggetto è aperto e le opzioni di apertura non sono congruenti con l'operazione richiesta. Azione correttiva: aprire l'oggetto con le opzioni di apertura appropriate e riprovare.
VERSIONE MQRC_WRONG(6128)	Un metodo non è riuscito perché un numero di versione specificato o rilevato non è corretto o non è supportato.

Strumento a livello di codice

IBM Service Team potrebbe richiedere il livello di codice installato.

Per informazioni, eseguire il programma di utilità 'MQAXLEV'.

Dal prompt dei comandi, passare alla directory contenente MQAX200.dll oppure aggiungere la lunghezza del percorso completo e immettere:

```
MQAXLev MQAX200.dll > MQAXLEV.OUT
```

dove MQAXLEV.OUT è il nome del file di output.

Se non si specifica un file di emissione, i dettagli vengono visualizzati sullo schermo.

Un file di output di esempio dallo strumento a livello di codice è descritto nel seguente esempio:

Esempio di file di output dallo strumento a livello di codice

```
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
```

Interfaccia ActiveX per MQAI

Per una breve panoramica delle interfacce COM e del loro uso in MQAI, consultare “Utilizzo di Component Object Model Interface (WebSphere MQ Automation Classes for ActiveX)” a pagina 1035.

MQAI consente alle applicazioni di creare e inviare comandi PCF (Programmable Command Format) senza ottenere e formattare direttamente i buffer a lunghezza variabile richiesti per PCF. Per ulteriori informazioni su MQAI, consultare Introduzione a WebSphere MQ Administration Interface (MQAI). La classe MQBag ActiveX MQAI incapsula i bag di dati supportati da MQAI in un modo che è possibile utilizzare in qualsiasi linguaggio che supporti la creazione di oggetti COM; ad esempio, Visual Basic, C + +, Java e altri client di script ActiveX .

L'interfaccia MQAI ActiveX deve essere utilizzata con le classi MQAX che forniscono un'interfaccia COM a MQI. Per ulteriori informazioni sulle classi MQAX, consultare “Progettazione di applicazioni MQAX che accedono ad applicazioni nonActiveX” a pagina 1036.

L'interfaccia ActiveX fornisce una singola classe denominata MQBag. Questa classe è utilizzata per creare i bag di dati MQAI e le relative proprietà e metodi vengono utilizzati per creare e gestire gli elementi dati all'interno di ciascun bag. Il metodo MQBag Execute invia i dati bag a un gestore code WebSphere MQ come messaggio PCF e raccoglie le risposte.

Per ulteriori informazioni sulla classe MQBag, le relative proprietà e metodi, consultare “La classe MQBag” a pagina 1112.

Il messaggio PCF viene inviato all'oggetto del gestore code specificato, utilizzando facoltativamente le code di richiesta e di risposta specificate. Le risposte vengono restituite in un nuovo oggetto MQBag. La serie completa di comandi e risposte è descritta in Definizioni dei formati di comando programmabili. I comandi possono essere inviati a qualsiasi gestore code nella rete WebSphere MQ selezionando le code di richiesta e risposta appropriate.

La classe MQBag

La classe, MQBag, viene utilizzata per creare oggetti MQBag come richiesto. Quando viene creata un'istanziazione, la classe MQBag restituisce un nuovo riferimento oggetto MQBag.

Creare un oggetto MQBag in Visual Basic come segue:

```
Dim mqbag As MQBag
Set mqbag = New MQBag
```

Proprietà MQBag

Le proprietà degli oggetti MQBag sono illustrate nel seguente elenco:

- “Proprietà elemento” a pagina 1113.
- “proprietà Conteggio” a pagina 1114.
- “proprietà Opzioni” a pagina 1115.

Metodi MQBag

I metodi degli oggetti MQBag vengono spiegati nel seguente elenco:

- “Aggiungi metodo” a pagina 1115.
- “metodo AddInquiry” a pagina 1116.
- “Metodo di cancellazione” a pagina 1116.
- “Metodo di esecuzione” a pagina 1117.
- “metodo FromMessage” a pagina 1117.
- “Metodo ItemType” a pagina 1118.
- “Rimuovi metodo” a pagina 1118.
- “Metodo selettore” a pagina 1119.
- “metodo ToMessage” a pagina 1120.
- “Metodo di troncamento” a pagina 1120.

