7.5

Amministrazione di IBM WebSphere MQ

Nota

Prima di utilizzare queste informazioni e il prodotto che supportano, leggere le informazioni in “Informazioni particolari” a pagina 163.

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

AmministrazioneIBM WebSphere MQ

La gestione dei gestori code e delle risorse associate include le attività che vengono eseguite di frequente per attivare e gestire tali risorse. Scegliere il metodo che si preferisce per gestire i gestori code e le risorse associate.

È possibile gestire gli oggetti IBM WebSphere MQ in locale o in remoto, consultare “Amministrazione locale e remota” a pagina 7.

Esistono diversi metodi che è possibile utilizzare per creare e gestire i gestori code e le risorse correlate in IBM WebSphere MQ. Questi metodi includono interfacce della riga comandi, una GUI (graphical user interface) e un'API di amministrazione. Consultare le sezioni e i link in questo argomento per ulteriori informazioni su ognuna di queste interfacce.

Esistono diverse serie di comandi che è possibile utilizzare per amministrare IBM WebSphere MQ a seconda della piattaforma:

- “Comandi di controllo IBM WebSphere MQ” a pagina 5
- “Comandi di script IBM WebSphere MQ (MQSC)” a pagina 5
- “PCF (Programmable Command Format)” a pagina 6

Ci sono anche le seguenti opzioni per la creazione e la gestione degli oggetti IBM WebSphere MQ :

- “Il IBM WebSphere MQ Explorer” a pagina 6
- “L'applicazione Configurazione predefinita Windows” a pagina 7
- “MSCS (Microsoft Cluster Service)” a pagina 7

È possibile automatizzare alcune attività di amministrazione e monitoraggio per i gestori code locali e remoti utilizzando i comandi PCF. Questi comandi possono essere semplificati anche mediante l'utilizzo di IBM WebSphere MQ Administration Interface (MQAI) su alcune piattaforme. Per ulteriori informazioni sull'automazione delle attività di amministrazione, consultare “Automazione delle attività di gestione” a pagina 8.

Comandi di controllo IBM WebSphere MQ

I comandi di controllo consentono di eseguire attività di gestione sui gestori code stessi.

IBM WebSphere MQ per Windows, i sistemi UNIX and Linux® forniscono i *comandi di controllo* immessi dalla riga comandi del sistema.

I comandi di controllo sono descritti in Creazione e gestione dei gestori code. Per il riferimento ai comandi per i comandi di controllo, consultare IBM WebSphere MQ Comandi di controllo.

Comandi di script IBM WebSphere MQ (MQSC)

Utilizzare i comandi MQSC per gestire gli oggetti del gestore code, inclusi il gestore code, le code, le definizioni di processo, gli elenchi nomi, i canali, i canali di connessione client, i listener, i servizi e gli oggetti delle informazioni di autenticazione.

Si immettono comandi MQSC a un gestore code utilizzando il comando `runmqsc` . È possibile eseguire questa operazione in modo interattivo, immettendo i comandi da una tastiera oppure reindirizzare l'unità di input standard (stdin) per eseguire una sequenza di comandi da un file di testo ASCII. In entrambi i casi, il formato dei comandi è lo stesso.

È possibile eseguire il comando `runmqsc` in tre modalità, a seconda degli indicatori impostati sul comando:

- *Modalità di verifica*, in cui i comandi MQSC vengono verificati su un gestore code locale, ma non eseguiti
- *Modalità diretta*, in cui i comandi MQSC vengono eseguiti su un gestore code locale

- *Modalità indiretta*, in cui i comandi MQSC vengono eseguiti su un gestore code remoto

Gli attributi dell'oggetto specificati nei comandi MQSC vengono mostrati in maiuscolo in questa sezione (ad esempio, RQMNAME), sebbene non siano sensibili al maiuscolo / minuscolo. I nomi attributo del comando MQSC sono limitati a otto caratteri.

I comandi MQSC sono disponibili su tutte le piattaforme. I comandi MQSC sono riepilogati in Confronto delle serie comandi.

Su Windows, UNIX o Linux, è possibile utilizzare MQSC come singoli comandi emessi sulla riga comandi di sistema. Per emettere comandi più complessi o multipli, MQSC può essere integrato in un file eseguito dalla riga comandi del sistema Windows, UNIX o Linux . MQSC può essere inviato a un gestore code remoto. Per i dettagli completi, consultare Riferimento MQSC.

"Comandi script (MQSC)" a pagina 74 contiene una descrizione di ciascun comando MQSC e la sua sintassi.

Consultare "Esecuzione di attività di amministrazione locale utilizzando i comandi MQSC" a pagina 73 per ulteriori informazioni sull'utilizzo dei comandi MQSC nella gestione locale.

PCF (Programmable Command Format)

I PCF (Programmable Command Format) definiscono i messaggi di comando e risposta che possono essere scambiati tra un programma e qualsiasi gestore code (che supporta PCF) in una rete. È possibile utilizzare comandi PCF in un programma applicativo di gestione dei sistemi per la gestione di oggetti IBM WebSphere MQ : oggetti delle informazioni di autenticazione, canali, listener di canali, elenchi nomi, definizioni di processo, gestori code, code, servizi e classi di memoria. L'applicazione può operare da un singolo punto nella rete per comunicare le informazioni sul comando e sulla risposta con qualsiasi gestore code, locale o remoto, utilizzando il gestore code locale.

Per ulteriori informazioni sui PCF, consultare "Introduzione ai formati di comando programmabili" a pagina 9.

Per la definizione di PCF e strutture per comandi e risposte, consultare Programmable command formats reference.

Il IBM WebSphere MQ Explorer

Utilizzando IBM WebSphere MQ Explorer, è possibile eseguire le seguenti azioni:

- Definire e controllare varie risorse, inclusi gestori code, code, definizioni di processi, elenchi nomi, canali, canali di connessione client, listener, servizi e cluster.
- Avviare o arrestare un gestore code locale e i relativi processi associati.
- Visualizzare i gestori code e i relativi oggetti associati sulla stazione di lavoro o da altre stazioni di lavoro.
- Verificare lo stato dei gestori code, dei cluster e dei canali.
- Verificare quali applicazioni, utenti o canali hanno una particolare coda aperta, dallo stato della coda.

Su sistemi Windows e Linux , è possibile avviare IBM WebSphere MQ Explorer utilizzando il menu di sistema, il file eseguibile `MQExplorer` o il comando **`strmqcfg`** .

Su Linux, per avviare correttamente IBM WebSphere MQ Explorer , è necessario essere in grado di scrivere un file nella directory home e la directory home deve esistere.

È possibile utilizzare IBM WebSphere MQ Explorer per gestire i gestori code remoti su altre piattaforme, incluso z/OS, per dettagli e per scaricare SupportPac MS0T, consultare https://www.ibm.com/support/docview.wss?uid=swg24021041.

Per ulteriori informazioni, fare riferimento a "Amministrazione mediante IBM WebSphere MQ Explorer" a pagina 57.

L'applicazione Configurazione predefinita Windows

È possibile utilizzare il programma Configurazione predefinita di Windows per creare una serie *starter* (o predefinita) di oggetti IBM WebSphere MQ . Un riepilogo degli oggetti predefiniti creati viene elencato nella tabella 1: oggetti creati dall'applicazione di configurazione predefinita Windows.

MSCS (Microsoft Cluster Service)

Microsoft Cluster Service (MSCS) consente di collegare i server in un *cluster*, fornendo una maggiore disponibilità di dati e applicazioni e rendendo più semplice la gestione del sistema. MSCS è in grado di rilevare e ripristinare automaticamente gli errori del server o dell'applicazione.

È importante non confondere i cluster in senso MSCS con i cluster IBM WebSphere MQ . La distinzione è:

IBM WebSphere MQCluster
sono gruppi di due o più gestori code su uno o più computer, che forniscono l'interconnessione automatica e consentono la condivisione delle code per il bilanciamento del carico e la ridondanza.

Cluster MSCS
Gruppi di computer, collegati tra loro e configurati in modo tale che, in caso di errore, MSCS esegua un *failover*, trasferendo i dati di stato delle applicazioni dal computer malfunzionante ad un altro computer nel cluster e inizializzando nuovamente le operazioni.

Il supporto di Microsoft Cluster Service (MSCS) fornisce informazioni dettagliate su come configurare il sistema IBM WebSphere MQ per Windows per utilizzare MSCS.

Concetti correlati

Panoramica tecnica di WebSphere MQ

“Amministrazione di oggetti IBM WebSphere MQ locali” a pagina 69
Questa sezione spiega come gestire gli oggetti IBM WebSphere MQ locali per supportare i programmi applicativi che utilizzano MQI (Message Queue Interface). In questo contesto, amministrazione locale significa creare, visualizzare, modificare, copiare ed eliminare oggetti IBM WebSphere MQ .

“Gestione di oggetti IBM WebSphere MQ remoti” a pagina 105

Considerazioni quando si perde il contatto con il gestore risorse XA

Attività correlate

Pianificazione

Configurazione

Riferimenti correlati

Scenari di supporto transazionale

Amministrazione locale e remota

È possibile gestire gli oggetti WebSphere MQ in locale o in remoto.

Amministrazione locale indica l'esecuzione di attività di gestione su qualsiasi gestore code definito sul sistema locale. È possibile accedere ad altri sistemi, ad esempio tramite **telnet**del programma di emulazione terminale TCP/IP, ed eseguire la gestione. In WebSphere MQ, è possibile considerarlo come amministrazione locale poiché non sono coinvolti canali, ovvero la comunicazione è gestita dal sistema operativo.

WebSphere MQ supporta la gestione da un singolo punto di contatto tramite ciò che è noto come *amministrazione remota*. Ciò consente di immettere comandi dal sistema locale che vengono elaborati su un altro sistema e si applicano anche a WebSphere MQ Explorer. Ad esempio, è possibile immettere un comando remoto per modificare una definizione di coda su un gestore code remoto. Non è necessario accedere a tale sistema, anche se è necessario definire i canali appropriati. Il gestore code e il server dei comandi sul sistema di destinazione devono essere in esecuzione.

Alcuni comandi non possono essere emessi in questo modo, in particolare, creando o avviando i gestori code e avviando i server dei comandi. Per eseguire questo tipo di attività, è necessario collegarsi al

sistema remoto e immettere i comandi da tale sistema oppure creare un processo che possa emettere i comandi per l'utente. Questa limitazione si applica anche a WebSphere MQ Explorer.

"Gestione di oggetti IBM WebSphere MQ remoti" a pagina 105 descrive in modo più dettagliato l'argomento della gestione remota.

Come utilizzare i comandi di controllo IBM WebSphere MQ

Questa sezione descrive come utilizzare i comandi di controllo IBM WebSphere MQ .

Se si desidera immettere comandi di controllo, l'ID utente deve essere un membro del gruppo mqm. Per ulteriori informazioni, consultare Authority to amministrare WebSphere MQ su sistemi UNIX, Linux e Windows . Inoltre, si notano le seguenti informazioni specifiche dell'ambiente:

WebSphere MQ per Windows
Tutti i comandi di controllo possono essere emessi da una riga comandi. I nomi dei comandi e i relativi indicatori non sono sensibili al maiuscolo / minuscolo: è possibile immetterli in maiuscolo, minuscolo o in una combinazione di maiuscolo e minuscolo. Tuttavia, gli argomenti per controllare i comandi (come i nomi delle code) sono sensibili al maiuscolo / minuscolo.

Nelle descrizioni della sintassi, il trattino (-) viene utilizzato come indicatore di indicatore. È possibile utilizzare la barra (/) invece del trattino.

Sistemi WebSphere MQ per UNIX and Linux
Tutti i comandi di controllo WebSphere MQ possono essere emessi da una shell. Tutti i comandi sono sensibili al maiuscolo / minuscolo.

Un sottoinsieme dei comandi di controllo può essere immesso utilizzando Esplora risorse di IBM WebSphere MQ .

Per ulteriori informazioni, consultare I comandi di controllo WebSphere MQ

Automazione delle attività di gestione

Si potrebbe decidere che sarebbe utile per l'installazione automatizzare alcune attività di gestione e monitoraggio. È possibile automatizzare le attività di gestione per gestori code locali e remoti utilizzando i comandi PCF (Programmable Command Format). Questa sezione presuppone che l'utente abbia esperienza nella gestione di WebSphere MQ oggetti.

Comandi PCF

WebSphere MQ i comandi PCF (programmable command format) possono essere utilizzati per programmare le attività di amministrazione in un programma di amministrazione. In tal modo, da un programma è possibile modificare gli oggetti del gestore code (code, definizioni di processi, elenchi nomi, canali, canali di connessione client, listener, servizi e oggetti delle informazioni di autenticazione) e persino modificare i gestori code stessi.

I comandi PCF coprono la stessa gamma di funzioni fornite dai comandi MQSC. È possibile scrivere un programma per emettere comandi PCF su qualsiasi gestore code nella rete da un singolo nodo. In questo modo, è possibile centralizzare e automatizzare le attività di gestione.

Ogni comando PCF è una struttura di dati integrata nella parte di dati dell'applicazione di un messaggio WebSphere MQ . Ciascun comando viene inviato al gestore code di destinazione utilizzando la funzione MQI `MQPUT` allo stesso modo di qualsiasi altro messaggio. Se il server dei comandi è in esecuzione sul gestore code che riceve il messaggio, il server dei comandi lo interpreta come messaggio di comando ed esegue il comando. Per ottenere le repliche, l'applicazione emette una chiamata `MQGET` e i dati di risposta vengono restituiti in un'altra struttura dati. La domanda può quindi elaborare la risposta e agire di conseguenza.

Nota: A differenza dei comandi MQSC, i comandi PCF e le relative risposte non sono in un formato di testo leggibile.

Brevemente, queste sono alcune delle cose necessarie per creare un messaggio di comando PCF:

Descrittore messaggio
Questo è un descrittore di messaggi WebSphere MQ standard, in cui:

- Il tipo di messaggio (*MsqType*) è MQMT_REQUEST.
- Il formato del messaggio (*Format*) è MQFMT_ADMIN.

Dati applicazione
Contiene il messaggio PCF che include l'intestazione PCF, in cui:

- Il tipo di messaggio PCF (*Type*) specifica MQCFT_COMMAND.
- L'identificativo del comando specifica il comando, ad esempio, *Change Queue* (MQCMD_CHANGE_Q).

Per una descrizione completa delle strutture di dati PCF e come implementarle, consultare "Introduzione ai formati di comando programmabili" a pagina 9.

Attributi oggetto PCF

Gli attributi degli oggetti in PCF non sono limitati a otto caratteri come per i comandi MQSC. Sono mostrati in questa guida in corsivo. Ad esempio, l'equivalente PCF di RQMNAME è *RemoteQMgrName*.

PCF di escape

I PCF di escape sono comandi PCF che contengono comandi MQSC all'interno del testo del messaggio. È possibile utilizzare i PCF per inviare comandi a un gestore code remoto. Per ulteriori informazioni sui PCF di escape, consultare Escape.

Introduzione ai formati di comando programmabili

I PCF (Programmable Command Format) definiscono i messaggi di comando e risposta che possono essere scambiati tra un programma e qualsiasi gestore code (che supporta PCF) in una rete. I PCF semplificano la gestione dei gestori code e di altre reti. Possono essere utilizzati per risolvere il problema della gestione complessa di reti distribuite, soprattutto quando le reti crescono in dimensioni e complessità.

I formati dei comandi programmabili descritti in questa documentazione sono supportati da:

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

Il problema che i comandi PCF risolvono

La gestione delle reti distribuite può diventare complessa. I problemi amministrativi continuano a crescere con l'aumento delle dimensioni e della complessità delle reti.

Esempi di gestione specifici per la messaggistica e l'accodamento includono:

- Gestione delle risorse.

  Ad esempio, creazione ed eliminazione della coda.
- Monitoraggio delle prestazioni.

  Ad esempio, la profondità massima della coda o la frequenza dei messaggi.
- Controllo.

Ad esempio, l'ottimizzazione dei parametri della coda come la profondità massima della coda, la lunghezza massima dei messaggi e l'abilitazione e disabilitazione delle code.

- Instradamento del messaggio.

  Definizione di percorsi alternativi attraverso una rete.

WebSphere MQ I comandi PCF possono essere usati per semplificare la gestione dei gestori code e di altre reti. I comandi PCF consentono di utilizzare una singola applicazione per eseguire la gestione della rete da un singolo gestore code nella rete.

Che cosa sono i PCF?

I PCF definiscono i messaggi di comando e di risposta che possono essere scambiati tra un programma e qualsiasi gestore code (che supporta PCF) in una rete. È possibile utilizzare i comandi PCF in un programma di applicazione di gestione dei sistemi per la gestione degli oggetti WebSphere MQ : oggetti delle informazioni di autenticazione, canali, listener dei canali, elenchi nomi, definizioni dei processi, gestori code, code, servizi e classi di memoria. L'applicazione può operare da un singolo punto nella rete per comunicare le informazioni sul comando e sulla risposta con qualsiasi gestore code, locale o remoto, utilizzando il gestore code locale.

Ciascun gestore code dispone di una coda di amministrazione con un nome coda standard e l'applicazione può inviare messaggi di comando PCF a tale coda. Ogni gestore code ha anche un server dei comandi per gestire i messaggi di comando dalla coda di amministrazione. I messaggi di comando PCF possono quindi essere elaborati da qualsiasi gestore code nella rete e i dati di risposta possono essere restituiti all'applicazione, utilizzando la coda di risposta specificata. I comandi PCF e i messaggi di risposta vengono inviati e ricevuti utilizzando la normale MQI (Message Queue Interface).

Per un elenco dei comandi PCF disponibili, inclusi i relativi parametri, consultare Definizioni dei formati dei comandi programmabili.

Utilizzo dei formati di comando programmabili

È possibile utilizzare i PCF in un programma di gestione sistemi per la gestione remota WebSphere MQ .

Questa sezione include:

- “Messaggi di comando PCF” a pagina 10
- “Risposte” a pagina 13
- Regole per la denominazione di oggetti IBM WebSphere MQ
- “Controllo autorizzazione per comandi PCF” a pagina 15

Messaggi di comando PCF

I messaggi di comando PCF sono costituiti da un'intestazione PCF, parametri identificati in tale intestazione e anche dati di messaggi definiti dall'utente. I messaggi vengono emessi utilizzando le chiamate dell'interfaccia della coda messaggi.

Ogni comando e i relativi parametri vengono inviati come un messaggio di comando separato contenente un'intestazione PCF seguita da un certo numero di strutture di parametro; per i dettagli dell'intestazione PCF, consultare MQCFH - PCF headere per un esempio di una struttura di parametro, consultare MQCFST - PCF string parameter. L'intestazione PCF identifica il comando e il numero di strutture di parametri che seguono nello stesso messaggio. Ogni struttura di parametri fornisce un parametro al comando.

Le risposte ai comandi, generate dal server dei comandi, hanno una struttura simile. C'è un'intestazione PCF, seguita da un numero di strutture di parametri. Le risposte possono essere costituite da più di un messaggio, ma i comandi sono sempre composti da un solo messaggio.

Su piattaforme diverse da z/OS, la coda a cui vengono inviati comandi PCF viene sempre denominata SYSTEM.ADMIN.COMMAND.QUEUE.

Come emettere i messaggi di comando PCF

Utilizzare le normali chiamate MQI (Message Queue Interface), MQPUT, MQGET e così via, per inserire e richiamare il comando PCF e i messaggi di risposta da e verso le relative code.

Nota:

Verificare che il server dei comandi sia in esecuzione sul gestore code di destinazione per il comando PCF da elaborare su tale gestore code.

Per un elenco dei file di intestazione forniti, vedere WebSphere MQ COPY, header, include e module files.

Descrittori di messaggi per un comando PCF

Il descrittore del messaggio WebSphere MQ è documentato in MQMD - Message descriptor.

Un messaggio di comando PCF contiene i seguenti campi nel descrittore del messaggio:

Report
Qualsiasi valore valido, come richiesto.

MsgType
Questo campo deve essere MQMT_REQUEST per indicare un messaggio che richiede una risposta.

Expiry
Qualsiasi valore valido, come richiesto.

Feedback
Imposta su MQFB_NONE

Encoding
Se si sta effettuando l'invio a sistemi Windows, UNIX o Linux , impostare questo campo sulla codifica utilizzata per i dati del messaggio; la conversione viene eseguita se necessario.

CodedCharSetId
Se si sta inviando aWindows, UNIX o sistemi Linux , impostare questo campo sull'identificativo della serie di caratteri codificati utilizzato per i dati del messaggio; la conversione viene eseguita se necessario.

Format
Impostare su MQFMT_ADMIN.

Priority
Qualsiasi valore valido, come richiesto.

Persistence
Qualsiasi valore valido, come richiesto.

MsgId
L'applicazione di invio può specificare qualsiasi valore oppure è possibile specificare MQMI_NONE per richiedere al gestore code di creare un identificativo di messaggio univoco.

CorrelId
L'applicazione di invio può specificare qualsiasi valore oppure MQCI_NONE può essere specificato per indicare nessun identificativo di correlazione.

ReplyToQ
Il nome della coda per ricevere la risposta.

ReplyToQMgr
Il nome del gestore code per la risposta (o vuoto).

Campi di contesto del messaggio
Questi campi possono essere impostati su qualsiasi valore valido, come richiesto. Di solito, l'opzione di inserimento del messaggio MQPMO_DEFAULT_CONTEXT viene utilizzata per impostare i campi di contesto del messaggio sui valori predefiniti.

Se si sta utilizzando una struttura MQMD version-2 , è necessario impostare i seguenti campi aggiuntivi:

GroupId
Imposta su MQGI_NONE

MsgSeqNumber
Imposta su 1

Offset
Imposta su 0

MsgFlags
Imposta su MQMF_NONE

OriginalLength
Imposta su MQOL_UNDEFINED

Invio di dati utente

Le strutture PCF possono essere utilizzate anche per inviare dati di messaggi definiti dall'utente. In questo caso, il campo *Format* del descrittore del messaggio deve essere impostato su MQFMT_PCF.

Invio e ricezione di messaggi PCF in una coda specificata

Invio di messaggi PCF a una coda specificata

Per inviare un messaggio a una coda specificata, la chiamata Bag mqPutconverte il contenuto della borsa specificata in un messaggio PCF e invia il messaggio alla coda specificata. Il contenuto della borsa rimane invariato dopo la chiamata.

Come input per questa chiamata, è necessario fornire:

- Un handle di connessione MQI.
- Un handle di oggetto per la coda su cui deve essere inserito il messaggio.
- Un descrizione del messaggio. Per ulteriori informazioni sul descrittore del messaggio, consultare MQMD - Descrittore del messaggio.
- Inserire le opzioni del messaggio utilizzando la struttura MQPMO. Per ulteriori informazioni sulla struttura MQPM, consultare MQPMO - Put - message options.
- L'handle della borsa da convertire in un messaggio.

  Nota: Se il contenitore contiene un messaggio di gestione e la chiamata di interrogazione mqAddè stata utilizzata per inserire i valori nel contenitore, il valore dell'elemento di dati MQIASY_COMMAND deve essere un comando INQUIRE riconosciuto da MQAI.

Per una descrizione completa della chiamata Bag mqPut, consultare mqPutBag.

Ricezione di messaggi PCF da una coda specificata

Per ricevere un messaggio da una coda specificata, la chiamata al contenitore mqGetriceve un messaggio PCF da una coda specificata e converte i dati del messaggio in un contenitore dati.

Come input per questa chiamata, è necessario fornire:

- Un handle di connessione MQI.
- Una gestione oggetto della coda da cui deve essere letto il messaggio.
- Un descrizione del messaggio. Nella struttura MQMD, il parametro `Format` deve essere MQFMT_ADMIN, MQFMT_EVENT o MQFMT_PCF.

  Nota: Se il messaggio viene ricevuto all'interno di un'unità di lavoro (con l'opzione MQGMO_SYNCPOINT) e il formato del messaggio non è supportato, è possibile eseguire il backout dell'unità di lavoro. Il messaggio viene quindi reintegrato nella coda e può essere recuperato utilizzando

la chiamata MQGET invece della chiamata Bag mqGet. Per ulteriori informazioni sul descrittore del messaggio, consultare MQGMO - Get - message options.

- Richiamare le opzioni del messaggio utilizzando la struttura MQGMO. Per ulteriori informazioni sulla struttura MQGMO, consultare MQMD - Message Descriptor.
- L'handle del contenitore per contenere il messaggio convertito.

Per una descrizione completa della chiamata Bag mqGet, consultare mqGetBag.

Risposte

In risposta a ciascun comando, il server dei comandi genera uno o più messaggi di risposta. Un messaggio di risposta ha un formato simile a un messaggio di comando.

L'intestazione PCF ha lo stesso valore identificativo del comando a cui è una risposta (consultare MQCFH - intestazione PCF per i dettagli). L'identificativo del messaggio e l'identificativo di correlazione vengono impostati in base alle opzioni di report della richiesta.

Se il tipo di intestazione PCF del messaggio di comando è MQCFT_COMMAND, vengono generate solo le risposte standard. Tali comandi sono supportati su tutte le piattaforme tranne z/OS. Le applicazioni precedenti non supportano PCF su z/OS; WebSphere MQ Windows Explorer è una di queste applicazioni (tuttavia, la versione 6.0 o successiva IBM WebSphere MQ Explorer supporta PCF su z/OS).

Se il tipo di intestazione PCF del messaggio di comando è MQCFT_COMMAND_XR, vengono generate risposte estese o standard. Tali comandi sono supportati su z/OS e su alcune altre piattaforme. I comandi emessi su z/OS generano solo risposte estese. Su altre piattaforme, è possibile che venga generato un tipo di risposta.

Se un singolo comando specifica un nome oggetto generico, viene restituita una risposta separata nel proprio messaggio per ogni oggetto corrispondente. Per la generazione della risposta, un singolo comando con un nome generico viene considerato come più comandi singoli (ad eccezione del campo di controllo MQCFC_LAST o di MQCFC_NOT_LAST). In caso contrario, un messaggio di comando genera un messaggio di risposta.

Alcune risposte PCF potrebbero restituire una struttura anche quando non è richiesta. Questa struttura viene mostrata nella definizione della risposta (Definizioni dei formati del comando programmabile) come *sempre restituito*. Il motivo per cui, per queste risposte, è necessario denominare gli oggetti nella risposta per identificare quale oggetto vengono applicati i dati.

Descrittori di messaggi per una risposta

Un messaggio di risposta ha i seguenti campi nel descrittore del messaggio:

MsgType
Questo campo è MQMT_REPLY.

MsgId
Questo campo viene generato dal gestore code.

CorrelId
Questo campo viene creato in base alle opzioni di prospetto del messaggio di comando.

Format
Questo campo è MQFMT_ADMIN.

Encoding
Impostare su MQENC_NATIVE.

CodedCharSetId
Impostare su MQCCSI_Q_MGR.

Persistence
Lo stesso come nel messaggio di comando.

Priority
Lo stesso come nel messaggio di comando.

La risposta viene generata con MQPMO_PASS_IDENTITY_CONTEXT.

Risposte standard

Messaggi di comando con un tipo di intestazione MQCFT_COMMAND, vengono generate risposte standard. Tali comandi sono supportati su tutte le piattaforme tranne z/OS.

Esistono tre tipi di risposta standard:

- Risposta OK
- Risposta di errore
- Risposta dati

Risposta OK

Questa risposta è costituita da un messaggio che inizia con un'intestazione del formato del comando, con un campo *CompCode* MQCC_OK o MQCC_WARNING.

Per MQCC_OK, *Reason* è MQRC_NONE.

Per MQCC_WARNING, *Reason* identifica la natura dell'avvertenza. In questo caso, l'intestazione del formato del comando potrebbe essere seguita da una o più strutture di parametri di avvertenza appropriate a questo codice di errore.

In entrambi i casi, per un comando inquire potrebbero seguire ulteriori strutture di parametri, come descritto nelle seguenti sezioni.

Risposta di errore

Se il comando ha un errore, vengono inviati uno o più messaggi di risposta di errore (più di uno potrebbe essere inviato anche per un comando che normalmente avrebbe un solo messaggio di risposta). Questi messaggi di risposta di errore hanno MQCFC_LAST o MQCFC_NOT_LAST impostati come appropriato.

Ogni messaggio inizia con un'intestazione del formato della risposta, con un valore *CompCode* MQCC_FAILED e un campo *Reason* che identifica il particolare errore. In generale, ogni messaggio descrive un errore differente. Inoltre, ogni messaggio ha zero o una (mai più di una) struttura di parametri di errore dopo l'intestazione. Questa struttura di parametri, se presente, è una struttura MQCFIN, con un campo *Parameter* che contiene uno dei seguenti:

- ID_PARAMETER_MQIACF

  Il campo *Value* nella struttura è l'identificativo del parametro che era in errore (ad esempio, MQCA_Q_NAME).

- ID_ERRORE_MQIACF

  Questo valore viene utilizzato con un valore *Reason* (nell'intestazione del formato del comando) di MQRC_UNEXPECTED_ERROR. Il campo *Value* nella struttura MQCFIN è il codice motivo non previsto ricevuto dal server dei comandi.

- MQIACF_SELECTOR

  Questo valore si verifica se una struttura di elenco (MQCFIL) inviata con il comando contiene un selettore duplicato o uno non valido. Il campo *Reason* nell'intestazione del formato del comando identifica l'errore e il campo *Value* nella struttura MQCFIN è il valore del parametro nella struttura MQCFIL del comando che era in errore.

- ERRORE_MQIACF_OFFSET

  Questo valore si verifica quando si verifica un errore di confronto dati nel comando Canale di ping. Il campo *Value* nella struttura è l'offset dell'errore di confronto del canale di ping.

- ID_MQIA_CODED_CHAR_SET_

  Questo valore si verifica quando il CCSID (coded character set identifier) nel descrittore del messaggio del comando PCF in entrata non corrisponde a quello del gestore code di destinazione. Il campo *Value* nella struttura è l'identificativo della serie di caratteri codificati del gestore code.

L'ultimo (o unico) messaggio di risposta di errore è una risposta di riepilogo, con un campo *CompCode* di MQCC_FAILED e un campo *Reason* di MQRCCF_COMMAND_FAILED. Questo messaggio non ha una struttura di parametri che segue l'intestazione.

Risposta dati

Questa risposta consiste in una risposta OK (come descritto in precedenza) a un comando inquire. La risposta OK è seguita da ulteriori strutture contenenti i dati richiesti come descritto in Definizioni dei formati di comando programmabili.

Le applicazioni non devono dipendere da queste strutture di parametri aggiuntive che vengono restituite in un ordine particolare.

Controllo autorizzazione per comandi PCF

Quando viene elaborato un comando PCF, il *UserIdentifier* dal descrittore del messaggio nel messaggio di comando viene utilizzato per i controlli dell'autorizzazione dell'oggetto WebSphere MQ richiesti. Il controllo dell'autorizzazione viene implementato in maniera diversa su ciascuna piattaforma, come descritto in questo argomento.

I controlli vengono eseguiti sul sistema su cui viene elaborato il comando; pertanto, questo ID utente deve esistere sul sistema di destinazione e disporre delle autorizzazioni richieste per elaborare il comando. Se il messaggio proviene da un sistema remoto, un modo per ottenere l'ID esistente sul sistema di destinazione è avere un ID utente corrispondente sia sul sistema locale che su quello remoto.

IBM WebSphere MQ per Windows, sistemi UNIX and Linux

UNIX Linux Windows

Per elaborare qualsiasi comando PCF, l'ID utente deve disporre dell'autorizzazione *dsp* per l'oggetto gestore code sul sistema di destinazione. Inoltre, vengono eseguiti i controlli dell'autorizzazione oggetto WebSphere MQ per determinati comandi PCF, come mostrato nella Tabella 1 a pagina 16.

Per elaborare uno dei seguenti comandi l'ID utente deve appartenere al gruppo *mqm*.

Nota: Solo per Windows , l'ID utente può appartenere al gruppo *Administrators* o al gruppo *mqm*.

- Modifica canale
- Copia canale
- Creazione canale
- Eliminazione canale
- Ping canale
- Reimposta canale
- Avvio canale
- Arresta canale
- Avvio iniziatore canale
- Avvio listener canale
- Risolvi canale
- Reimposta cluster
- Aggiornamento cluster
- Sospensione gestore code
- Ripristino gestore code

WebSphere MQ per HP Integrity NonStop Server

Per elaborare qualsiasi comando PCF, l'ID utente deve disporre dell'autorizzazione *dsp* per l'oggetto gestore code sul sistema di destinazione. Inoltre, vengono eseguiti controlli dell'autorizzazione dell'oggetto IBM WebSphere MQ per alcuni comandi PCF, come mostrato nella Tabella 1 a pagina 16.

Per elaborare uno dei seguenti comandi l'ID utente deve appartenere al gruppo *mqm*:

- Modifica canale
- Copia canale
- Creazione canale
- Eliminazione canale
- Ping canale
- Reimposta canale
- Avvio canale
- Arresta canale
- Avvio iniziatore canale
- Avvio listener canale
- Risolvi canale
- Reimposta cluster
- Aggiornamento cluster
- Sospensione gestore code
- Ripristino gestore code

WebSphere MQ

Tabella 1. Windows, HP Integrity NonStop Server, UNIX and Linux sistemi - autorizzazioni oggetto

Comando	**Autorizzazione oggetto WebSphere MQ**	**Autorizzazione classe (per tipo di oggetto)**
Modifica informazioni di autenticazione	dsp e chg	n/a
Modifica canale	dsp e chg	n/a
Modifica listener canale	dsp e chg	n/a
Modifica canale di connessione client	dsp e chg	n/a
Modifica elenco nomi	dsp e chg	n/a
Modifica processo	dsp e chg	n/a
Modifica coda	dsp e chg	n/a
Modifica gestore code	chg *vedere Nota 3 e Nota 5*	n/a
Modifica servizio	dsp e chg	n/a
Cancellazione coda	clr	n/a
Copia informazioni di autenticazione	dsp	crt
Copia informazioni di autenticazione (Sostituisci) *vedere la Nota 1*	*da:* dsp *a* chg	crt

Tabella 1. Windows, HP Integrity NonStop Server, UNIX and Linux sistemi - autorizzazioni oggetto (Continua)		
Comando	**Autorizzazione oggetto WebSphere MQ**	**Autorizzazione classe (per tipo di oggetto)**
Copia canale	dsp	crt
Copia canale (Sostituisci) *vedere la nota 1*	*da:* dsp *a* chg	crt
Copia listener canale	dsp	crt
Copiare il listener del canale (Sostituisci) *vedere la Nota 1*	*da:* dsp *a* chg	crt
Copia canale di connessione client	dsp	crt
Copia canale di connessione client (Sostituisci) *vedere la nota 1*	*da:* dsp *a* chg	crt
Copia elenco nomi	dsp	crt
Copia elenco nomi (Sostituisci) *vedere la nota 1*	*da:* dsp *a* dsp e chg	crt
Copia processo	dsp	crt
Copia processo (Sostituisci) *vedere Nota 1*	*da:* dsp *a* chg	crt
Copia coda	dsp	crt
Copia coda (Sostituisci) *vedere la nota 1*	*da:* dsp *a* dsp e chg	crt
Creazione informazioni di autenticazione	*(informazioni di autenticazione predefinite del sistema)* dsp	crt
Crea informazioni di autenticazione (Sostituisci) *vedere la nota 1*	*(informazioni di autenticazione predefinite del sistema)* dsp *per:* chg	crt
Creazione canale	*(canale predefinito del sistema)* dsp	crt
Crea canale (Sostituisci) *vedere la Nota 1*	*(canale predefinito di sistema)* dsp *in:* chg	crt
Crea listener del canale	*(listener predefinito di sistema)* dsp	crt
Crea listener canale (Sostituisci) *vedere la nota 1*	*(listener predefinito di sistema)* dsp *in:* chg	crt
Crea canale di connessione client	*(canale predefinito del sistema)* dsp	crt
Crea canale di connessione client (Sostituisci) *vedere la nota 1*	*(canale predefinito di sistema)* dsp *in:* chg	crt
Creazione elenco nomi	*(elenco nomi predefinito di sistema)* dsp	crt
Crea elenco nomi (sostituisci) *vedere la Nota 1*	*(elenco nomi predefinito di sistema)* dsp *su:* dsp e chg	crt
Creazione processo	*(processo predefinito del sistema)* dsp	crt

Tabella 1. Windows, HP Integrity NonStop Server, UNIX and Linux sistemi - autorizzazioni oggetto (Continua)		
Comando	**Autorizzazione oggetto WebSphere MQ**	**Autorizzazione classe (per tipo di oggetto)**
Crea processo (Sostituisci) *vedere la Nota 1*	*(processo predefinito di sistema)* dsp *to:* chg	crt
Creazione coda	*(coda predefinita di sistema)* dsp	crt
Crea coda (Sostituisci) *vedere la Nota 1*	*(coda predefinita del sistema)* dsp *a:* dsp e chg	crt
Crea servizio	*(coda predefinita di sistema)* dsp	crt
Crea servizio (Sostituisci) *vedere la Nota 1*	*(coda predefinita di sistema)* dsp *in:* chg	crt
Eliminazione informazioni di autenticazione	dsp e dlt	n/a
Eliminare il record di autorizzazione	*(oggetto gestore code)* chg *vedere la nota 4*	*vedere la Nota 4*
Eliminazione canale	dsp e dlt	n/a
Elimina listener canale	dsp e dlt	n/a
Elimina canale di connessione client	dsp e dlt	n/a
Eliminazione elenco nomi	dsp e dlt	n/a
Eliminazione processo	dsp e dlt	n/a
Eliminazione coda	dsp e dlt	n/a
Elimina servizio	dsp e dlt	n/a
Interrogazione informazioni di autenticazione	dsp	n/a
Interrogazione record autorizzazione	*vedere la Nota 4*	*vedere la Nota 4*
Interrogazione canale	dsp	n/a
Interroga listener canale	dsp	n/a
Inquire Channel Status (per **ChannelType** MQCHT_CLSSDR)	inq	n/a
Interroga canale di connessione client	dsp	n/a
Interrogazione elenco nomi	dsp	n/a
Interrogazione processo	dsp	n/a
Interrogazione coda	dsp	n/a
Interrogazione gestore code	*vedere la nota 3*	n/a
Interrogazione stato coda	dsp	n/a
Interrogazione servizio	dsp	n/a
Ping canale	ctrl	n/a