Gestione errori

Se viene rilevato un errore durante un'operazione su un oggetto MQBag, inclusi gli errori restituiti al contenitore da un oggetto MQAX o MQAI sottostante, viene generata un'errore. La classe MQBag supporta l'interfaccia COM ISupportErrorInfo in modo che le seguenti informazioni siano disponibili per la routine di gestione degli errori:

- Numero di errore: composto dal codice di errore WebSphere MQ per l'errore rilevato e da un codice funzione COM. Il campo funzione, come standard per COM, indica l'area di responsabilità per l'errore. Per gli errori rilevati da WebSphere MQ è sempre FACILITY_ITF.
- Origine errore: identifica il tipo e la versione dell'oggetto che ha rilevato l'errore. Per gli errori rilevati durante le operazioni MQBag, l'origine errore è sempre MQBag.MQBag1.
- Descrizione dell'errore: la stringa che fornisce il nome simbolico per il codice motivo WebSphere MQ .

Il modo in cui si accede alle informazioni di errore dipende dal linguaggio di script; ad esempio, in Visual Basic le informazioni vengono restituite nell'oggetto Err e il codice di errore WebSphere MQ si ottiene sottraendo la costante vbObjectError da Err.Number.

ReasonCode = Err.Number - Errore vbObject

Se il messaggio MQBag Execute invia un messaggio PCF e viene ricevuta una risposta, l'operazione viene considerata riuscita anche se il comando inviato potrebbe non essere riuscito. In questo caso, il contenitore di risposta stesso contiene i codici di errore e di completamento come descritto in Definizioni dei formati dei comandi programmabili .

Proprietà elemento

Finalità

La proprietà Item rappresenta un item in un bag. Viene utilizzato per impostare o interrogare il valore di un elemento. L'utilizzo di questa proprietà corrisponde alle seguenti chiamate MQAI:

- "StringamqSet"
- "mqSetNumero intero"
- "Numero interomqInquire"
- "StringamqInquire"

- "BorsamqInquire"

in Riferimento formati comando programmabile.

Formato

Elemento (Selettore, ItemIndex, Valore)

Parametri

***Selector* (VARIANT) - immissione**
Selettore dell'elemento da impostare o interrogare.

Quando si interroga un elemento, MQSEL_ANY_USER_SELECTOR è il valore predefinito. Quando si imposta un elemento, MQIA_LIST o MQCA_LIST è il valore predefinito.

Se `Selector` non è di tipo long, ne risulta MQRC_SELECTOR_TYPE_ERROR.

Questo parametro è facoltativo.

***ItemIndex* (LONG) - input**
Questo valore identifica la ricorrenza dell'elemento del selettore specificato che deve essere impostato o interrogato. MQIND_NONE è il valore predefinito.

Questo parametro è facoltativo.

***Value* (VARIANT) - input/output**
Il valore restituito o il valore da impostare. Quando si interroga un elemento, il valore di ritorno può essere di tipo long, string o MQBag. Tuttavia, quando si imposta un elemento, il valore deve essere di tipo long o string; in caso contrario, risulta MQRC_ITEM_VALUE_ERROR.

Richiamo linguaggio Visual Basic

Quando si richiede un valore di un articolo all'interno di una borsa:

```
Value = mqbag[.Item]([Selector],
[ItemIndex])
```

Per i riferimenti MQBag:

```
Set abag = mqbag[.Item]([Selector].
[ItemIndex])
```

Per impostare il valore di un elemento in un contenitore:

```
mqbag[.Item]([Selector],
[ItemIndex]) = Value
```

proprietà Conteggio

Finalità

La proprietà Conteggio rappresenta il numero di elementi dati all'interno di un contenitore.
Questa proprietà corrisponde alla chiamata MQAI, "mqCountItems," in Riferimento formati comando programmabile.

Formato

Conteggio (*Selector, Value*)

Parametri

***Selector* (VARIANT) - immissione**
Selettore degli elementi dati da inserire nel conteggio.

MQSEL_ALL_USER_SELECTORS è il valore predefinito.

Se `Selector` non è di tipo long, viene restituito MQRC_SELECTOR_TYPE_ERROR.