Tabella 1. Windows, HP Integrity NonStop Server, UNIX and Linux sistemi - autorizzazioni oggetto (Continua)		
Comando	**Autorizzazione oggetto WebSphere MQ**	**Autorizzazione classe (per tipo di oggetto)**
Ping gestore code	*vedere la nota 3*	n/a
Aggiornamento gestore code	(oggetto gestore code) chg	n/a
Aggiorna sicurezza (per **SecurityType** MQSECTYPE_SSL)	(oggetto gestore code) chg	n/a
Reimposta canale	ctrlx	n/a
Reimpostazione gestore code	(oggetto gestore code) chg	n/a
Reimposta statistiche coda	dsp e chg	n/a
Risolvi canale	ctrlx	n/a
Imposta record di autorizzazione	*(oggetto gestore code)* chg *vedere la nota 4*	*vedere la Nota 4*
Avvio canale	ctrl	n/a
Arresta canale	ctrl	n/a
Arresta connessione	(oggetto gestore code) chg	n/a
Avvia listener	ctrl	n/a
Arresto del listener	ctrl	n/a
Avvia servizio	ctrl	n/a
Arresta servizio	ctrl	n/a
Esc	*vedere la Nota 2*	*vedere la Nota 2*

Note:

1. Questo comando si applica se l'oggetto da sostituire esiste, altrimenti il controllo dell'autorizzazione è quello per la creazione o la copia senza sostituzione.
2. L'autorizzazione richiesta è determinata dal comando MQSC definito dal testo di escape ed è equivalente a uno dei precedenti comandi.
3. Per poter elaborare qualsiasi comando PCF, l'ID utente deve disporre dell'autorizzazione dsp per l'oggetto gestore code sul sistema di destinazione.
4. Questo comando PCF è autorizzato a meno che il server dei comandi non sia stato avviato con il parametro -a. Per impostazione predefinita, il server dei comandi viene avviato all'avvio del gestore code e senza il parametro -a. Per ulteriori informazioni, consultare il manuale System Administration Guide.
5. La concessione dell'autorizzazione *chg* di un ID utente per un gestore code consente di impostare i record di autorizzazione per tutti i gruppi e gli utenti. Non concedere questa autorizzazione agli utenti ordinari o alle applicazioni.

WebSphere MQ fornisce anche alcuni punti di uscita di sicurezza del canale in modo che sia possibile fornire i propri programmi di uscita utente per il controllo di sicurezza. I dettagli sono forniti nel manuale Visualizzazione di un canale .

Utilizzo di MQAI per semplificare l'utilizzo dei PCF

MQAI è un'interfaccia di gestione per WebSphere MQ disponibile sulle piattaforme AIX, HP-UX, IBM i, Linux, Solaris e Windows .

MQAI esegue le attività di amministrazione su un gestore code mediante l'utilizzo di *bag di dati*. I bag di dati consentono di gestire le proprietà (o i parametri) degli oggetti in modo più semplice rispetto all'uso di PCF.

Utilizzare MQAI nei modi seguenti:

Semplificare l'uso dei messaggi PCF
MQAI è un modo semplice per amministrare WebSphere MQ; non è necessario scrivere i propri messaggi PCF, evitando i problemi associati alle strutture dati complesse.

Per passare i parametri nei programmi scritti utilizzando chiamate MQI, il messaggio PCF deve contenere il comando e i dettagli della stringa o i dati interi. Per fare ciò, sono necessarie diverse istruzioni nel programma per ogni struttura e lo spazio di memoria deve essere assegnato. Questo compito può essere lungo e laborioso.

I programmi scritti utilizzando MQAI passano i parametri nel contenitore di dati appropriato ed è necessaria una sola istruzione per ciascuna struttura. L'utilizzo dei contenitori di dati MQAI elimina la necessità di gestire gli array e di assegnare la memoria e fornisce un certo livello di isolamento dai dettagli del PCF.

Per gestire più facilmente le condizioni di errore
È difficile recuperare i codici di ritorno dai comandi PCF, ma MQAI semplifica la gestione delle condizioni di errore da parte del programma.

Una volta creato e popolato il contenitore dati, è possibile inviare un messaggio di comando di gestione al server dei comandi di un gestore code, utilizzando la chiamata `mqExecute` , che attende eventuali messaggi di risposta. La chiamata `mqExecute` gestisce lo scambio con il server dei comandi e restituisce le risposte in un *pacchetto di risposte*.

Per ulteriori informazioni su MQAI, consultare “Introduzione a IBM WebSphere MQ Administration Interface (MQAI)” a pagina 20.

Introduzione a IBM WebSphere MQ Administration Interface (MQAI)

MQAI (IBM WebSphere MQ Administration Interface) è un'interfaccia di programmazione per IBM WebSphere MQ. Esegue le attività di gestione su un gestore code IBM WebSphere MQ utilizzando i contenitori di dati per gestire le proprietà (o i parametri) degli oggetti in modo più semplice rispetto all'utilizzo di PCF (Programmable Command Format).

Concetti e terminologia MQAI

MQAI è un'interfaccia di programmazione per WebSphere MQ, che utilizza il linguaggio C e anche Visual Basic per Windows. È disponibile su piattaforme diverse da z/OS.

Esegue le attività di gestione su un gestore code WebSphere MQ utilizzando i bag di dati. I data bag consentono di gestire le proprietà (o i parametri) degli oggetti in modo più semplice rispetto all'utilizzo dell'altra interfaccia di gestione, PCF (Programmable Command Formats). MQAI offre una manipolazione più semplice dei PCF rispetto all'utilizzo delle chiamate MQGET e MQPUT.

Per ulteriori informazioni sui bag di dati, consultare “Bag di dati” a pagina 46. Per ulteriori informazioni sui PCF, consultare “Introduzione ai formati di comando programmabili” a pagina 9

Utilizzo di MQAI

È possibile utilizzare MQAI per:

- Semplificare l'utilizzo dei messaggi PCF. MQAI è un modo semplice per gestire WebSphere MQ; non è necessario scrivere i propri messaggi PCF e quindi evitare i problemi associati a strutture dati complesse.

- Gestire le condizioni di errore più facilmente. È difficile richiamare i codici di ritorno dai comandi di script WebSphere MQ (MQSC), ma MQAI rende più semplice per il programma gestire le condizioni di errore.
- Scambiare dati tra le applicazioni. I dati dell'applicazione vengono inviati in formato PCF e compressi e decompressi da MQAI. Se i dati del messaggio sono costituiti da numeri interi e stringhe di caratteri, è possibile utilizzare MQAI per sfruttare la conversione dei dati integrata WebSphere MQ per i dati PCF. Ciò evita la necessità di scrivere uscite di conversione dati. Per ulteriori informazioni sull'utilizzo di MQAI per la gestione di WebSphere MQ e per lo scambio di dati tra le applicazioni, consultare “Utilizzo di MQAI per semplificare l'utilizzo dei PCF” a pagina 19.

Esempi di utilizzo di MQAI

L'elenco visualizzato fornisce alcuni programmi di esempio che dimostrano l'utilizzo di MQAI. Gli esempi eseguono le seguenti attività:

1. Creare una coda locale. “Programma C di esempio per la creazione di una coda locale (amqsaicq.c)” a pagina 21
2. Visualizzare gli eventi sullo schermo utilizzando un semplice controllo eventi. “Programma C di esempio per la visualizzazione di eventi utilizzando un controllo eventi (amqsaiem.c)” a pagina 25
3. Stampare un elenco di tutte le code locali e delle relative profondità correnti. “Programma C di esempio per l'interrogazione delle code e delle informazioni di stampa (amqsailq.c)” a pagina 37
4. Stampare un elenco di tutti i canali e i relativi tipi. “Programma C di esempio per l'interrogazione degli oggetti canale (amqsaicl.c)” a pagina 32

Creazione dell'applicazione MQAI

Per creare l'applicazione utilizzando MQAI, collegarsi alle stesse librerie di WebSphere MQ. Per informazioni su come creare le applicazioni WebSphere MQ , vedere Creazione di un' WebSphere MQ .

Suggerimenti e consigli per la configurazione di WebSphere MQ mediante MQAI

MQAI utilizza i messaggi PCF per inviare i comandi di gestione al server dei comandi anziché gestire direttamente il server dei comandi stesso. Suggerimenti per la configurazione di WebSphere MQ utilizzando MQAI sono disponibili in “Suggerimenti e consigli per la configurazione di IBM WebSphere MQ” a pagina 41

MQAI (IBM WebSphere MQ Administration Interface)

IBM WebSphere MQ per Windows, AIX, Linux, HP-UXe Solaris supportano l'interfaccia di amministrazione IBM WebSphere MQ (MQAI). MQAI è un'interfaccia di programmazione per IBM WebSphere MQ che fornisce un'alternativa all'MQI, per l'invio e la ricezione di PCF.

MQAI utilizza i *data bag* che consentono di gestire le proprietà (o i parametri) degli oggetti più facilmente rispetto all'utilizzo diretto dei PCF tramite MQAI.

MQAI fornisce un accesso di programmazione più semplice ai messaggi PCF passando i parametri nella serie di dati, in modo che sia richiesta una sola istruzione per ogni struttura. Questo accesso elimina la necessità per il programmatore di gestire schiere e allocare memoria e fornisce un certo isolamento dai dettagli di PCF.

MQAI gestisce WebSphere MQ inviando messaggi PCF al server dei comandi e attendendo una risposta.

MQAI è descritto nella seconda sezione di questo manuale. Consultare la documentazione Utilizzo di Java per una descrizione di un'interfaccia del modello oggetto componente per MQAI.

Programma C di esempio per la creazione di una coda locale (amqsaicq.c)

Il programma C di esempio `amqsaicq.c` crea una coda locale utilizzando MQAI.

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

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

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

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

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

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

```
     exit(99);
   }

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

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


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

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

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

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


   /*****************************************************************************/
   /* Create a command Bag for the mqExecute call. Exit the function if the     */
   /* create fails.                                                             */
   /*****************************************************************************/
```

```
mqCreateBag(MQCBO_ADMIN_BAG, &commandBag, &compCode, &reason);
CheckCallResult("Create the command bag", compCode, reason);
if (compCode !=MQCC_OK)
   return;

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

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

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

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

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


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

      /**************************************************************************/
      /* Get the completion code and reason code, returned by the command  */
      /* server, from the embedded error bag.                                  */
      /**************************************************************************/
      mqInquireInteger(resultBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                       &compCode, &reason);
      CheckCallResult("Get the completion code from the result bag",
                       compCode, reason);
```

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

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


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

}
```

Programma C di esempio per la visualizzazione di eventi utilizzando un controllo eventi (amqsaiem.c)

Il programma C di esempio amqsaiem.c illustra un controllo eventi di base utilizzando MQAI.

```
*************************************************************************/
/*                                                                       */
/* Program name: AMQSAIEM.C                                              */
/*                                                                       */
/* Description:  Sample C program to demonstrate a basic event monitor   */
/*               using the WebSphere MQ Admin Interface (MQAI).          */
/* Licensed Materials - Property of IBM                                  */
/*                                                                       */
/* 63H9336                                                               */
/* (c) Copyright IBM Corp. 1999, 2024. All Rights Reserved.              */
/*                                                                       */
/* US Government Users Restricted Rights - Use, duplication or           */
/* disclosure restricted by GSA ADP Schedule Contract with               */
/* IBM Corp.                                                             */
/*************************************************************************/
/*                                                                       */
/* Function:                                                             */
/*    AMQSAIEM is a sample C program that demonstrates how to write a simple  */
/*    event monitor using the mqGetBag call and other MQAI calls.        */
/*                                                                       */
```

```
/*    The name of the event queue to be monitored is passed as a parameter    */
/*    to the program. This would usually be one of the system event queues:-  */
/*            SYSTEM.ADMIN.QMGR.EVENT        Queue Manager events             */
/*            SYSTEM.ADMIN.PERFM.EVENT       Performance events               */
/*            SYSTEM.ADMIN.CHANNEL.EVENT     Channel events                   */
/*            SYSTEM.ADMIN.LOGGER.EVENT      Logger events                    */
/*                                                                            */
/*    To monitor the queue manager event queue or the performance event queue,*/
/*    the attributes of the queue manager needs to be changed to enable       */
/*    these events. For more information about this, see Part 1 of the        */
/*    Programmable System Management book. The queue manager attributes can   */
/*    be changed using either MQSC commands or the MQAI interface.            */
/*    Channel events are enabled by default.                                  */
/*                                                                            */
/* Program logic                                                              */
/*    Connect to the Queue Manager.                                           */
/*    Open the requested event queue with a wait interval of 30 seconds.      */
/*    Wait for a message, and when it arrives get the message from the queue  */
/*    and format it into an MQAI bag using the mqGetBag call.                 */
/*    There are many types of event messages and it is beyond the scope of    */
/*    this sample to program for all event messages. Instead the program      */
/*    prints out the contents of the formatted bag.                           */
/*    Loop around to wait for another message until either there is an error  */
/*    or the wait interval of 30 seconds is reached.                          */
/*                                                                            */
/******************************************************************************/
/*                                                                            */
/* AMQSAIEM has 2 parameters - the name of the event queue to be monitored    */
/*                           - the queue manager name (optional)              */
/*                                                                            */
/******************************************************************************

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

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

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

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

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

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

```
   }

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

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

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

   return 0;

}

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

}

/*********************************************************************/
/*                                                                   */
/* Function: GetQEvents                                              */
/*                                                                   */
/*********************************************************************/
/*                                                                   */
/* Input Parameters:  Handle to the queue manager                    */
/*                    Name of the event queue to be monitored        */
/*                                                                   */
/* Output Parameters: None                                           */
/*
/* Logic:   Open the event queue.                                    */
/*          Get a message off the event queue and format the message into     */
/*          a bag.                                                   */
/*          A real event monitor would need to be programmed to deal with     */
/*          each type of event that it receives from the queue. This is       */
/*          outside the scope of this sample, so instead, the contents of     */
/*          the bag are printed.                                     */
/*          The program waits for 30 seconds for an event message and then    */
/*          terminates if no more messages are available.            */
/*                                                                   */
/*********************************************************************/
```

```
void GetQEvents(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG openReason;                       /* MQOPEN reason code             */
   MQLONG reason;                           /* reason code                    */
   MQLONG compCode;                         /* completion code                */
   MQHOBJ eventQueue;                       /* handle to event queue          */

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

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

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

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

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

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

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

        /*****************************************************************************/
        /* If get fails because no message available then we have timed out, */
        /* so report this, otherwise report an error.                        */
        /*****************************************************************************/
        if (reason == MQRC_NO_MSG_AVAILABLE)
        {
           printf("No more messages\n");
        }
        else
        {
           CheckCallResult("Get bag", compCode, reason);
        }
     }

     /*****************************************************************************/
     /* Event message read - Print the contents of the event bag                  */
     /*****************************************************************************/
```

```
     else
     {
       if ( PrintBag(eventBag) )
           printf("\nError found while printing bag contents\n");

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

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

} /* end of GetQEvents */

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

int PrintBag(MQHBAG dataBag)
{
    int errors;

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

    return errors;
}

/*****************************************************************************/
/*                                                                           */
/* Function: PrintBagContents                                                */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Bag Handle                                             */
/*                    Indentation level of bag                               */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Returns:           Number of errors found                                 */
/*                                                                           */
/* Logic: Count the number of items in the bag                               */
/*        Obtain selector and item type for each item in the bag.            */
/*        Obtain the value of the item depending on item type and display the */
/*        index of the item, the selector and the value.                     */
/*        If the item is an embedded bag handle then call this function again */
/*        to print the contents of the embedded bag increasing the           */
/*        indentation level.                                                 */
/*                                                                           */
/*****************************************************************************/
int PrintBagContents(MQHBAG dataBag, int indent)
{
   /*****************************************************************************/
   /* Definitions                                                               */
   /*****************************************************************************/
   #define LENGTH 500                      /* Max length of string to be read*/
```

```
#define INDENT 4                        /* Number of spaces to indent     */
                                        /* embedded bag display           */


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

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

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

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

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

       if (compCode != MQCC_OK)
          errors++;

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

            if (compCode != MQCC_OK)
               errors++;
            else
               printf("%.*s  %-2d     %-4d     (%d)\n",
```

```
                  indent, blanks, i, selector, iValue);
        break

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

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


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

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

     case MQITEM_BYTE_STRING:
        /*****************************************************************/
        /* Item is a byte string. Obtain the byte string in a buffer,  */
        /* prepare the byte string for displaying and display the      */
        /* index, selector and string.                                   */
        /*****************************************************************/
        mqInquireByteString(dataBag,      /* Bag handle                  */
                            MQSEL_ANY_SELECTOR, /* Allow any selector  */
                            i,            /* Index position in the bag */
                            LENGTH,       /* Maximum length of buffer  */
                            byteStringVal, /* Buffer to receive string */
                            &stringLength, /* Actual length of string  */
                            &compCode,    /* Completion code           */
                            &reason);     /* Reason Code

        /*****************************************************************/
        /* The call can return a warning if the string is too long for */
        /* the output buffer and has been truncated, so only check     */
        /* explicitly for call failure.                                  */
```

```
               /*************************************************************/
               if (compCode == MQCC_FAILED)
                   errors++;
               else
               {
                  printf("%.*s  %-2d     %-4d     X'",
                         indent, blanks, i, selector);

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

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

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

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

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

Programma C di esempio per l'interrogazione degli oggetti canale (amqsaicl.c)

Il programma C di esempio amqsaicl.c interroga gli oggetti canale utilizzando MQAI.

```
 /*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAICL.C                                                  */
/*                                                                           */
/* Description:  Sample C program to inquire channel objects                 */
/*               using the WebSphere MQ Administration Interface (MQAI)      */
/*                                                                           */
/* <N_OCO_COPYRIGHT>                                                         */
/* Licensed Materials - Property of IBM                                      */
/*                                                                           */
/* 63H9336                                                                   */
/* (c) Copyright IBM Corp. 2008, 2024. All Rights Reserved.                  */
/*                                                                           */
/* US Government Users Restricted Rights - Use, duplication or               */
/* disclosure restricted by GSA ADP Schedule Contract with                   */
/* IBM Corp.                                                                 */
/* <NOC_COPYRIGHT>                                                           */
/*****************************************************************************/
/*                                                                           */
/* Function:                                                                 */
/*    AMQSAICL is a sample C program that demonstrates how to inquire        */
/*    attributes of the local queue manager using the MQAI interface. In     */
/*    particular, it inquires all channels and their types.                  */
/*                                                                           */
```

```
/*     - A PCF command is built from items placed into an MQAI administration */
/*       bag.                                                                 */
/*       These are:-                                                          */
/*            - The generic channel name "*"                                  */
/*            - The attributes to be inquired. In this sample we just want    */
/*              name and type attributes                                      */
/*                                                                            */
/*     - The mqExecute MQCMD_INQUIRE_CHANNEL call is executed.                */
/*       The call generates the correct PCF structure.                        */
/*       The default options to the call are used so that the command is sent */
/*       to the SYSTEM.ADMIN.COMMAND.QUEUE.                                   */
/*       The reply from the command server is placed on a temporary dynamic   */
/*       queue.                                                               */
/*       The reply from the MQCMD_INQUIRE_CHANNEL is read from the            */
/*       temporary queue and formatted into the response bag.                 */
/*                                                                            */
/*     - The completion code from the mqExecute call is checked and if there  */
/*       is a failure from the command server, then the code returned by the  */
/*       command server is retrieved from the system bag that has been        */
/*       embedded in the response bag to the mqExecute call.                  */
/*                                                                            */
/* Note: The command server must be running.                                  */
/*                                                                            */
/******************************************************************************/
/*                                                                            */
/* AMQSAICL has 2 parameter - the queue manager name (optional)               */
/*                          - output file (optional) default varies           */
/******************************************************************************/

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

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

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

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

/******************************************************************************/
/* Constants                                                                  */
/******************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
const struct
{
  char name[9];
} ChlTypeMap[9] =
{
  "*SDR     ",    /* MQCHT_SENDER    */
  "*SVR     ",    /* MQCHT_SERVER    */
  "*RCVR    ",    /* MQCHT_RECEIVER  */
  "*RQSTR   ",    /* MQCHT_REQUESTER */
  "*ALL     ",    /* MQCHT_ALL       */
  "*CLTCN   ",    /* MQCHT_CLNTCONN  */
  "*SVRCONN ",    /* MQCHT_SVRCONN   */
  "*CLUSRCVR",    /* MQCHT_CLUSRCVR  */
  "*CLUSSDR "     /* MQCHT_CLUSSDR   */
};
#else
const struct
{
  char name[9];
```

```
} ChlTypeMap[9] =
{
  "sdr      ",    /* MQCHT_SENDER    */
  "svr      ",    /* MQCHT_SERVER    */
  "rcvr     ",    /* MQCHT_RECEIVER  */
  "rqstr    ",    /* MQCHT_REQUESTER */
  "all      ",    /* MQCHT_ALL       */
  "cltconn  ",    /* MQCHT_CLNTCONN  */
  "svrcn    ",    /* MQCHT_SVRCONN   */
  "clusrcvr ",    /* MQCHT_CLUSRCVR  */
  "clussdr  "     /* MQCHT_CLUSSDR   */
};
#endif

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

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

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

#endif

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

/********************************************************************************/
/* Function: main                                                               */
/********************************************************************************/
int main(int argc, char *argv[])
{
   /*****************************************************************************/
   /* MQAI variables                                                            */
   /*****************************************************************************/
   MQHCONN hConn;                          /* handle to MQ connection         */
   MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name             */
   MQLONG reason;                          /* reason code                     */
   MQLONG connReason;                      /* MQCONN reason code              */
   MQLONG compCode;                        /* completion code                 */
   MQHBAG adminBag = MQHB_UNUSABLE_HBAG;   /* admin bag for mqExecute         */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute       */
   MQHBAG cAttrsBag;                       /* bag containing chl attributes   */
   MQHBAG errorBag;                        /* bag containing cmd server error */
   MQLONG mqExecuteCC;                     /* mqExecute completion code       */
   MQLONG mqExecuteRC;                     /* mqExecute reason code           */
   MQLONG chlNameLength;                   /* Actual length of chl name       */
   MQLONG chlType;                         /* Channel type                    */
   MQLONG i;                               /* loop counter                    */
   MQLONG numberOfBags;                    /* number of bags in response bag  */
   MQCHAR chlName[MQ_OBJECT_NAME_LENGTH+1];/* name of chl extracted from bag  */
   MQCHAR OutputBuffer[100];               /* output data buffer              */
   OUTFILEHDL *outfp = NULL;               /* output file handle              */

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

   return 0;
}


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

Programma C di esempio per l'interrogazione delle code e delle informazioni di stampa (amqsailq.c)

Il programma C di esempio amqsailq.c interroga la profondità corrente delle code locali utilizzando MQAI.

```
/*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAILQ.C                                                  */
/*                                                                           */
/* Description:  Sample C program to inquire the current depth of the local  */
/*               queues using the WebSphere MQ Administration Interface (MQAI)*/
/*                                                                           */
/* Statement:    Licensed Materials - Property of IBM                        */
/*                                                                           */
/*               84H2000, 5765-B73                                           */
```

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

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

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

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

/****************************************************************************/
/* Function: main                                                           */
/****************************************************************************/
int main(int argc, char *argv[])
{
   /*************************************************************************/
   /* MQAI variables                                                        */
   /*************************************************************************/
   MQHCONN hConn;                       /* handle to WebSphere MQ connection   */
   MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name           */
   MQLONG reason;                       /* reason code                      */
   MQLONG connReason;                   /* MQCONN reason code               */
   MQLONG compCode;                     /* completion code                  */
   MQHBAG adminBag = MQHB_UNUSABLE_HBAG;   /* admin bag for mqExecute       */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute     */
```

```
   MQHBAG qAttrsBag;                       /* bag containing q attributes     */
   MQHBAG errorBag;                        /* bag containing cmd server error */
   MQLONG mqExecuteCC;                     /* mqExecute completion code       */
   MQLONG mqExecuteRC;                     /* mqExecute reason code           */
   MQLONG qNameLength;                     /* Actual length of q name         */
   MQLONG qDepth;                          /* depth of queue                  */
   MQLONG i;                               /* loop counter                    */
   MQLONG numberOfBags;                    /* number of bags in response bag  */
   MQCHAR qName[MQ_Q_NAME_LENGTH+1];       /* name of queue extracted from bag*/


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

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

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

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

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

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

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

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


   /*****************************************************************************/
   /* Check the command server is started. If not exit.                         */
   /*****************************************************************************/
   if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
```

```
{
   printf("Please start the command server: <strmqcsv QMgrName>\n");
   MQDISC(&hConn, &compCode, &reason);
   CheckCallResult("Disconnect from Queue Manager", compCode, reason);
   exit(98);
}

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

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

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

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

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

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

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

   /*************************************************************************/
   /* Get the completion code and reason code, returned by the command     */
   /* server, from the embedded error bag.                                  */
   /*************************************************************************/
   mqInquireInteger(errorBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                    &compCode, &reason );
   CheckCallResult("Get the completion code from the result bag",
                    compCode, reason);
   mqInquireInteger(errorBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                     &compCode, &reason);
   CheckCallResult("Get the reason code from the result bag",
                    compCode, reason);
```

```
      printf("Error returned by the command server: Completion Code = %d :
              Reason = %d\n", mqExecuteCC, mqExecuteRC);
    }
  }

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

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

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

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

Suggerimenti e consigli per la configurazione di IBM WebSphere MQ

Suggerimenti e suggerimenti di programmazione quando si utilizza MQAI.

MQAI utilizza i messaggi PCF per inviare i comandi di gestione al server dei comandi anziché gestire direttamente il server dei comandi stesso. Di seguito sono riportati alcuni suggerimenti per configurare WebSphere MQ utilizzando MQAI:

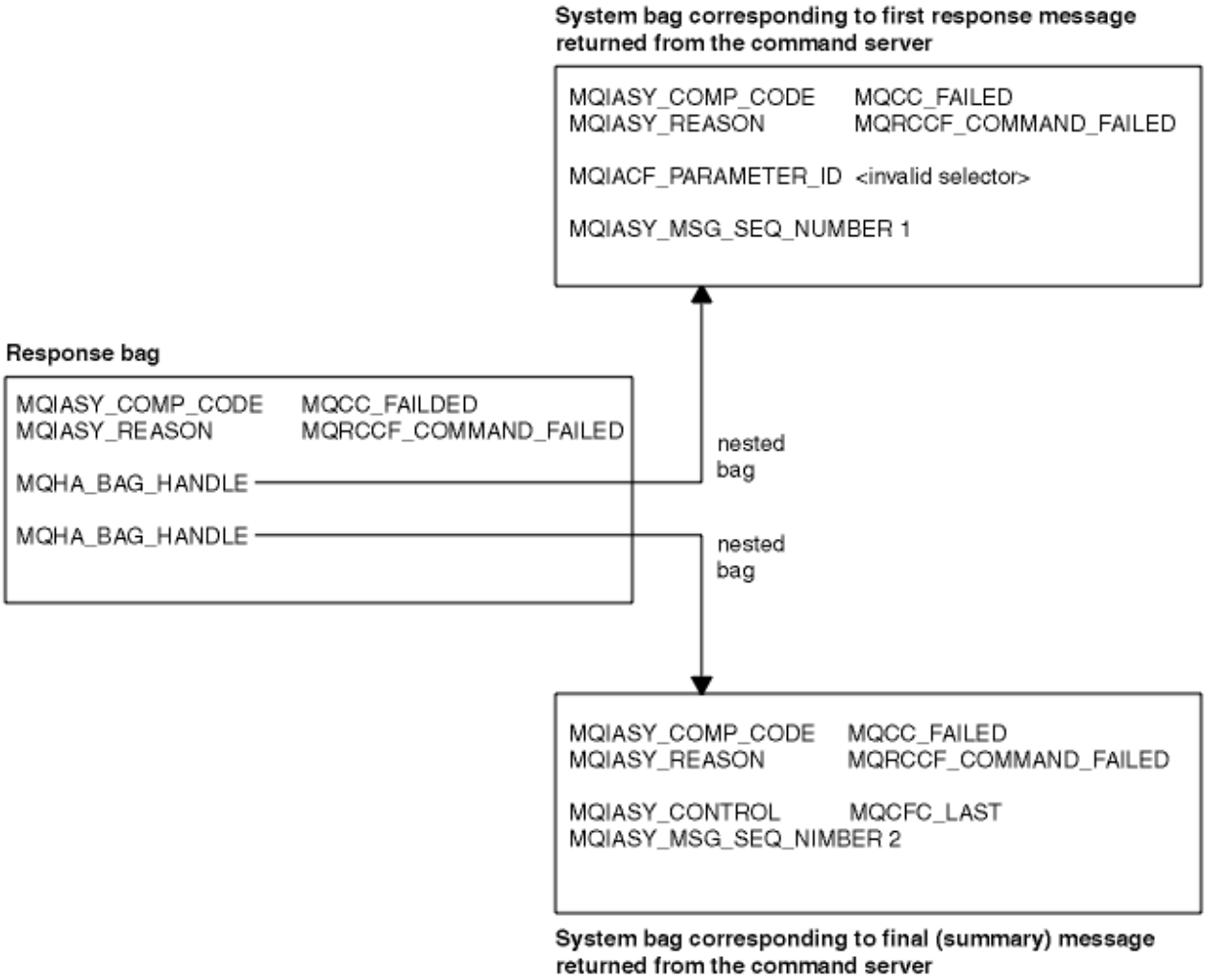
- Le stringhe di caratteri in WebSphere MQ vengono riempite con spazi vuoti fino a una lunghezza fissa. Utilizzando C, le stringhe con terminazione null possono essere normalmente fornite come parametri di input per le interfacce di programmazione WebSphere MQ .
- Per cancellare il valore di un attributo stringa, impostarlo su un singolo spazio piuttosto che su una stringa vuota.
- Considerare in anticipo gli attributi che si desidera modificare e analizzare solo questi attributi.
- Alcuni attributi non possono essere modificati, ad esempio un nome coda o un tipo di canale. Accertarsi di tentare di modificare solo gli attributi che possono essere modificati. Fare riferimento all'elenco di

parametri obbligatori e facoltativi per l'oggetto di modifica PCF specifico. Consultare Definitions of the Programmable Command Formats.

- Se una chiamata MQAI ha esito negativo, alcuni dettagli dell'errore vengono restituiti alla serie di risposte. Ulteriori dettagli possono essere trovati in un contenitore nidificato a cui può accedere il selettore MQHA_BAG_HANDLE. Ad esempio, se una chiamata mqExecute ha esito negativo con un codice di errore MQRCCF_COMMAND_FAILED, queste informazioni vengono restituite nella serie di risposte. Un motivo possibile per questo codice di errore è che un selettore specificato non era valido per il tipo di messaggio di comando e questo dettaglio di informazioni si trova in un contenitore nidificato a cui può accedere un gestore contenitore.

  Per ulteriori informazioni su MQExecute, consultare "Invio di comandi di gestione al server dei comandi utilizzando la chiamata mqExecute" a pagina 56

  Il seguente diagramma mostra questo scenario:



Argomenti MQAI avanzati

Informazioni sull'indicizzazione, sulla conversione dei dati e sull'uso del descrittore del messaggio

- Indicizzazione

  Gli indici vengono utilizzati durante la sostituzione o la rimozione di elementi di dati esistenti da un contenitore per preservare l'ordine di inserimento. I dettagli completi sull'indicizzazione sono disponibili in "Indicizzazione in MQAI" a pagina 43.

- Conversione dati

Le stringhe contenute in un contenitore di dati MQAI possono essere in una serie di caratteri codificati e possono essere convertite utilizzando la chiamata numero intero mqSet. I dettagli completi sulla conversione dei dati sono disponibili in “Conversione dati in MQAI” a pagina 44.

- Utilizzo del descrittore del messaggio

  MQAI genera un descrittore di messaggi impostato su un valore iniziale quando viene creato il contenitore di dati. I dettagli completi sull'utilizzo del descrittore del messaggio sono disponibili in “Utilizzo del descrittore del messaggio in MQAI” a pagina 45.

Indicizzazione in MQAI

Gli indici vengono utilizzati durante la sostituzione o la rimozione di elementi dati esistenti da un contenitore. Esistono tre tipi di indicizzazione, che consentono di richiamare facilmente gli elementi dati.

Ogni selettore e valore in un elemento dati in un contenitore ha tre numeri di indice associati:

- L'indice relativo ad altri elementi che hanno lo stesso selettore.
- L'indice relativo alla categoria del selettore (utente o sistema) a cui appartiene l'elemento.
- L'indice relativo a tutti gli elementi dati nel contenitore (utente e sistema).

Ciò consente l'indicizzazione da parte dei selettori utente, dei selettori di sistema o di entrambi, come mostrato in Figura 1 a pagina 43.

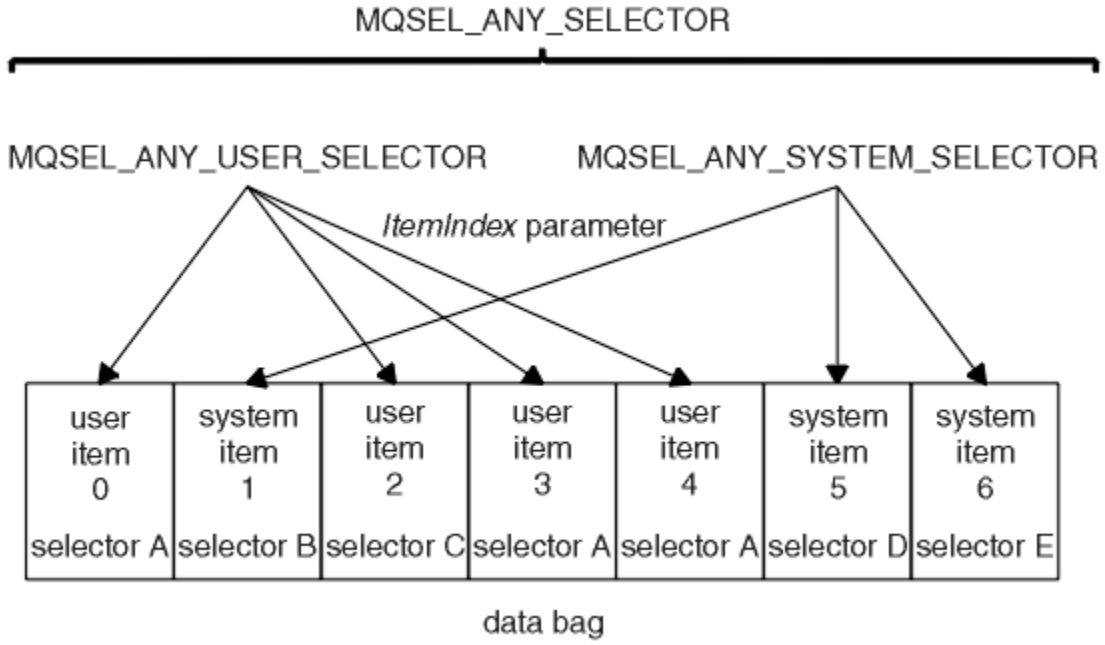


Figura 1. Indicizzazione

Nella Figura Figura 1 a pagina 43, la voce utente 3 (selettore A) può essere indicata dalle seguenti coppie di indici:

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

L'indice è basato su zero come un array in C; se ci sono 'n' ricorrenze, l'indice varia da zero a'n-1', senza interruzioni.

Gli indici vengono utilizzati durante la sostituzione o la rimozione di elementi dati esistenti da un contenitore. Quando viene utilizzato in questo modo, l'ordine di inserimento viene conservato, ma gli

indici di altri elementi dati possono essere influenzati. Per esempi, consultare Modifica delle informazioni all'interno di un contenitore e Eliminazione degli elementi dati.

I tre tipi di indicizzazione consentono un facile richiamo degli elementi dati. Ad esempio, se ci sono tre istanze di un particolare selettore in un contenitore, la chiamata mqCountItems può contare il numero di istanze di quel selettore e le chiamate mqInquire* possono specificare sia il selettore che l'indice per interrogare solo tali valori. Ciò è utile per gli attributi che possono avere un elenco di valori come alcune delle uscite sui canali.

Conversione dati in MQAI

Le stringhe contenute in un contenitore di dati MQAI possono essere in una varietà di serie di caratteri codificati. Queste stringhe possono essere convertite utilizzando la chiamata numero intero mqSet.