***Value* (LONG) - output**
Il numero di elementi nel contenitore inclusi da *`Selector`*.

Richiamo linguaggio Visual Basic

Per restituire il numero di articoli in un sacchetto:

```
ItemCount = mqbag.Count([Selector])
```

proprietà Opzioni

Finalità

La proprietà Opzioni imposta le opzioni per l'utilizzo di una borsa. Questa proprietà corrisponde al parametro `Options` della chiamata MQAI, "mqCreateBag," in Riferimento formati comando programmabile.

Formato

Opzioni (*`Options`*)

Parametri

***Options* (LONG) - input/output**
Le opzioni della borsa.

Nota: Le opzioni del contenitore devono essere impostate ***prima che gli elementi di dati*** vengano aggiunti o impostati all'interno del contenitore. Se le opzioni vengono modificate quando il contenitore non è vuoto, ne risulta MQRC_OPTIONS_ERROR. Ciò vale anche se il sacchetto viene successivamente sdoganato.

Richiamo linguaggio Visual Basic

Quando si interroga sulle opzioni di un articolo all'interno di una borsa:

```
Options = mqbag.Options
```

Per impostare un'opzione di un articolo in un contenitore:

```
mqbag.Options = Options
```

Metodi MQBag

I metodi degli oggetti MQBag sono descritti nelle pagine seguenti.

Aggiungi metodo

Finalità

Il metodo Add aggiunge un elemento dati ad un contenitore. Questo metodo corrisponde alle chiamate MQAI, "mqAddInteger" e "mqAddString," in Riferimento formati comando programmabile.

Formato

Aggiungi (*Value, Selector*)

Parametri

***Value* (VARIANT) - immissione**
Valore intero o stringa dell'elemento dati.

***Selector* (VARIANT) - immissione**
Selettore che identifica l'elemento da aggiungere.

In base al tipo di `Value`, MQIA_LIST o MQCA_LIST è il valore predefinito. Se il parametro `Selector` non è di tipo long, viene visualizzato MQRC_SELECTOR_TYPE_ERROR.

Richiamo linguaggio Visual Basic

Per aggiungere un articolo a una borsa:

```
mqbag.Add(Value,[Selector])
```

metodo AddInquiry

Finalità

Il metodo AddInquiry aggiunge un selettore specificando l'attributo da restituire quando viene inviato un contenitore di gestione per l'esecuzione di un comando INQUIRE. Questo metodo corrisponde alla chiamata MQAI, "mqAddInquiry,", in Riferimento formati comando programmabile.

Formato

AddInquiry *(Inquiry*)

Parametri

***Inquiry* (LONG) - input**
Selettore dell'attributo WebSphere MQ che deve essere restituito dal comando di gestione INQUIRE.

Richiamo linguaggio Visual Basic

Per utilizzare il metodo AddInquiry :

```
mqbag.AddInquiry(Inquiry)
```

Metodo di cancellazione

Finalità

Il metodo Clear elimina tutti gli elementi dati da un contenitore. Questo metodo corrisponde alla chiamata MQAI, "mqClearBag,", in Riferimento formati comando programmabile.

Formato

Azzera

Richiamo linguaggio Visual Basic

Per eliminare tutti gli elementi di dati da un contenitore:

```
mqbag.Clear
```

Metodo di esecuzione

Finalità

Il metodo Execute invia un messaggio di comando di gestione al server dei comandi e attende eventuali messaggi di risposta. Questo metodo corrisponde alla chiamata MQAI, "mqExecute,", in Riferimento formati comando programmabile.

Formato

Eseguire *(QueueManager, Command, OptionsBag, RequestQ, ReplyQ, ReplyBag*)

Parametri

QueueManager **(MQQueueManager) - input**
Il gestore code a cui è connessa l'applicazione.

Command **(LONG) - input**
Il comando da eseguire.

OptionsBag **(MQBag) - input**
Il contenitore contenente le opzioni che influiscono sull'elaborazione della chiamata.

RequestQ **(MQQueue) - input**
La coda in cui verrà inserito il messaggio del comando di gestione.

ReplyQ **(MQQueue) - input**
La coda in cui vengono ricevuti i messaggi di risposta.

ReplyBag **(MQBag) - output**
Un riferimento bag contenente i dati dei messaggi di risposta.