Come i messaggi PCF, le stringhe contenute in un contenitore di dati MQAI possono trovarsi in una varietà di serie di caratteri codificati. Di solito, tutte le stringhe in un messaggio PCF si trovano nella stessa serie di caratteri codificati, ovvero la stessa serie del gestore code.

Ogni elemento stringa in un contenitore di dati contiene due valori; la stringa stessa e il CCSID. La stringa che viene aggiunta al contenitore si ottiene dal parametro *Buffer* della chiamata stringa mqAddo mqSet. Il CCSID viene ottenuto dalla voce di sistema contenente un selettore di MQIASY_CODED_CHAR_SET_ID. Questo è noto come *bag CCSID* e può essere modificato utilizzando la chiamata numero intero mqSet.

Quando si interroga il valore di una stringa contenuta in un contenitore di dati, il CCSID è un parametro di emissione dalla chiamata.

Tabella 2 a pagina 44 mostra le regole applicate durante la conversione dei bag di dati in messaggi e viceversa:

Tabella 2. Elaborazione CCSID

Chiamata MQAI	CCSID	Input da richiamare	Output da richiamare
mqBagToBuffer	CCSID serie (1)	Ignorato	Non modificato
mqBagToBuffer	CCSID stringa nel contenitore	Utilizzato	Non modificato
mqBagToBuffer	CCSID stringa nel buffer	Non applicabile	Copiato da CCSID stringa nel contenitore
mqBufferToBag	CCSID serie (1)	Ignorato	Non modificato
mqBufferToBag	CCSID stringa nel buffer	Utilizzato	Non modificato
mqBufferToBag	CCSID stringa nel contenitore	Non applicabile	Copiato da CCSID stringa nel buffer
mqPutSacchetto	CCSID MQMD	Utilizzato	Non modificato (2)
mqPutSacchetto	CCSID serie (1)	Ignorato	Non modificato
mqPutSacchetto	CCSID stringa nel contenitore	Utilizzato	Non modificato
mqPutSacchetto	CCSID stringa nel messaggio inviato	Non applicabile	Copiato da CCSID stringa nel contenitore
mqGetBag	CCSID MQMD	Utilizzato per la conversione dei dati del messaggio	Impostato su CCSID dei dati restituiti (3)
mqGetBag	CCSID serie (1)	Ignorato	Non modificato
mqGetBag	CCSID stringa nel messaggio	Utilizzato	Non modificato

Tabella 2. Elaborazione CCSID (Continua)			
Chiamata MQAI	**CCSID**	**Input da richiamare**	**Output da richiamare**
mqGetBag	CCSID stringa nel contenitore	Non applicabile	Copiato dai CCSID stringa nel messaggio
mqExecute	CCSID della serie di richiesta	Utilizzato per MQMD del messaggio di richiesta (4)	Non modificato
mqExecute	CCSID della serie di risposte	Utilizzato per la conversione dati del messaggio di risposta (4)	Impostato su CCSID dei dati restituiti (3)
mqExecute	CCSID stringa nella serie di richieste	Utilizzato per il messaggio di richiesta	Non modificato
mqExecute	CCSID stringa nel bag di risposta	Non applicabile	Copiato da CCSID stringa nel messaggio di risposta

Note:

1. Bag CCSID è l'elemento di sistema con selettore MQIASY_CODED_CHAR_SET_ID.
2. MQCCSI_Q_MGR è stato modificato nel CCSID del gestore code effettivo.
3. Se viene richiesta la conversione dei dati, il CCSID dei dati restituiti è uguale al valore di emissione. Se la conversione dei dati non è richiesta, il CCSID dei dati restituiti è uguale al valore del messaggio. Notare che non viene restituito alcun messaggio se la conversione dei dati è richiesta ma non riesce.
4. Se il CCSID è MQCCSI_DEFAULT, viene utilizzato il CCSID del gestore code.

Utilizzo del descrittore del messaggio in MQAI

Il descrittore del messaggio generato da MQAI viene impostato su un valore iniziale quando viene creato il contenitore di dati.

Il tipo di comando PCF si ottiene dalla voce di sistema con il selettore MQIASY_TYPE. Quando si crea un contenitore dati, il valore iniziale di questo elemento viene impostato in base al tipo di contenitore creato:

Tabella 3. Tipo di comando PCF	
Tipo di sacchetto	**Valore iniziale dell'elemento MQIASY_TYPE**
BAG MQCBO_ADMIN_	COMANDO MQCFT
BAG MQCBO_COMMAND_	COMANDO MQCFT
MQCBO_ *	UTENTE MQCFT

Quando MQAI genera un descrittore del messaggio, i valori utilizzati nei parametri *Format* e *MsgType* dipendono dal valore dell'elemento di sistema con il selettore MQIASY_TYPE come mostrato in Tabella 3 a pagina 45.

Tabella 4. Formato e parametri MsgType di MQMD		
Tipo di comando PCF	**Formato**	**MsgType**
COMANDO MQCFT	MMQFMT_ADMIN	MQMT_REQUEST
REPORT MQCFT	MMQFMT_ADMIN	REPORT MQMT
MQCF_XX_ENCODE_CASE_ONE risposta	MMQFMT_ADMIN	MQMT_REPLY

Tabella 4. Formato e parametri MsgType di MQMD (Continua)		
Tipo di comando PCF	**Formato**	**MsgType**
MQCF_TRACE_ROUTE	MMQFMT_ADMIN	MQMT_DATAGRAM
EVENTO MQCFT	EVENTO MQFMT	MQMT_DATAGRAM
MQCFT_ *	MQFMT_PCF	MQMT_DATAGRAM

Tabella 4 a pagina 45 mostra che, se si crea un contenitore di gestione o un contenitore di comandi, il *Format* del descrittore del messaggio è MQFMT_ADMIN e il *MsgType* è MQMT_REQUEST. Questo è adatto per un messaggio di richiesta PCF inviato al server dei comandi quando è prevista una risposta.

Altri parametri nel descrittore del messaggio assumono i valori mostrati in Tabella 5 a pagina 46.

Tabella 5. Valori del descrittore del messaggio	
Parametro	**Valore**
StrucId	ID_STRUC_MQMD
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	Consultare Tabella 4 a pagina 45
Expiry	30 secondi (nota “1” a pagina 46)
Feedback	MQFB_NONE
Encoding	MQEN_NATIVE
CodedCharSetId	dipende dal CCSID del contenitore (nota “2” a pagina 46)
Format	Consultare Tabella 4 a pagina 45
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	vedere nota “3” a pagina 46
ReplyToQMgr	vuoto

Note:

1. Questo valore può essere sovrascritto nella chiamata **mqExecute** utilizzando il parametro **OptionsBag** . Per informazioni a riguardo, consultare **mqExecute**.
2. Consultare “Conversione dati in MQAI” a pagina 44.
3. Nome della coda di risposta specificata dall'utente o della coda dinamica temporanea generata da MQAI per i messaggi di tipo **MQMT_REQUEST**. In caso contrario, lasciare vuoto.

Bag di dati

Un contenitore di dati è un mezzo per gestire le proprietà o i parametri degli oggetti che utilizzano MQAI.

Borse dati

- La serie di dati contiene zero o più *elementi dati*. Questi elementi di dati vengono ordinati all'interno del contenitore quando vengono inseriti nel contenitore. Questo è chiamato *ordine di inserimento*. Ogni elemento dati contiene un *selettore* che identifica l'elemento dati e un *valore* di tale elemento dati che può essere un numero intero, un numero intero a 64 bit, un filtro di numeri interi, una stringa, un filtro di stringhe, una stringa di byte, un filtro di stringhe di byte o un handle di un altro contenitore. Gli elementi dati sono descritti in dettaglio in "Elemento di dati" a pagina 49

  Esistono due tipi di selettori: *selettori utente* e *selettori di sistema*. Questi sono descritti in Selettori MQAI. I selettori sono di solito univoci, ma è possibile avere più valori per lo stesso selettore. In questo caso, un *indice* identifica la particolare ricorrenza del selettore richiesta. Gli indici sono descritti in "Indicizzazione in MQAI" a pagina 43.

  Una gerarchia di questi concetti viene mostrata nella Figura 1.

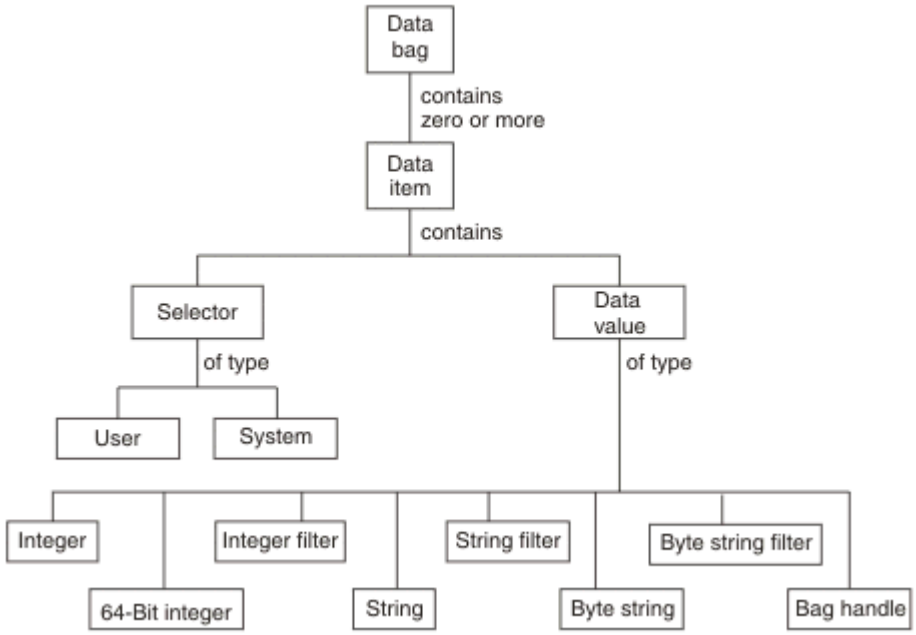


Figura 2. Gerarchia dei concetti MQAI

La gerarchia è stata illustrata in un paragrafo precedente.

Tipi di serie di dati

È possibile scegliere il tipo di contenitore dati che si desidera creare in base all'attività che si desidera eseguire:

serie utente
Un semplice contenitore utilizzato per i dati utente.

serie di gestione
Un contenitore creato per i dati utilizzati per gestire gli oggetti WebSphere MQ inviando messaggi di gestione a un server dei comandi. Il contenitore di amministrazione implica automaticamente alcune opzioni come descritto in "Creazione ed eliminazione di bag di dati" a pagina 48.

serie di comandi
Un contenitore creato anche per i comandi per la gestione degli oggetti WebSphere MQ . Tuttavia, a differenza del contenitore di amministrazione, il contenitore di comandi non implica automaticamente alcune opzioni, sebbene queste opzioni siano disponibili. Per ulteriori informazioni sulle opzioni, consultare "Creazione ed eliminazione di bag di dati" a pagina 48.

gruppo
Un contenitore utilizzato per contenere una serie di elementi dati raggruppati. I gruppi non possono essere utilizzati per la gestione di oggetti WebSphere MQ .

Inoltre, la **serie di sistemi** viene creata da MQAI quando un messaggio di risposta viene restituito dal server dei comandi e inserito in una serie di output dell'utente. Un contenitore di sistema non può essere modificato dall'utente.

I diversi modi di utilizzare i data bag sono elencati in questo argomento:

Utilizzo dei data bag

I diversi modi di utilizzare i data bag sono mostrati nel seguente elenco:

- È possibile creare ed eliminare i data bag "Creazione ed eliminazione di bag di dati" a pagina 48.
- È possibile inviare dati tra le applicazioni utilizzando i bag di dati "Inserimento e ricezione di contenitori di dati" a pagina 49.
- È possibile aggiungere elementi dati ai bag di dati "Aggiunta di elementi dati ai bag" a pagina 50.
- È possibile aggiungere un comando di interrogazione all'interno di un contenitore dati "Aggiunta di un comando di interrogazione ad un contenitore" a pagina 51.
- È possibile richiedere informazioni all'interno dei data bag "Richiesta all'interno dei bag di dati" a pagina 52.
- È possibile contare gli elementi dati all'interno di una serie di dati "Conteggio elementi dati" a pagina 54.
- È possibile modificare le informazioni in una serie di dati "Modifica delle informazioni all'interno di un contenitore" a pagina 52.
- È possibile cancellare un contenitore dati "Cancellazione di un contenitore utilizzando la chiamata di borsa mqClear" a pagina 53.
- È possibile troncare una serie di dati "Troncamento di un bag utilizzando la chiamata al bag mqTruncate" a pagina 53.
- È possibile convertire i bag e i buffer "Conversione di buste e buffer" a pagina 54.

Creazione ed eliminazione di bag di dati

Creazione di bag di dati

Per utilizzare MQAI, è necessario prima creare un contenitore di dati utilizzando la chiamata al contenitore mqCreate. Come input per questa chiamata, fornire una o più opzioni per controllare la creazione della borsa.

Il parametro *Options* della chiamata MQCreateBag consente di scegliere se creare un contenitore utente, un contenitore comandi, un contenitore gruppi o un contenitore di gestione.

Per creare una serie di utenti, una serie di comandi o una serie di gruppi, è possibile scegliere una o più ulteriori opzioni per:

- Utilizzare il modulo di elenco quando ci sono due o più ricorrenze adiacenti dello stesso selettore in un contenitore.
- Riordinare le voci di dati man mano che sono aggiunte a un messaggio PCF per garantire che i parametri siano nell'ordine corretto. Per ulteriori informazioni sugli elementi dati, consultare "Elemento di dati" a pagina 49.
- Controllare i valori dei selettori utente per gli elementi che si aggiungono al contenitore.

Le borse di amministrazione implicano automaticamente queste opzioni.

Una serie di dati viene identificata dal relativo handle. L'handle del contenitore viene restituito dal contenitore mqCreatee deve essere fornito su tutte le altre chiamate che utilizzano il contenitore dati.

Per una descrizione completa della chiamata della borsa mqCreate, consultare mqCreateBag.

Eliminazione dei bag di dati

Qualsiasi contenitore di dati creato dall'utente deve essere eliminato anche utilizzando la chiamata mqDeleteBag. Ad esempio, se un contenitore viene creato nel codice utente, deve essere eliminato anche nel codice utente.

Le borse di sistema vengono create ed eliminate automaticamente da MQAI. Per ulteriori informazioni, consultare “Invio di comandi di gestione al server dei comandi utilizzando la chiamata mqExecute” a pagina 56. Il codice utente non può eliminare un contenitore di sistema.

Per una descrizione completa della chiamata della borsa mqDelete, consultare mqDeleteBag.

Inserimento e ricezione di contenitori di dati

I dati possono anche essere inviati tra le applicazioni inserendo e ottenendo i bag di dati utilizzando le chiamate di borsa mqPute mqGet. Ciò consente a MQAI di gestire il buffer piuttosto che l'applicazione. La chiamata al contenitore mqPutconverte il contenuto del contenitore specificato in un messaggio PCF e invia il messaggio alla coda specificata e la chiamata al contenitore mqGetrimuove il messaggio dalla coda specificata e lo riconverte in un contenitore di dati. Pertanto, la chiamata al sacchetto mqPutè l'equivalente della chiamata mqBagToBuffer seguita da MQPUT e il sacchetto mqGetè l'equivalente della chiamata MQGET seguita da mqBufferToBag.

Per ulteriori informazioni sull'invio e la ricezione di messaggi PCF in una coda specifica, consultare “Invio e ricezione di messaggi PCF in una coda specificata” a pagina 12

Nota: Se si sceglie di utilizzare la chiamata del sacchetto mqGet, i dettagli PCF all'interno del messaggio devono essere corretti; in caso contrario, si verifica un errore appropriato e il messaggio PCF non viene restituito.

Elemento di dati

Gli elementi dati vengono utilizzati per popolare i bag di dati quando vengono creati. Questi elementi dati possono essere elementi utente o di sistema.

Questi elementi utente contengono dati utente, ad esempio attributi di oggetti gestiti. Gli elementi di sistema devono essere utilizzati per un maggiore controllo sui messaggi generati: ad esempio, la creazione di intestazioni di messaggi. Per ulteriori informazioni sugli item di sistema, consultare “Elementi di sistema” a pagina 50.

Tipi di elementi dati

Una volta creato un contenitore di dati, è possibile popolarlo con elementi interi o stringa di caratteri. È possibile richiedere informazioni su tutti e tre i tipi di elemento.

L'elemento dati può essere un numero intero o un elemento stringa di caratteri. Di seguito sono riportati i tipi di elementi dati disponibili all'interno di MQAI:

- Intero
- Intero a 64 bit
- Filtro numero intero
- Stringa di caratteri
- Filtro stringa
- Stringa byte
- Filtro stringa di byte
- Maniglia del sacchetto

Utilizzo di elementi dati

Questi sono i seguenti modi di utilizzare gli elementi dati:

- “Conteggio elementi dati” a pagina 54.
- “Eliminazione di elementi dati” a pagina 54.
- “Aggiunta di elementi dati ai bag” a pagina 50.
- “Filtro e query degli elementi dati” a pagina 51.

Elementi di sistema

Gli item di sistema possono essere utilizzati per:

- La creazione di intestazione PCF. Le voci di sistema possono controllare l'identificativo del comando PCF, le opzioni di controllo, il numero di sequenza del messaggio e il tipo di comando.
- Conversione dati. Le voci del sistema gestiscono l'identificativo della serie di caratteri per le voci della stringa di caratteri nel contenitore.

Come tutti gli elementi di dati, gli elementi di sistema sono costituiti da un selettore e un valore. Per informazioni su questi selettori e sulla loro funzione, consultare Selettori MQAI.

Gli elementi di sistema sono univoci. Uno o più elementi di sistema possono essere identificati da un selettore di sistema. C'è solo una ricorrenza di ogni selettore di sistema.

La maggior parte degli elementi del sistema possono essere modificati (consultare “Modifica delle informazioni all'interno di un contenitore” a pagina 52), ma le opzioni di creazione del contenitore non possono essere modificate dall'utente. Non è possibile eliminare elementi di sistema. (Consultare “Eliminazione di elementi dati” a pagina 54.)

Aggiunta di elementi dati ai bag

Quando viene creata una serie di dati, è possibile popolarla con elementi dati. Questi elementi dati possono essere elementi utente o di sistema. Per ulteriori informazioni sugli elementi dati, consultare “Elemento di dati” a pagina 49.

MQAI consente di aggiungere elementi interi, elementi interi a 64 bit, elementi di filtro interi, elementi stringa di caratteri, filtri stringa, elementi stringa di byte e elementi filtro stringa di byte ai bag, come mostrato in Figura 3 a pagina 50. Gli elementi sono identificati da un selettore. Di solito un selettore identifica solo un elemento, ma non sempre è così. Se un elemento dati con il selettore specificato è già presente nel contenitore, un'ulteriore istanza di tale selettore viene aggiunta alla fine del contenitore.

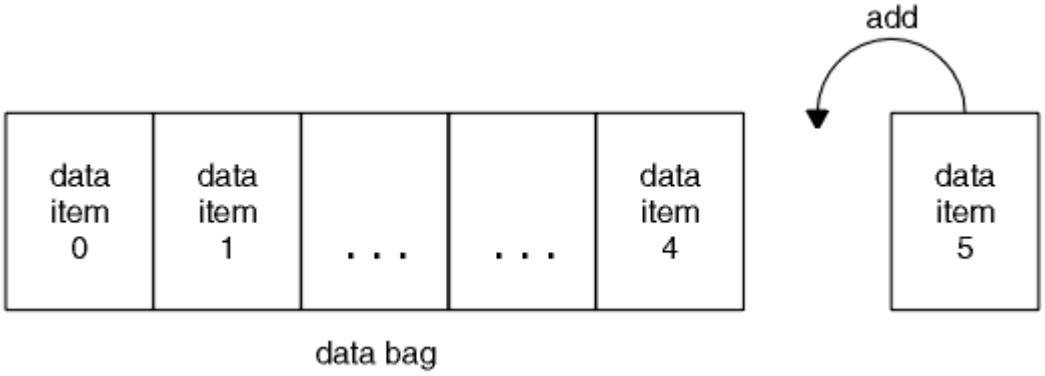


Figura 3. aggiunte degli elementi dati

Aggiungere elementi dati a un contenitore utilizzando le chiamate mqAdd*:

- Per aggiungere elementi interi, utilizzare la chiamata mqAddInteger come descritto in mqAddInteger
- Per aggiungere elementi interi a 64 bit, utilizzare la chiamata mqAddInteger64 come descritto in mqAddInteger64

- Per aggiungere elementi di filtro interi, utilizzare la chiamata mqAddIntegerFilter come descritto in mqAddIntegerFilter
- Per aggiungere elementi stringa di caratteri, utilizzare la chiamata stringa mqAddcome descritto in mqAddString
- Per aggiungere elementi filtro stringa, utilizzare la chiamata mqAddStringFilter come descritto in mqAddStringFilter
- Per aggiungere elementi stringa di byte, utilizzare la chiamata mqAddByteString come descritto in mqAddByteString
- Per aggiungere elementi filtro stringa di byte, utilizzare la chiamata filtro mqAddByteStringcome descritto in mqAddByteStringFilter

Per ulteriori informazioni sull'aggiunta di elementi dati a un contenitore, consultare “Elementi di sistema” a pagina 50.

Aggiunta di un comando di interrogazione ad un contenitore

La chiamata di interrogazione mqAddviene utilizzata per aggiungere un comando di interrogazione ad un contenitore. La chiamata è specificamente per scopi di amministrazione, quindi può essere utilizzata solo con sacchetti di amministrazione. Consente di specificare i selettori di attributi su cui si desidera eseguire l'interrogazione da WebSphere MQ.

Per una descrizione completa della chiamata di interrogazione mqAdd, consultare mqAddInquiry.

Filtro e query degli elementi dati

Quando si utilizza MQAI per richiedere informazioni sugli attributi di oggetti WebSphere MQ , è possibile controllare i dati restituiti al programma in due modi.

- È possibile ***filtrare*** i dati restituiti utilizzando le chiamate mqAddInteger e mqAddString. Questo approccio consente di specificare una coppia *Selector* e *ItemValue* , ad esempio:

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

  Questo esempio specifica che il tipo di coda (*Selector*) deve essere locale (*ItemValue*) e che questa specifica deve essere corrispondente agli attributi dell'oggetto (in questo caso, una coda) che si sta analizzando.

  Altri attributi che possono essere filtrati corrispondono ai comandi PCF Inquire * che possono essere trovati in “Introduzione ai formati di comando programmabili” a pagina 9. Ad esempio, per informazioni sugli attributi di un canale, consultare il comando Interroga canale nella documentazione di questo prodotto. I "Parametri obbligatori" e i "Parametri facoltativi" del comando Inquire Channel identificano i selettori che è possibile utilizzare per il filtro.
- È possibile ***interrogare*** particolari attributi di un oggetto utilizzando la chiamata di interrogazione mqAdd. Specifica il selettore a cui si è interessati. Se non si specifica il selettore, vengono restituiti tutti gli attributi dell'oggetto.

Di seguito viene riportato un esempio di filtro ed esecuzione di query degli attributi di una coda:

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

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

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

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

Per ulteriori esempi di filtro e query degli elementi dati, consultare “Esempi di utilizzo di MQAI” a pagina 21.

Richiesta all'interno dei bag di dati

È possibile richiedere informazioni su:

- Il valore di un elemento intero che utilizza la chiamata numero intero mqInquire. Vedere mqInquireInteger.
- Il valore di un elemento intero a 64 bit utilizzando la chiamata mqInquireInteger64 . Vedere mqInquireInteger64.
- Il valore di un elemento filtro intero che utilizza la chiamata mqInquireIntegerFilter . Vedere mqInquireIntegerFilter.
- Il valore di un elemento stringa di caratteri che utilizza la chiamata stringa mqInquire. Vedere mqInquiremqInquire.
- Il valore di un elemento filtro stringa utilizzando la chiamata mqInquireStringFilter . Vedere mqInquireStringFilter.
- Il valore di un elemento stringa di byte utilizzando la chiamata mqInquireByteString . Vedere mqInquireByteString.
- Il valore di un elemento filtro stringa di byte utilizzando la chiamata Filtro mqInquireByteString. Consultare mqInquireByteStringFilter.
- Il valore di un handle di borsa utilizzando la chiamata di borsa mqInquire. Vedere mqInquireBag.

È anche possibile richiedere informazioni sul tipo (integer, 64 - bit integer, integer filter, character string, string filter, byte string, byte string filter o bag handle) di un elemento specifico utilizzando la chiamata mqInquireItemInfo . Vedere mqInquireItemInfo.

Modifica delle informazioni all'interno di un contenitore

MQAI consente di modificare le informazioni in un contenitore utilizzando le chiamate mqSet*. È possibile:

1. Modificare gli elementi dati all'interno di un contenitore. L'indice consente di sostituire una singola istanza di un parametro identificando la ricorrenza dell'elemento da modificare (consultare Figura 4 a pagina 52).

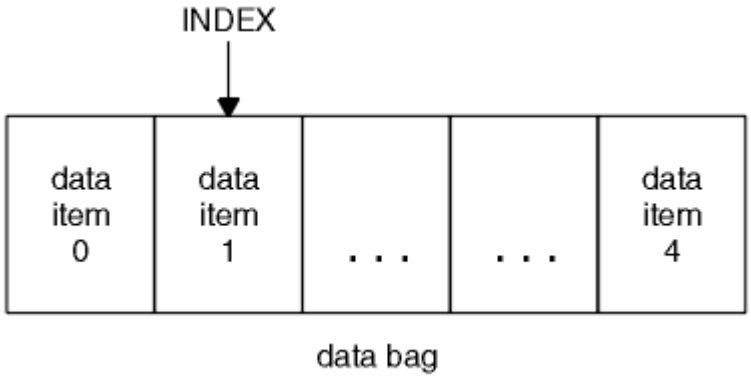


   Figura 4. Modifica di un singolo elemento dati

2. Eliminare tutte le ricorrenze esistenti del selettore specificato e aggiungere una nuova ricorrenza alla fine del contenitore. (Consultare Figura 5 a pagina 53.) Un valore di indice speciale consente di sostituire ***tutte le*** istanze di un parametro.

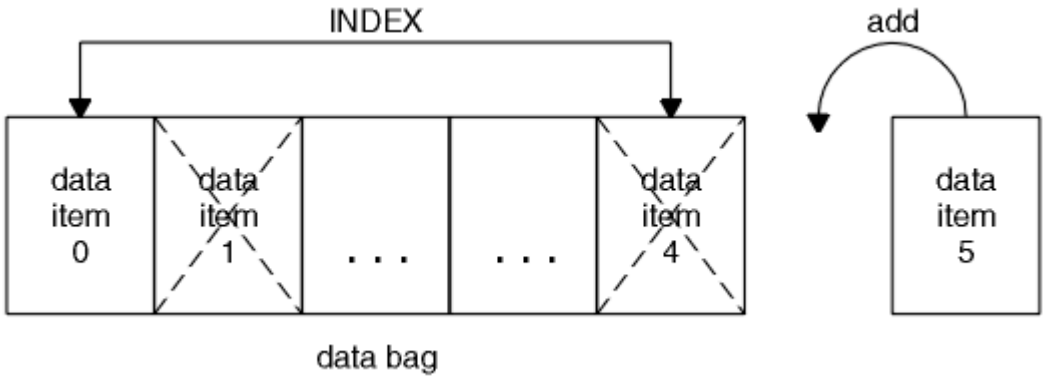


Figura 5. Modifica di tutti gli elementi dati

Nota: L'indice conserva l'ordine di inserimento all'interno del contenitore ma può influire sugli indici di altri elementi di dati.

La chiamata numero intero mqSetconsente di modificare gli elementi interi all'interno di un contenitore. La chiamata mqSetInteger64 consente di modificare elementi interi a 64 bit. La chiamata mqSetIntegerFilter consente di modificare elementi di filtro interi. La chiamata stringa mqSetconsente di modificare gli elementi stringa di caratteri. La chiamata mqSetStringFilter consente di modificare gli elementi filtro stringa. La chiamata mqSetByteString consente di modificare gli elementi della stringa di byte. La chiamata del filtro mqSetByteStringconsente di modificare gli elementi filtro della stringa di byte. In alternativa, è possibile utilizzare queste chiamate per eliminare tutte le ricorrenze esistenti del selettore specificato e aggiungere una nuova ricorrenza alla fine del contenitore. L'elemento dati può essere un elemento utente o un elemento di sistema.

Per una descrizione completa di queste chiamate, consultare:

- mqSetNumero intero
- mqSetInteger64
- mqSetIntegerFilter
- StringamqSet
- mqSetStringFilter
- mqSetByteString
- mqSetByteStringByteString

Cancellazione di un contenitore utilizzando la chiamata di borsa mqClear

La chiamata mqClearBag rimuove tutti gli elementi utente da una serie di utenti e reimposta gli elementi di sistema sui valori iniziali. Vengono eliminati anche i sacchetti di sistema contenuti all'interno del sacchetto.

Per una descrizione completa della chiamata della borsa mqClear, consultare mqClearBag.

Troncamento di un bag utilizzando la chiamata al bag mqTruncate

La chiamata mqTruncateriduce il numero di elementi utente in un contenitore utente eliminando gli elementi dalla fine del contenitore, a partire dall'elemento aggiunto più di recente. Ad esempio, può essere utilizzato quando si utilizzano le stesse informazioni di intestazione per generare più di un messaggio.

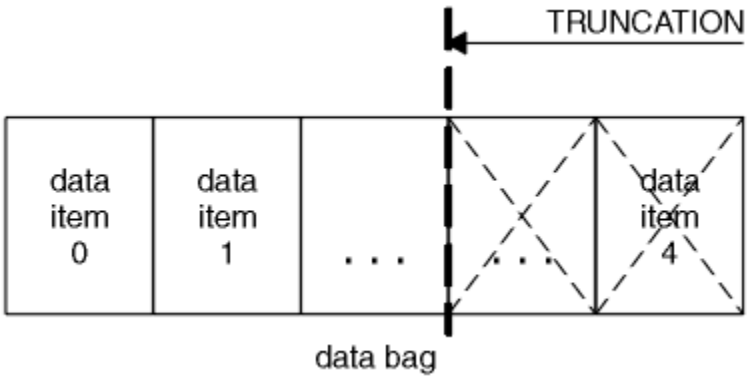


Figura 6. Troncamento di un sacchetto

Per una descrizione completa della chiamata Bag mqTruncate, consultare mqTruncateBag.

Conversione di buste e buffer

Per inviare i dati tra le applicazioni, innanzitutto i dati del messaggio vengono inseriti in un contenitore. Quindi, i dati nel contenitore vengono convertiti in un messaggio PCF utilizzando la chiamata mqBagToBuffer . Il messaggio PCF viene inviato alla coda richiesta utilizzando la chiamata MQPUT. Ciò viene mostrato nella figura Figura 7 a pagina 54. Per una descrizione completa della chiamata mqBagToBuffer , consultare mqBagToBuffer.

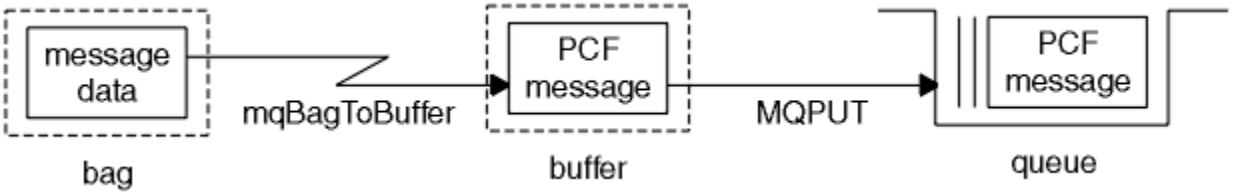


Figura 7. Conversione di sacchetti in messaggi PCF

Per ricevere i dati, il messaggio viene ricevuto in un buffer utilizzando la chiamata MQGET. I dati nel buffer vengono quindi convertiti in un contenitore utilizzando la chiamataToBag di mqBuffer, purché il buffer contenga un messaggio PCF valido. Ciò è mostrato in Figura Figura 8 a pagina 54. Per una descrizione completa della chiamata mqBufferToBag , vedere mqBufferToBag.

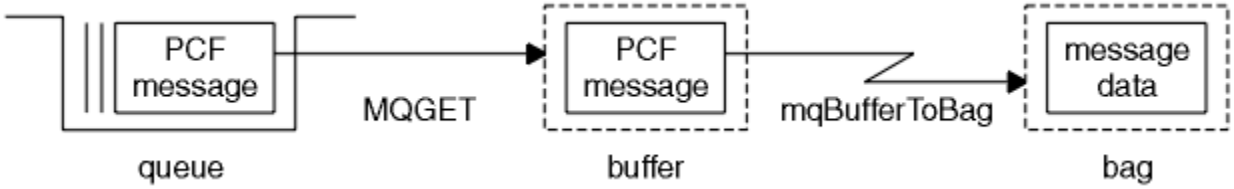


Figura 8. Conversione dei messaggi PCF in formato bag

Conteggio elementi dati

La chiamata mqCountelementi conta il numero di elementi utente, elementi di sistema o entrambi, memorizzati in un contenitore di dati e restituisce questo numero. Ad esempio, `mqCountItems(`*Bag*`, 7, ...)`, restituisce il numero di elementi nella borsa con un selettore di 7. Può contare gli elementi per singolo selettore, per selettori utente, per selettori di sistema o per tutti i selettori.

Nota: Questa chiamata conta il numero di elementi dati, non il numero di selettori univoci nel contenitore. Un selettore può verificarsi più volte, quindi potrebbero esserci meno selettori univoci nel contenitore rispetto agli elementi di dati.

Per una descrizione completa della chiamata mqCountItem, vedere mqCountItems.

Eliminazione di elementi dati

È possibile eliminare gli articoli dalle borse in diversi modi. È possibile:

- Rimuovere uno o più elementi utente da un contenitore. Per informazioni dettagliate, consultare “Eliminazione di elementi dati da un contenitore utilizzando la chiamata di elemento mqDelete” a pagina 55.
- Eliminare ***tutto*** elementi utente da un contenitore, ossia *cancella* un contenitore. Per informazioni dettagliate, consultare “Cancellazione di un contenitore utilizzando la chiamata di borsa mqClear” a pagina 53.
- Eliminare gli elementi utente dalla fine di un contenitore, ossia *troncare* un contenitore. Per informazioni dettagliate, consultare “Troncamento di un bag utilizzando la chiamata al bag mqTruncate” a pagina 53.

Eliminazione di elementi dati da un contenitore utilizzando la chiamata di elemento mqDelete

La chiamata di elemento mqDeleterimuove uno o più elementi utente da un contenitore. L'indice viene utilizzato per eliminare:

1. Una singola ricorrenza del selettore specificato. (Consultare Figura 9 a pagina 55.)

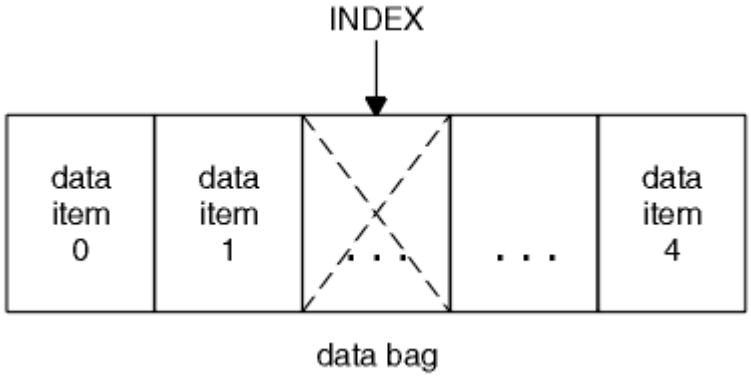


Figura 9. Eliminazione di un singolo elemento dati

oppure

2. Tutte le ricorrenze del selettore specificato. (Consultare Figura 10 a pagina 55.)

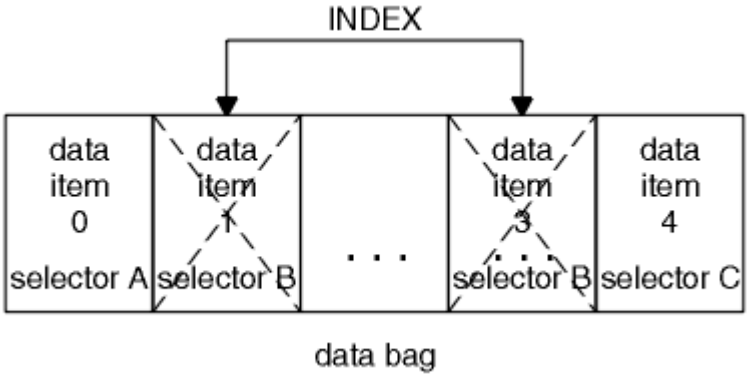


Figura 10. Eliminazione di tutti gli elementi dati

Nota: L'indice conserva l'ordine di inserimento all'interno del contenitore ma può influire sugli indici di altri elementi di dati. Ad esempio, la chiamata dell'elemento mqDeletenon conserva i valori di indice degli elementi dati che seguono l'elemento eliminato perché gli indici sono riorganizzati per colmare il divario che rimane dall'elemento eliminato.

Per una descrizione completa della chiamata mqDeleteItem, consultare mqDeleteItem.