Richiamo linguaggio Visual Basic

Per inviare un messaggio di comando di gestione e attendere eventuali messaggi di risposta:

```
Set ReplyBag = mqbag.Execute(QueueManager, Command,
[OptionsBag],[RequestQ],[ReplyQ])
```

metodo FromMessage

Finalità

Il metodo FromMessage carica i dati da un messaggio in un contenitore. Questo metodo corrisponde alla chiamata MQAI, "mqBufferToBag," in Riferimento formati comando programmabile.

Formato

FromMessage *(Message, OptionsBag*)

Parametri

***Message* (MQMessage) - input**
Il messaggio contenente i dati da convertire.

***OptionsBag* (MQBag) - input**
Opzioni per controllare l'elaborazione della chiamata.

Richiamo linguaggio Visual Basic

Per caricare i dati da un messaggio in un contenitore:

```
mqbag.FromMessage(Message,[OptionsBag])
```

Metodo ItemType

Finalità

Il metodo ItemType restituisce il tipo del valore in un elemento specificato in un contenitore.
Questo metodo corrisponde alla chiamata MQAI, "mqInquireItemInfo," in Riferimento formati comando programmabile.

Formato

ItemType (*Selector, ItemIndex, ItemType*)

Parametri

***Selector* (VARIANT) - immissione**
Selettore che identifica l'elemento da interrogare.

MQSEL_ANY_USER_SELECTOR è il valore predefinito. Se il parametro `Selector` non è di tipo long, viene visualizzato MQRC_SELECTOR_TYPE_ERROR.

***ItemIndex* (LONG) - input**
Indice delle voci da interrogare.

MQIND_NONE è il valore predefinito.

***ItemType* (LONG) - output**
Tipo di dati dell'elemento specificato.

Nota: È necessario specificare il parametro `Selector` , il parametro `ItemIndex` o entrambi. Se non è presente alcun parametro, MQRC_PARAMETER_MISSING risulta.

Richiamo linguaggio Visual Basic

Per restituire il tipo di un valore:

```
ItemType = mqbag.ItemType([Selector],
[ItemIndex])
```

Rimuovi metodo

Finalità

Il metodo Rimuovi elimina un elemento da un contenitore. Questo metodo corrisponde alla chiamata MQAI, "mqDeleteItem," in Riferimento formati comando programmabile.

Formato

Rimuovere *(Selector, ItemIndex)*

Parametri

***Selector* (VARIANT) - immissione**
Selettore che identifica l'elemento da eliminare.

MQSEL_ANY_USER_SELECTOR è il valore predefinito. Se il parametro `Selector` non è di tipo long, viene visualizzato MQRC_SELECTOR_TYPE_ERROR.

***ItemIndex* (LONG) - input**
Indice dell'elemento da eliminare.

MQIND_NONE è il valore predefinito.

Nota: È necessario specificare il parametro `Selector` , il parametro `ItemIndex` o entrambi. Se non è presente alcun parametro, MQRC_PARAMETER_MISSING risulta.

Richiamo linguaggio Visual Basic

Per eliminare un elemento da un contenitore:

```
mqbag.Remove([Selector],[ItemIndex])
```

Metodo selettore

Finalità

Il metodo Selector restituisce il selettore di un elemento specificato all'interno di un contenitore. Questo metodo corrisponde alla chiamata MQAI, "mqInquireItemInfo," in Riferimento formati comando programmabile.

Formato

Selettore *(Selector, ItemIndex, OutSelector)*

Parametri

***Selector* (VARIANT) - immissione**
Selettore che identifica l'elemento da interrogare.

MQSEL_ANY_USER_SELECTOR è il valore predefinito. Se il parametro `Selector` non è di tipo long, viene visualizzato MQRC_SELECTOR_TYPE_ERROR.

***ItemIndex* (LONG) - input**
Indice dell'elemento da interrogare.

MQIND_NONE è il valore predefinito.

***OutSelector* (VARIANT) - emissione**
Selettore dell'elemento specificato.

Nota: È necessario specificare il parametro `Selector` , il parametro `ItemIndex` o entrambi. Se non è presente alcun parametro, MQRC_PARAMETER_MISSING risulta.

Richiamo linguaggio Visual Basic

Per restituire il selettore di un item:

```
OutSelector = mqbag.Selector([Selector],
[ItemIndex])
```

metodo ToMessage

Finalità

Il metodo ToMessage restituisce un riferimento ad un oggetto MQMessage. Il riferimento contiene dati da un contenitore. Questo metodo corrisponde alla chiamata MQAI, "mqBagToBuffer," in Riferimento formati comando programmabile.