Invio di comandi di gestione al server dei comandi utilizzando la chiamata mqExecute

Quando una serie di dati è stata creata e popolata, è possibile inviare un messaggio di comando di gestione al server dei comandi di un gestore code utilizzando la chiamata mqExecute . Questa operazione gestisce lo scambio con il server dei comandi e restituisce le risposte in un contenitore.

Una volta creato e popolato il contenitore dati, è possibile inviare un messaggio di comando di gestione al server dei comandi di un gestore code. Il modo più semplice per eseguire questa operazione è utilizzando la chiamata mqExecute . La chiamata mqExecute invia un messaggio di comando di gestione come messaggio non persistente e attende eventuali risposte. Le risposte vengono restituite in un contenitore di risposte. Potrebbero contenere informazioni relative agli attributi relativi a diversi oggetti WebSphere MQ o a una serie di messaggi di risposta di errore PCF, ad esempio. Pertanto, il contenitore della risposta potrebbe contenere solo un codice di ritorno o potrebbe contenere *contenitori nidificati*.

I messaggi di risposta vengono inseriti nei bag di sistema creati dal sistema. Ad esempio, per richieste relative ai nomi degli oggetti, viene creata una serie di sistemi per contenere tali nomi oggetto e la serie viene inserita nella serie utente. Le maniglie di queste borse vengono quindi inserite nel sacchetto di risposta e al sacchetto nidificato può accedere il selettore MQHA_BAG_HANDLE. Il contenitore di sistema rimane nella memoria, se non viene eliminato, fino a quando il contenitore di risposta non viene eliminato.

Il concetto di *nidificazione* viene mostrato in Figura 11 a pagina 56.

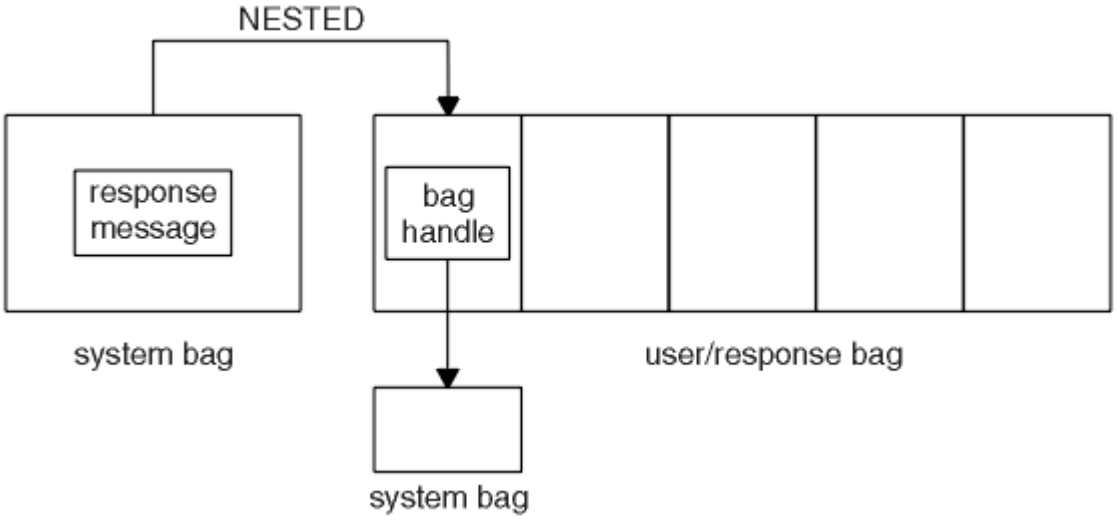


Figura 11. Nidificazione

Come input per la chiamata mqExecute , è necessario fornire:

- Un handle di connessione MQI.
- Il comando da eseguire. Deve essere uno dei valori MQCMD_ *.

  Nota: Se questo valore non viene riconosciuto da MQAI, viene comunque accettato. Tuttavia, se la chiamata mqAddInquiry è stata utilizzata per inserire i valori nel contenitore, questo parametro deve essere un comando INQUIRE riconosciuto da MQAI. Ovvero, il parametro deve essere nel formato MQCMD_INQUIRE_ *.
- Facoltativamente, un handle del contenitore contenente le opzioni che controllano l'elaborazione della chiamata. Questo è anche il punto in cui è possibile specificare il tempo massimo in millisecondi che MQAI deve attendere per ogni messaggio di risposta.
- Un handle del contenitore di gestione che contiene i dettagli del comando di gestione da immettere.
- Un handle del contenitore risposte che riceve i messaggi di risposta.

I seguenti sono facoltativi:

- Un handle di oggetto della coda in cui deve essere inserito il comando di gestione.

Se non viene specificato alcun handle di oggetto, il comando di gestione viene posizionato su SYSTEM.ADMIN.COMMAND.QUEUE appartenente al Gestore code attualmente connesso. Questa è l'opzione predefinita.

- Una gestione oggetto della coda in cui devono essere inseriti i messaggi di risposta.

  È possibile scegliere di inserire i messaggi di risposta in una coda dinamica creata automaticamente da MQAI. La coda creata esiste solo per la durata della chiamata e viene eliminata dall'MQAI all'uscita dalla chiamata mqExecute .

Per esempi di utilizzo della chiamata mqExecute , consultare Codice di esempio

Amministrazione mediante IBM WebSphere MQ Explorer

IBM WebSphere MQ Explorer consente di eseguire la gestione locale o remota della rete solo da un computer su cui è in esecuzione Windowso Linux (piattaformex86 e x86-64).

IBM WebSphere MQ per Windowse IBM WebSphere MQ per Linux (piattaformex86 e x86-64) forniscono un'interfaccia di gestione denominata IBM WebSphere MQ Explorer per eseguire le attività di gestione come alternativa all'utilizzo dei comandi di controllo o MQSC. Confronto di serie di comandi mostra le operazioni che è possibile eseguire utilizzando IBM WebSphere MQ Explorer.

IBM WebSphere MQ Explorer consente di eseguire la gestione locale o remota della propria rete da un computer su cui è in esecuzione Windowso Linux (piattaformex86-64), puntando il IBM WebSphere MQ Explorer sui gestori code e sui cluster a cui si è interessati. Le piattaforme e i livelli di IBM WebSphere MQ che è possibile gestire utilizzando IBM WebSphere MQ Explorer sono descritti in “Gestori code remoti” a pagina 58.

Per configurare i gestori code IBM WebSphere MQ remoti in maniera che IBM WebSphere MQ Explorer possa gestirli, consultare “Software prerequisito e definizioni” a pagina 59.

Consente di eseguire attività, di solito associate all'impostazione e all'ottimizzazione dell'ambiente di lavoro per IBM WebSphere MQ, localmente o in remoto all'interno di un dominio di sistema Windows o Linux (piattaformex86 e x86-64).

Su Linux, l'avvio di IBM WebSphere MQ Explorer potrebbe non riuscire se si dispone di più di un'installazione di Eclipse . Se ciò si verifica, avviare IBM WebSphere MQ Explorer utilizzando un ID utente diverso da quello utilizzato per l'altra installazione di Eclipse .

Su Linux, per avviare correttamente IBM WebSphere MQ Explorer , è necessario essere in grado di scrivere un file nella propria directory home e la directory home deve esistere.

Operazioni che è possibile eseguire con IBM WebSphere MQ Explorer

Questo è un elenco delle attività che è possibile eseguire utilizzando IBM WebSphere MQ Explorer.

Con IBM WebSphere MQ Explorer, è possibile:

- Creare ed eliminare un gestore code (solo sulla macchina locale).
- Avviare e arrestare un gestore code (solo sulla macchina locale).
- Definire, visualizzare e modificare le definizioni degli oggetti di WebSphere MQ come code e canali.
- Esaminare i messaggi su una coda.
- Avviare e arrestare un canale.
- Visualizzare le informazioni sullo stato di un canale, listener, coda o oggetti di servizio.
- Visualizzare i gestori code in un cluster.
- Verificare quali applicazioni, utenti o canali hanno una particolare coda aperta.
- Creare un nuovo cluster di gestori code utilizzando la procedura guidata *Crea nuovo cluster* .
- Aggiungere un gestore code a un cluster utilizzando il wizard *Aggiungi gestore code al cluster* .
- Gestire l'oggetto delle informazioni di autenticazione, utilizzato con la sicurezza del canale SSL (Secure Sockets Layer).

- Creare ed eliminare gli iniziatori di canali, i controlli dei trigger e i listener.
- Avviare o arrestare i server dei comandi, gli iniziatori dei canali, i controlli dei trigger e i listener.
- Impostare i servizi specifici in modo che vengano avviati automaticamente all'avvio di un gestore code.
- Modificare le proprietà dei gestori code.
- Modificare il gestore code predefinito locale.
- Richiamare la GUI ikeyman per gestire i certificati SSL (secure sockets layer), associare i certificati ai gestori code e configurare e configurare gli archivi di certificati (solo sulla macchina locale).
- Creare oggetti JMS da oggetti WebSphere MQ e oggetti WebSphere MQ da oggetti JMS.
- Creare un factory di connessione JMS per uno dei tipi attualmente supportati.
- Modificare i parametri per qualsiasi servizio, come il numero di porta TCP per un listener o il nome della coda dell'iniziatore di canali.
- Avviare o arrestare la traccia del servizio.

Le attività di gestione vengono eseguite utilizzando una serie di *Viste contenuto* e *Finestra di dialogo Proprietà*.

Vista Contenuto
Una vista Contenuto è un pannello che può visualizzare quanto segue:

- Attributi e opzioni di gestione relativi a WebSphere MQ .
- Attributi e opzioni di gestione relativi a uno o più oggetti correlati.
- Attributi e opzioni di amministrazione per un cluster.

Finestre delle proprietà
Una finestra di dialogo delle proprietà è un pannello che visualizza gli attributi relativi ad un oggetto in una serie di campi, alcuni dei quali è possibile modificare.

Si naviga attraverso WebSphere MQ Explorer utilizzando la *vistaNavigator*. Il Navigator consente di selezionare la vista Contenuto richiesta.

Gestori code remoti

Esistono due eccezioni ai gestori code supportati a cui è possibile connettersi.

Da un sistema Windows o Linux (piattaformex86 e x86-64), WebSphere MQ Explorer può connettersi a tutti i gestori code supportati con le seguenti eccezioni:

- WebSphere MQ per z/OS gestori code precedenti alla versione 6.0.
- Gestori code MQSeries V2 attualmente supportati.

IBM WebSphere MQ Explorer gestisce le differenze nelle capacità tra i diversi livelli di comando e piattaforme. Tuttavia, se rileva un attributo che non riconosce, l'attributo non sarà visibile.

Se si intende amministrare in remoto un gestore code V6.0 o successivo su Windows utilizzando IBM WebSphere MQ Explorer su un computer WebSphere MQ V5.3 , è necessario installare il Fix Pack 9 (CSD9) o successivo sul computer WebSphere MQ per Windows V5.3 .

Se si intende gestire in remoto un gestore code V5.3 su iSeries utilizzando WebSphere MQ Explorer su un computer WebSphere MQ V6.0 o successivo, è necessario installare il Fix Pack 11 (CSD11) o successivo sul computer WebSphere MQ per iSeries V5.3 . Questo fix pack corregge problemi di collegamento tra WebSphere MQ Explorer e il gestore code iSeries .

Decidere se utilizzare il IBM WebSphere MQ Explorer

Quando si decide se utilizzare IBM WebSphere MQ Explorer durante l'installazione, considerare le informazioni elencate in questo argomento.

È necessario essere consapevoli dei seguenti punti:

Nomi oggetto
Se si utilizzano i nomi in minuscolo per i gestori code e altri oggetti con IBM WebSphere MQ Explorer, quando si utilizzano gli oggetti utilizzando i comandi MQSC, è necessario racchiudere i nomi oggetto tra virgolette singole oppure WebSphere MQ non li riconosce.

Gestori code di grandi dimensioni
IBM WebSphere MQ Explorer funziona meglio con piccoli gestori code. Se si dispone di un numero elevato di oggetti su un singolo gestore code, si potrebbero verificare dei ritardi mentre WebSphere MQ Explorer estrae le informazioni richieste da presentare in una vista.

Cluster
WebSphere I cluster MQ possono contenere potenzialmente centinaia o migliaia di gestori code. WebSphere MQ Explorer presenta i gestori code in un cluster utilizzando una struttura ad albero. La dimensione fisica di un cluster non influisce in modo significativo sulla velocità di Esplora risorse di IBM WebSphere MQ poiché Esplora risorse di IBM WebSphere MQ non si connette ai gestori code nel cluster finché non vengono selezionati.

Impostazione di IBM WebSphere MQ Explorer

Questa sezione illustra i passi che è necessario eseguire per configurare IBM WebSphere MQ Explorer.

- "Software prerequisito e definizioni" a pagina 59
- "Sicurezza" a pagina 59
- "Visualizzazione e nascondimento di gestori code e cluster" a pagina 63
- "Appartenenza cluster" a pagina 64
- "Conversione dati" a pagina 64

Software prerequisito e definizioni

Assicurarsi di soddisfare i seguenti requisiti prima di provare a utilizzare IBM WebSphere MQ Explorer.

IBM WebSphere MQ Explorer può connettersi ai gestori code remoti utilizzando solo il protocollo di comunicazione TCP/IP.

Verificare che:

1. Un server dei comandi è in esecuzione su ogni gestore code gestito in remoto.
2. Un oggetto listener TCP/IP adatto deve essere in esecuzione su ogni gestore code remoto. Questo oggetto può essere il listener IBM WebSphere MQ o, sui sistemi UNIX and Linux , il daemon inetd.
3. Un canale di connessione server, per impostazione predefinita denominato SYSTEM.ADMIN.SVRCONN, esiste su tutti i gestori code remoti.

   È possibile creare il canale utilizzando il seguente comando MQSC:

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

   Questo comando crea una definizione di canale di base. Se si desidera una definizione più sofisticata (per impostare la sicurezza, ad esempio), sono necessari ulteriori parametri. Per ulteriori informazioni, vedere DEFINE CHANNEL .
4. La coda di sistema, SYSTEM.MQEXPLORER.REPLY.MODEL, deve esistere.

Sicurezza

Se si sta utilizzando WebSphere MQ in un ambiente in cui è importante controllare l'accesso dell'utente a determinati oggetti, potrebbe essere necessario considerare gli aspetti di sicurezza dell'utilizzo di IBM WebSphere MQ Explorer.

Autorizzazione all'utilizzo di IBM WebSphere MQ Explorer

Qualsiasi utente può utilizzare IBM WebSphere MQ Explorer, ma sono necessarie determinate autorizzazioni per connettersi, accedere e gestire i gestori code.

Per eseguire le attività di gestione locali utilizzando WebSphere MQ Explorer, è necessario che un utente disponga dell'autorizzazione necessaria per eseguire le attività di gestione. Se l'utente è un membro del gruppo mqm , dispone dell'autorizzazione per eseguire tutte le attività amministrative locali.

Per connettersi a un gestore code remoto ed eseguire le attività di gestione remota utilizzando WebSphere MQ Explorer, l'utente che esegue WebSphere MQ Explorer deve disporre delle seguenti autorizzazioni:

- Autorizzazione CONNECT sull'oggetto gestore code di destinazione
- Autorizzazione INQUIRE sull'oggetto gestore code di destinazione
- Autorizzazione DISPLAY per l'oggetto gestore code di destinazione
- Autorizzazione INQUIRE per la coda, SYSTEM.MQEXPLORER.REPLY.MODEL
- Autorizzazione DISPLAY alla coda, SYSTEM.MQEXPLORER.REPLY.MODEL
- Autorizzazione INPUT (get) per la coda, SYSTEM.MQEXPLORER.REPLY.MODEL
- Autorizzazione OUTPUT (inserimento) per la coda, SYSTEM.ADMIN.COMMAND.QUEUE
- Autorizzazione INQUIRE sulla coda, SYSTEM.ADMIN.COMMAND.QUEUE
- Autorizzazione per eseguire l'azione selezionata

Nota: L'autorizzazione INPUT è relativa all'input per l'utente da una coda (un'operazione get). L'autorizzazione OUTPUT si riferisce all'output dall'utente ad una coda (un'operazione di inserimento).

Per connettersi a un gestore code remoto su WebSphere MQ per z/OS ed effettuare attività di gestione remote utilizzando IBM WebSphere MQ Explorer, è necessario fornire quanto segue:

- Un profilo RACF per la coda di sistema, SYSTEM.MQEXPLORER.REPLY.MODEL
- Un profilo RACF per le code, AMQ.MQEXPLORER.*

Inoltre, l'utente che esegue WebSphere MQ Explorer deve disporre delle seguenti autorizzazioni:

- Autorizzazione RACF UPDATE per la coda di sistema, SYSTEM.MQEXPLORER.REPLY.MODEL
- RACF Autorizzazione UPDATE alle code, AMQ.MQEXPLORER.*
- Autorizzazione CONNECT sull'oggetto gestore code di destinazione
- Autorizzazione per eseguire l'azione selezionata
- Autorizzazione READ per tutti i profili hlq.DISPLAY.object nella classe MQCMDS

Per informazioni su come concedere l'autorizzazione agli oggetti WebSphere MQ , consultare Concessione dell'accesso a un oggetto WebSphere MQ su sistemi UNIX o Linux e Windows.

Se un utente tenta di eseguire un'operazione che non è autorizzato ad eseguire, il gestore code di destinazione richiama le procedure di errore di autorizzazione e l'operazione non riesce.

Il filtro predefinito in WebSphere MQ Explorer è visualizzare tutti gli oggetti WebSphere MQ . Se sono presenti oggetti WebSphere MQ per i quali un utente non dispone dell'autorizzazione DISPLAY, vengono generati errori di autorizzazione. Se gli eventi di autorizzazione vengono registrati, limitare l'intervallo di oggetti visualizzati agli oggetti per i quali l'utente dispone dell'autorizzazione DISPLAY.

Sicurezza per la connessione ai gestori code remoti

È necessario proteggere il canale tra Esplora risorse di IBM WebSphere MQ e ciascun gestore code remoto.

IBM WebSphere MQ Explorer si connette ai gestori code remoti come applicazione client MQI. Ciò significa che ogni gestore code remoto deve avere una definizione di un canale di connessione server e un listener TCP/IP adatto. Se non si protegge il canale di connessione server, è possibile che un'applicazione dannosa si connetta allo stesso canale di connessione server e ottenga l'accesso agli oggetti del gestore code con autorizzazione illimitata. Per proteggere il canale di connessione del server, specificare un valore

non vuoto per l'attributo MCAUSER del canale, utilizzare i record di autenticazione del canale o utilizzare un'uscita di sicurezza.

Il valore predefinito dell'attributo MCAUSER è l'ID utente locale. Se si specifica un nome utente non vuoto come attributo MCAUSER del canale di connessione del server, tutti i programmi che si collegano al gestore code utilizzando questo canale vengono eseguiti con l'identità dell'utente denominato e dispongono dello stesso livello di autorizzazione. Ciò non si verifica se si utilizzano i record di autenticazione di canale.