Formato

ToMessage *(OptionsBag, Message)*

Parametri

OptionsBag **(MQBag) - input**
Un contenitore contenente opzioni che controllano l'elaborazione del metodo.

Message **(MQMessage) - output**
Un riferimento oggetto MQMessage contenente i dati dal contenitore.

Richiamo linguaggio Visual Basic

Per utilizzare il metodo ToMessage :

```
Set Message = mqbag.ToMessage([OptionsBag])
```

Metodo di troncamento

Finalità

Il metodo Tronca riduce il numero di elementi utente in un contenitore. Questo metodo corrisponde alla chiamata MQAI, "mqTruncateBag," in Riferimento formati comando programmabile.

Formato

Tronca ***(ItemCount)***

Parametri

ItemCount **(LONG) - input**
Il numero di elementi utente che devono rimanere nel contenitore dopo il troncamento.

Richiamo linguaggio Visual Basic

Per ridurre il numero di elementi utente in un contenitore:

```
mqbag.Truncate(ItemCount)
```

Informazioni sugli esempi di WebSphere MQ Automation Classes for ActiveX Starter

Questa appendice descrive gli esempi WebSphere MQ Automation Classes for ActiveX Starter e spiega come utilizzarli.

WebSphere MQ per Windows fornisce i seguenti programmi di esempio Visual Basic:

- MQAXTRIV.VBP
- MQAXBSRV.VBP
- MQAXDLST.VBP
- MQAXCLSS.VBP

Questi esempi vengono eseguiti su Visual Basic 4 o Visual Basic 5. Si troveranno nella directory ... \tools\mqax\samples\vb.

Nella stessa directory si trovano anche gli esempi per Microsoft Excel e html. Sono:

- MQAX.XLS
- MQAXTRIV.XLS
- MQAXTRIV.HTM

Nota: Se si utilizza Visual Basic 5, è **necessario** selezionare e installare il componente Visual Basic grid32.ocx.

Cosa viene dimostrato nei campioni

Gli esempi illustrano come utilizzare WebSphere MQ Automation Classes for ActiveX per:

- Connetterti a un gestore code
- Accesso a una coda
- Inserire un messaggio su una coda
- Richiamare un messaggio da una coda

La parte centrale dell'esempio Visual Basic viene visualizzata nelle seguenti pagine.

“Preparazione all'esecuzione degli esempi” a pagina 1122 e

“Gestione degli errori negli esempi” a pagina 1122

Esecuzione degli esempi starter ActiveX

Prima di eseguire gli esempi WebSphere MQ Automation Classes for ActiveX Starter, verificare che sia in esecuzione un gestore code predefinito e che siano state create le definizioni di coda richieste. Per i dettagli sulla creazione e l'esecuzione di un gestore code e sulla creazione di una coda, fare riferimento a Amministrazione. L'esempio utilizza la coda SYSTEM.DEFAULT.LOCAL.QUEUE che deve essere definito su qualsiasi server WebSphere MQ normalmente impostato.

Le diverse modalità di utilizzo dei bag di dati sono riportate nel seguente elenco:

- Connetterti a un gestore code
- Accesso a una coda
- Inserire un messaggio su una coda
- Richiamare un messaggio da una coda

Per informazioni sugli esempi starter MQAX per Microsoft Basic Versione 4 o successiva, consultare “Esecuzione dell'esempio di MQAXTRIV” a pagina 1122

Per informazioni su un esempio che consente di esplorare le proprietà e i metodi dei gestori code e degli oggetti coda, consultare “Avvio dell'esempio MQAXCLSS” a pagina 1124

Per informazioni sull'esempio MQAXDLST, “L'esempio MQAXDLST” a pagina 1124

Per informazioni sull'esecuzione dell'esempio starter MQAX per Microsoft Excel 95 o versioni successive, MQAXTRIV.XLS, consultare “Esecuzione di MQAXTRIV.XLS” a pagina 1124.