Utilizzo di un'uscita di sicurezza con WebSphere MQ Explorer

È possibile specificare un'uscita di protezione predefinita e le uscite di sicurezza specifiche del gestore code utilizzando WebSphere MQ Explorer.

È possibile definire un'uscita di sicurezza predefinita, che può essere utilizzata per tutte le nuove connessioni client da WebSphere MQ Explorer. Questa uscita predefinita può essere sovrascritta al momento della connessione. È anche possibile definire un'uscita di sicurezza per un singolo gestore code o una serie di gestori code, che diventa effettiva quando viene effettuata una connessione. Le uscite vengono specificate utilizzando WebSphere MQ Explorer. Per ulteriori informazioni, consultare il Centro assistenza di WebSphere MQ .

Utilizzo di IBM WebSphere MQ Explorer per stabilire una connessione a un gestore code remoto utilizzando i canali MQI abilitati SSL

IBM WebSphere MQ Explorer si connette ai gestori code remoti utilizzando un canale MQI. Se si desidera proteggere il canale MQI utilizzando la sicurezza SSL, è necessario stabilire il canale utilizzando una tabella di definizione del canale del client.

Per informazioni su come stabilire un canale MQI utilizzando una tabella di definizione del canale client, consultare Panoramica dei client MQI IBM WebSphere MQ.

Una volta stabilito il canale utilizzando una tabella di definizione del canale del client, è possibile utilizzare IBM WebSphere MQ Explorer per connettersi a un gestore code remoto utilizzando il canale MQI abilitato per SSL, come descritto in “Attività sul sistema che ospita il gestore code remoto” a pagina 61 e “Attività sul sistema che ospita IBM WebSphere MQ Explorer” a pagina 62.

Attività sul sistema che ospita il gestore code remoto

Sul sistema che ospita il gestore code remoto, effettuare le seguenti attività:

1. Definire una coppia di canali di connessione server e client e specificare il valore appropriato per la variabile *SSLCIPH* sulla connessione server su entrambi i canali. Per ulteriori informazioni sulla variabile *SSLCIPH* , vedi Protezione dei canali con SSL
2. Inviare la tabella di definizione del canale `AMQCLCHL.TAB` , che si trova nella directory `@ipcc` del gestore code, al sistema su cui è presente IBM WebSphere MQ Explorer.
3. Avviare un listener TCP/IP su una porta designata.
4. Inserire i certificati CA e SSL personali nella directory SSL del gestore code:
   - `/var/mqm/qmgrs/+QMNAME+/SSL` per i sistemi UNIX and Linux
   - `C:\Program Files\WebSphere MQ\qmgrs\+QMNAME+\SSL` per i sistemi Windows

   Dove `+QMNAME+` è un token che rappresenta il nome del gestore code.
5. Creare un file database di chiavi di tipo CMS denominato `key.kdb` . Eseguire lo stash della password in un file selezionando l'opzione nella GUI di iKeyman o utilizzando l'opzione `-stash` con i comandi **`runmqckm`** .
6. Aggiungere i certificati CA al database delle chiavi creato nel passo precedente.
7. Importare il certificato personale per il gestore code nel database delle chiavi.

Per informazioni più dettagliate sull'utilizzo di SSL (Secure Sockets Layer) su sistemi Windows , consultare Utilizzo di SSL o TLS su sistemi UNIX, Linux e Windows .

Attività sul sistema che ospita IBM WebSphere MQ Explorer

Sul sistema che ospita IBM WebSphere MQ Explorer, effettuare le seguenti attività:

1. Creare un file database delle chiavi di tipo JKS denominato `key.jks`. Impostare una password per questo file di database delle chiavi.

   Il IBM WebSphere MQ Explorer utilizza i file keystore Java (JKS) per la sicurezza SSL e quindi il file keystore creato per configurare SSL per IBM WebSphere MQ Explorer deve corrispondere a questo.
2. Aggiungere i certificati CA al database delle chiavi creato nel passo precedente.
3. Importare il certificato personale per il gestore code nel database delle chiavi.
4. Su sistemi Windows e Linux , avviare MQ Explorer utilizzando il menu di sistema, il file eseguibile `MQExplorer` o il comando **`strmqcfg`** .
5. Dalla barra degli strumenti di IBM WebSphere MQ Explorer , fare clic su **Finestra -> Preferenze**, quindi espandere **WebSphere MQ Explorer** e fare clic su **Archivi certificati client SSL**. Immettere il nome e la password per il file JKS creato al passo 1 di “Attività sul sistema che ospita IBM WebSphere MQ Explorer” a pagina 62, sia nell'archivio certificati attendibili che nell'archivio certificati personali, quindi fare clic su **OK**.
6. Chiudere la finestra **Preferenze** e fare clic con il tasto destro del mouse su **Gestori code**. Fare clic su **Mostra / nascondi gestori code**, quindi su **Aggiungi** nella schermata **Mostra / nascondi gestori code** .
7. Immettere il nome del gestore code e selezionare l'opzione **Connetti direttamente** . Fare clic su Avanti.
8. Selezionare **Utilizza CCDT (client channel definition table)** e specificare l'ubicazione del file di tabella del canale trasferito dal gestore code remoto nel passo 2 in “Attività sul sistema che ospita il gestore code remoto” a pagina 61 sul sistema su cui è presente il gestore code remoto.
9. Fare clic su **Fine**. È ora possibile accedere al gestore code remoto da IBM WebSphere MQ Explorer.

Connessione tramite un altro gestore code

Esplora risorse di IBM WebSphere MQ consente di connettersi a un gestore code tramite un gestore code intermedio, a cui IBM WebSphere MQ Explorer è già connesso.

In questo caso, Esplora risorse di IBM WebSphere MQ inserisce i messaggi di comando PCF nel gestore code intermedio, specificando quanto segue:

- Il parametro *ObjectQMgrName* nel MQOD (object descriptor) come nome del gestore code di destinazione. Per ulteriori informazioni sulla risoluzione dei nomi delle code, consultare Risoluzione dei nomi .
- Il parametro *UserIdentifier* nel descrittore del messaggio (MQMD) come userIdlocale.

Se la connessione viene quindi utilizzata per connettersi al gestore code di destinazione tramite un gestore code intermedio, l' userId viene trasmesso nuovamente nel parametro *UserIdentifier* del descrittore del messaggio (MQMD). Per consentire al listener MCA sul gestore code di destinazione di accettare questo messaggio, è necessario impostare l'attributo MCAUSER oppure è necessario che userId esista già con l'autorizzazione di inserimento.

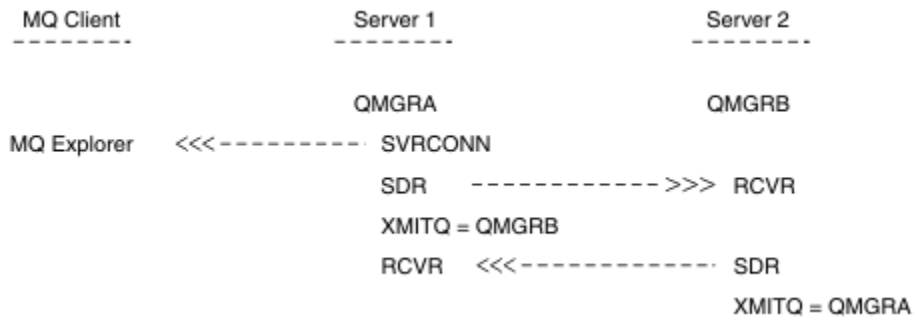
Il server dei comandi sul gestore code di destinazione inserisce i messaggi nella coda di trasmissione specificando userId nel parametro *UserIdentifier* nel descrittore del messaggio (MQMD). Perché questo inserimento abbia esito positivo, è necessario che l' userId esista già sul gestore code di destinazione con autorizzazione di inserimento.

L'esempio riportato di seguito mostra come connettere un gestore code, tramite un gestore code intermedio, a WebSphere MQ Explorer.

Stabilire una connessione di amministrazione remota a un gestore code. Verificare che:

- Il gestore code sul server è attivo e ha un canale di connessione server (SVRCONN) definito.
- Il listener è attivo.
- Il server dei comandi è attivo.

- SYSTEM.MQ EXPLORER.REPLY.MODEL è stata creata e si dispone dell'autorità sufficiente.
- I listener del gestore code, i server dei comandi e i canali mittente vengono avviati.



In questo esempio:

- IBM WebSphere MQ Explorer è connesso al gestore code QMGRA (in esecuzione su Server1) utilizzando una connessione client.
- Il gestore code QMGRB su Server2 può ora essere connesso a IBM WebSphere MQ Explorer tramite un gestore code intermedio (QMGRA)
- Quando ci si connette a QMGRB con WebSphere MQ Explorer, selezionare QMGRA come gestore code intermedio

In questa situazione, non vi è alcuna connessione diretta a QMGRB da IBM WebSphere MQ Explorer; la connessione a QMGRB avviene tramite QMGRA.

Il gestore code QMGRB su Server2 si connette a QMGRA su Server1 utilizzando i canali mittente - destinatario. Il canale tra QMGRA e QMGRB deve essere configurato in modo tale che l'amministrazione remota sia possibile; consultare “Preparazione di canali e code di trasmissione per la gestione remota” a pagina 107.

Visualizzazione e nascondimento di gestori code e cluster

IBM WebSphere MQ Explorer può visualizzare più di un gestore code alla volta. Dal pannello Mostra / Nascondi gestore code (selezionabile dal menu per il nodo della struttura ad albero Gestori code), è possibile scegliere se visualizzare le informazioni su un'altra macchina (remota). I gestori code locali vengono rilevati automaticamente.

Per visualizzare un gestore code remoto:

1. Fare clic con il tasto destro del mouse sul nodo della struttura ad albero `Queue Managers` , quindi selezionare *Mostra / nascondi gestori code ...*.
2. Fare clic su **Aggiungi**. Viene visualizzato il pannello Mostra / nascondi gestori code.
3. Immettere il nome del gestore code remoto e il nome host o l'indirizzo IP nei campi forniti.

   Il nome host o l'indirizzo IP viene utilizzato per stabilire una connessione client al gestore code remoto utilizzando il relativo canale di connessione server predefinito, SYSTEM.ADMIN.SVRCONNo un canale di connessione server definito dall'utente.
4. Fare clic su **Fine**.

Il pannello Mostra / nascondi gestori code visualizza anche un elenco di tutti i gestori code visibili. È possibile utilizzare questo pannello per nascondere i gestori code dalla vista di navigazione.

Se Esplora risorse di IBM WebSphere MQ visualizza un gestore code che è un membro di un cluster, il cluster viene rilevato e visualizzato automaticamente.

Per esportare l'elenco di gestori code remoti da questo pannello:

1. Chiudere il pannello Mostra / nascondi gestori code.

2. Fare clic con il tasto destro del mouse sul nodo della struttura ad albero **IBM IBM WebSphere MQ** nel riquadro di navigazione di WebSphere MQ Explorer, quindi selezionare **Esporta MQ Explorer Settings**
3. Fare clic su **MQ Explorer> MQ Explorer Settings**
4. Selezionare **Informazioni di connessione> Gestori code remoti**.
5. Selezionare un file in cui memorizzare le impostazioni esportate.
6. Infine, fare clic su **Fine** per esportare le informazioni di connessione del gestore code remoto nel file specificato.

Per importare un elenco di gestori code remoti:

1. Fare clic con il pulsante destro del mouse sul nodo della struttura ad albero **IBM IBM WebSphere MQ** nel pannello di navigazione di WebSphere MQ Explorer, quindi selezionare **Importa MQ Explorer Settings**
2. Fare clic su **MQ Explorer> MQ Explorer Settings**
3. Fare clic su **Sfoglia**e passare al percorso del file che contiene le informazioni di connessione del gestore code remoto.
4. Fare clic su **Apri**. Se il file contiene un elenco di gestori code remoti, la casella **Informazioni di connessione> Gestori code remoti** è selezionata.
5. Infine, fare clic su **Fine** per importare le informazioni di connessione del gestore code remoto in WebSphere MQ Explorer.

Appartenenza cluster

IBM WebSphere MQ Explorer richiede informazioni sui gestori code che sono membri di un cluster.

Se un gestore code è membro di un cluster, il nodo della struttura ad albero del cluster verrà popolato automaticamente.

Se i gestori code diventano membri dei cluster mentre Esplora risorse di IBM WebSphere MQ è in esecuzione, è necessario mantenere Esplora risorse di IBM WebSphere MQ con dati di gestione aggiornati sui cluster in modo che possano comunicare in modo efficace con essi e visualizzare le informazioni corrette sui cluster quando richiesto. Per eseguire questa operazione, WebSphere MQ Explorer richiede le seguenti informazioni:

- Il nome di un gestore code del repository
- Il nome della connessione del gestore code del repository se si trova su un gestore code remoto

Con queste informazioni, WebSphere MQ Explorer può:

- Utilizzare il gestore code del repository per ottenere un elenco di gestori code nel cluster.
- Gestire i gestori code che sono membri del cluster e si trovano su piattaforme e livelli di comando supportati.

La somministrazione non è possibile se:

- Il repository scelto diventa non disponibile. WebSphere MQ Explorer non passa automaticamente a un repository alternativo.
- Impossibile contattare il repository scelto tramite TCP/IP.
- Il repository scelto è in esecuzione su un gestore code in esecuzione su una piattaforma e a livello di comando non supportato da WebSphere MQ Explorer.

I membri del cluster che possono essere gestiti possono essere locali o remoti se possono essere contattati utilizzando TCP/IP. IBM WebSphere MQ Explorer si connette direttamente ai gestori code locali che sono membri di un cluster, senza utilizzare una connessione client.

Conversione dati

IBM WebSphere MQ Explorer funziona in CCSID 1208 (UTF-8). Ciò consente a Esplora risorse di IBM WebSphere MQ di visualizzare correttamente i dati dei gestori code remoti. Se ci si connette direttamente

a un gestore code o utilizzando un gestore code intermedio, IBM WebSphere MQ Explorer richiede che tutti i messaggi in entrata vengano convertiti in CCSID 1208 (UTF-8).

Se si tenta di stabilire una connessione tra Esplora risorse di IBM WebSphere MQ e un gestore code con un CCSID non riconosciuto da Esplora risorse di IBM WebSphere MQ , viene emesso un messaggio di errore.

Le conversioni supportate sono descritte in Conversione codepage.

Sicurezza su Windows

La procedura guidata Prepara WebSphere MQ crea un account utente particolare in modo che il servizio Windows possa essere condiviso dai processi che devono utilizzarlo.

Un servizio Windows viene condiviso tra processi client per un'installazione IBM WebSphere MQ . Viene creato un servizio per ciascuna installazione. Ciascun servizio è denominato `MQ_`*`InstallationName`*e ha un nome di visualizzazione `IBM WebSphere MQ(`*`InstallationName`*`)`. Prima di Version 7.1, con una singola installazione su un server, il servizio Windows era denominato `MQSeriesServices` con il nome di visualizzazione `IBM MQSeries`.

Poiché ogni servizio deve essere condiviso tra sessioni di accesso interattivo e non interattivo, è necessario avviarle con un account utente speciale. È possibile utilizzare un account utente speciale per tutti i servizi oppure creare diversi account utente speciali. Ogni account utente speciale deve avere il diritto utente su "Accedi come servizio", per ulteriori informazioni consultare “Diritti utente richiesti per un servizio IBM WebSphere MQ Windows” a pagina 66. Se l'ID utente non dispone dell'autorizzazione per eseguire il servizio, il servizio non viene avviato e restituisce un errore nel log eventi del sistema Windows . In genere, verrà eseguita la procedura guidata Prepara IBM WebSphere MQ e l'ID utente verrà impostato correttamente. Tuttavia, se l'ID utente è stato configurato manualmente, è possibile che si sia verificato un problema che sarà necessario risolvere.

Quando si installa IBM WebSphere MQ e si esegue la procedura guidata Prepara IBM WebSphere MQ per la prima volta, viene creato un account utente locale per il servizio denominato MUSR_MQADMIN con le autorizzazioni e le impostazioni richieste, incluso "Accedi come servizio".

Per le installazioni successive, la procedura guidata Prepara IBM WebSphere MQ crea un account utente denominato MUSR_MQADMINx, dove x è il successivo numero disponibile che rappresenta un ID utente che non esiste. La password per MUSR_MQADMINx viene generata in modo casuale quando l'account viene creato e utilizzata per configurare l'ambiente di accesso per il servizio. La password generata non scade.

Questo account IBM WebSphere MQ non è influenzato dalle politiche di account che sono impostate sul sistema per richiedere che le parole d'ordine dell'account vengano modificate dopo un certo periodo di tempo.

La password non è nota all'esterno di questa elaborazione monouso ed è memorizzata dal sistema operativo Windows in una parte sicura del registro.

Utilizzo di Active Directory (soloWindows)

In alcune configurazioni di rete, in cui gli account utente sono definiti su controller di dominio che utilizzano Active Directory, l'account utente locale IBM WebSphere MQ in esecuzione potrebbe non disporre dell'autorizzazione richiesta per eseguire la query dell'appartenenza al gruppo di altri account utente del dominio. La procedura guidata Prepara IBM WebSphere MQ identifica se questo è il caso eseguendo test e ponendo domande all'utente sulla configurazione della rete.

Se l'account utente locale con cui è in esecuzione IBM WebSphere MQ non dispone dell'autorizzazione richiesta, la procedura guidata Prepara IBM WebSphere MQ richiederà all'utente i dettagli dell'account di un account utente di dominio con particolari diritti utente. Per i diritti utente richiesti dall'account utente del dominio, consultare “Diritti utente richiesti per un servizio IBM WebSphere MQ Windows” a pagina 66. Una volta immessi i dettagli dell'account validi per l'account utente del dominio nella procedura guidata Prepara IBM WebSphere MQ , l'utente configura un servizio IBM WebSphere MQ Windows da

eseguire con il nuovo account. I dettagli dell'account sono conservati nella parte protetta del registro e non possono essere letti dagli utenti.

Quando il servizio è in esecuzione, viene avviato un servizio IBM WebSphere MQ Windows che rimane in esecuzione per tutto il tempo in cui il servizio è in esecuzione. Un amministratore IBM WebSphere MQ che accede al server dopo l'avvio del servizio Windows può utilizzare IBM WebSphere MQ Explorer per gestire i gestori code sul server. Ciò connette IBM WebSphere MQ Explorer al processo del servizio Windows esistente. Queste due azioni hanno bisogno di diversi livelli di autorizzazione prima di poter funzionare:

- Il processo di avvio richiede un'autorizzazione di avvio.
- L'amministratore IBM WebSphere MQ richiede l'autorizzazione di accesso.

Diritti utente richiesti per un servizio IBM WebSphere MQ Windows

La tabella in questo argomento elenca i diritti utente