Per informazioni sull'esecuzione della dimostrazione Banca con MQAX.XLS, consultare “Esecuzione della dimostrazione della banca con MQAX.XLS XLS” a pagina 1124

Per informazioni sull'esempio starter utilizzando un browser WWW compatibile ActiveX , consultare “Esempio starter che utilizza un browser WWW compatibile ActiveX” a pagina 1125

Preparazione all'esecuzione degli esempi

Per eseguire uno degli esempi è necessario uno dei seguenti a seconda di quale degli esempi si intende eseguire.

- Microsoft Visual Basic Versione 4 (o successiva)
- Microsoft Excel 95 (o versione successiva)
- Un browser Web

Hai anche bisogno di:

- Un gestore code WebSphere MQ in esecuzione.
- Una coda WebSphere MQ è già definita.

Gestione degli errori negli esempi

La maggior parte degli esempi forniti nel pacchetto WebSphere MQ Automation Classes for ActiveX mostra una gestione degli errori minima o nulla. Per ulteriori informazioni sulla gestione degli errori, fare riferimento a “Gestione degli errori” a pagina 1040.

Esecuzione dell'esempio di MQAXTRIV

1. Avviare il gestore code.
2. In Esplora risorse di Windows o File Manager, selezionare l'icona per l'esempio, MQAXTRIV.VBP (Visual Basic Project file) e aprire il file.

   Il programma Visual Basic avvia e apre il file, MQAXTRIV.VBP.
3. In Visual Basic, premere il tasto funzionale 5 (F5) per eseguire l'esempio.
4. Fare clic in un punto qualsiasi del modulo della finestra, "MQAX banale tester".

Se tutto funziona correttamente, lo sfondo della finestra dovrebbe cambiare in verde. Se si verifica un problema con la configurazione, lo sfondo della finestra dovrebbe diventare rosso e verranno visualizzate le informazioni sull'errore.

La seguente figura mostra la parte centrale dell'esempio 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
```

Avvio dell'esempio MQAXCLSS

Questo esempio consente di sfogliare le proprietà e i metodi dei gestori code e degli oggetti coda.

1. Avviare il gestore code.
2. Aprire il file MQAXCLSS.VBP, facendo doppio clic sull'icona del documento in Windows Explorer o facendo clic su File - Apri dal menu File in Visual Basic.
3. Avviare l'esempio.
4. Immettere il gestore code e i nomi delle code appropriati, quindi fare clic sui pulsanti corrispondenti.

L'esempio MQAXDLST

L'esempio MQAXDLST di Visual Basic mostra l'utilizzo di un elenco di distribuzione per inviare lo stesso messaggio a due code con una sola immissione. Per eseguire l'esempio, eseguire la stessa operazione dell'esempio MQAXCLSS.

Esempio di MQAX Starter per Microsoft Excel 95 o successivo

Questa sezione spiega come eseguire l'esempio starter MQAX per Microsoft Excel 95 o versione successiva, MQAXTRIV.XLSXLS.

Esecuzione di MQAXTRIV.XLS

1. Avviare il gestore code.
2. In Explorer o File Manager, selezionare l'icona per l'esempio MQAX MQAXTRIV.XLSXLS.
3. Fare clic sul pulsante nel foglio di calcolo.
4. La schermata viene aggiornata con un messaggio di esito positivo (o negativo).

Esecuzione della dimostrazione della banca con MQAX.XLS XLS

Seguire questa procedura per eseguire la dimostrazione della banca.

1. Avviare il gestore code.
2. Eseguire il file di comandi IBM WebSphere MQ MQSC, BANK.TST. Ciò imposta le definizioni di coda IBM WebSphere MQ necessarie.

   Per informazioni su come utilizzare un file di comandi MQSC, fare riferimento a Comandi di script (MQSC).
3. Eseguire MQAXBSRV.VBP. Questo programma di esempio è il server che simula un'applicazione di back - end e deve essere eseguito con Microsoft Excel.
4. Eseguire MQAX.XLSXLS. Questo esempio è la dimostrazione IBM WebSphere MQ del client.
5. Selezionare un cliente dall'elenco.
6. Fai clic su **Inoltra**.

Dopo una breve pausa (circa 3 secondi), i campi vengono popolati con valori e viene visualizzato un diagramma a barre.

Esempio starter che utilizza un browser WWW compatibile ActiveX

Nota: Per eseguire questo esempio, è necessario eseguire un browser Web compatibile con ActiveX . Microsoft Internet Explorer (ma non Netscape Navigator) è un browser Web compatibile.

Esecuzione dell'esempio HTML

Questo esempio dimostra come richiamare MQAX sia da VBScript che JavaScript.

1. Avviare il gestore code.
2. Aprire il file, "MQAXTRIV.HTM", nel browser Web compatibile ActiveX .

   È possibile eseguire questa operazione facendo doppio clic sull'icona del file in Esplora risorse di Windows oppure scegliere File - Apri dal menu File del browser Web compatibile con ActiveX .
3. Seguire le istruzioni sullo schermo.

Informazioni particolari

Queste informazioni sono state sviluppate per i prodotti ed i servizi offerti negli Stati Uniti.

IBM potrebbe non offrire i prodotti, i servizi o le funzioni descritti in questo documento in altri paesi. Consultare il rappresentante IBM locale per informazioni sui prodotti e sui servizi disponibili nel proprio paese. Ogni riferimento relativo a prodotti, programmi o servizi IBM non implica che solo quei prodotti, programmi o servizi IBM possano essere utilizzati. In sostituzione a quelli forniti da IBM possono essere usati prodotti, programmi o servizi funzionalmente equivalenti che non comportino la violazione dei diritti di proprietà intellettuale o di altri diritti dell'IBM. È comunque responsabilità dell'utente valutare e verificare la possibilità di utilizzare altri programmi e/o prodotti, fatta eccezione per quelli espressamente indicati dall'IBM.

IBM potrebbe disporre di applicazioni di brevetti o brevetti in corso relativi all'argomento descritto in questo documento. La fornitura di tale documento non concede alcuna licenza a tali brevetti. Chi desiderasse ricevere informazioni relative a licenze può rivolgersi per iscritto a:

Director of Commercial Relations
IBM Corporation
Schoenaicher Str. 220
D-7030 Boeblingen
U.S.A.

Per richieste di licenze relative ad informazioni double-byte (DBCS), contattare il Dipartimento di Proprietà Intellettuale IBM nel proprio paese o inviare richieste per iscritto a:

Intellectual Property Licensing
Legge sulla proprietà intellettuale e legale
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

Il seguente paragrafo non si applica al Regno Unito o a qualunque altro paese in cui tali dichiarazioni sono incompatibili con le norme locali: INTERNATIONAL BUSINESS MACHINES CORPORATION FORNISCE LA PRESENTE PUBBLICAZIONE "NELLO STATO IN CUI SI TROVA" SENZA GARANZIE DI ALCUN TIPO, ESPRESSE O IMPLICITE, IVI INCLUSE, A TITOLO DI ESEMPIO, GARANZIE IMPLICITE DI NON VIOLAZIONE, DI COMMERCIABILITÀ E DI IDONEITÀ PER UNO SCOPO PARTICOLARE. Alcuni stati non consentono la rinuncia a garanzie esplicite o implicite in determinate transazioni; quindi la presente dichiarazione potrebbe non essere applicabile.

Questa pubblicazione potrebbe contenere imprecisioni tecniche o errori tipografici. Le informazioni incluse in questo documento vengono modificate su base periodica; tali modifiche vengono incorporate nelle nuove edizioni della pubblicazione. IBM si riserva il diritto di apportare miglioramenti o modifiche al prodotto/i e/o al programma/i descritti nella pubblicazione in qualsiasi momento e senza preavviso.

Qualsiasi riferimento a siti Web non IBM contenuto nelle presenti informazioni è fornito per consultazione e non vuole in alcun modo promuovere i suddetti siti Web. I materiali presenti in tali siti Web non sono parte dei materiali per questo prodotto IBM e l'utilizzo di tali siti Web è a proprio rischio.

Tutti i commenti e i suggerimenti inviati potranno essere utilizzati liberamente da IBM e diventeranno esclusiva della stessa.

Coloro che detengono la licenza su questo programma e desiderano avere informazioni su di esso allo scopo di consentire (i) uno scambio di informazioni tra programmi indipendenti ed altri (compreso questo) e (ii) l'uso reciproco di tali informazioni, dovrebbero rivolgersi a:

IBM Corporation
Coordinatore interoperabilità software, Dipartimento 49XA
Autostrada 3605 52 N

Rochester, MN 55901
U.S.A.

Queste informazioni possono essere rese disponibili secondo condizioni contrattuali appropriate, compreso, in alcuni casi, il pagamento di un addebito.

Il programma su licenza descritto in queste informazioni e tutto il materiale su licenza disponibile per esso sono forniti da IBM in base ai termini dell' IBM Customer Agreement, IBM International Program License Agreement o qualsiasi altro accordo equivalente tra le parti.

Tutti i dati relativi alle prestazioni contenuti in questo documento sono stati determinati in un ambiente controllato. Pertanto, i risultati ottenuti in altri ambienti operativi possono variare in modo significativo. Alcune misurazioni potrebbero essere state fatte su sistemi a livello di sviluppo e non vi è alcuna garanzia che queste misurazioni saranno le stesse sui sistemi generalmente disponibili. Inoltre, alcune misurazioni potrebbero essere state stimate mediante estrapolazione. I risultati quindi possono variare. Gli utenti di questo documento dovrebbero verificare i dati applicabili per il loro ambiente specifico.

Le informazioni relative a prodotti non IBM provengono dai fornitori di tali prodotti, dagli annunci pubblicati o da altre fonti pubblicamente disponibili. IBM non ha verificato tali prodotti e, pertanto, non può garantirne l'accuratezza delle prestazioni. Eventuali commenti relativi alle prestazioni dei prodotti non IBM devono essere indirizzati ai fornitori di tali prodotti.

Tutte le dichiarazioni riguardanti la direzione o l'intento futuro di IBM sono soggette a modifica o ritiro senza preavviso e rappresentano solo scopi e obiettivi.

Questa pubblicazione contiene esempi di dati e prospetti utilizzati quotidianamente nelle operazioni aziendali, Per illustrarle nel modo più completo possibile, gli esempi includono i nomi di individui, società, marchi e prodotti. Tutti questi nomi sono fittizi e qualsiasi somiglianza con nomi ed indirizzi adoperati da imprese realmente esistenti sono una mera coincidenza.

LICENZA SUL COPYRIGHT:

Queste informazioni contengono programmi applicativi di esempio in lingua originale, che illustrano le tecniche di programmazione su diverse piattaforme operative. È possibile copiare, modificare e distribuire questi programmi di esempio sotto qualsiasi forma senza alcun pagamento alla IBM, allo scopo di sviluppare, utilizzare, commercializzare o distribuire i programmi applicativi in conformità alle API (application programming interface) a seconda della piattaforma operativa per cui i programmi di esempio sono stati scritti. Questi esempi non sono stati testati approfonditamente tenendo conto di tutte le condizioni possibili. IBM, quindi, non può garantire o sottintendere l'affidabilità, l'utilità o il funzionamento di questi programmi.

Se si sta visualizzando queste informazioni in formato elettronico, le fotografie e le illustrazioni a colori potrebbero non apparire.

Informazioni sull'interfaccia di programmazione

Le informazioni sull'interfaccia di programmazione, se fornite, consentono di creare software applicativo da utilizzare con questo programma.

Questo manuale contiene informazioni sulle interfacce di programmazione che consentono al cliente di scrivere programmi per ottenere i servizi di IBM WebSphere MQ.

Queste informazioni, tuttavia, possono contenere diagnosi, modifica e regolazione delle informazioni. La diagnosi, la modifica e la regolazione delle informazioni vengono fornite per consentire il debug del software applicativo.

Importante: Non utilizzare queste informazioni di diagnosi, modifica e ottimizzazione come interfaccia di programmazione poiché sono soggette a modifica.

Marchi

IBM, il logo IBM , ibm.com, sono marchi di IBM Corporation, registrati in molte giurisdizioni nel mondo. Un elenco aggiornato dei marchi IBM è disponibile sul web in "Copyright and trademark

information"www.ibm.com/legal/copytrade.shtml. Altri nomi di prodotti e servizi potrebbero essere marchi di IBM o altre società.

Microsoft e Windows sono marchi di Microsoft Corporation negli Stati Uniti e/o in altri paesi.

UNIX è un marchio registrato di The Open Group negli Stati Uniti e/o in altri paesi.

Linux è un marchi registrato di Linus Torvalds negli Stati Uniti e/o in altri paesi.

Questo prodotto include il software sviluppato da Eclipse Project (http://www.eclipse.org/).

Java e tutti i marchi e i logo Java sono marchi registrati di Oracle e/o di società affiliate.

Numero parte:

(1P) P/N: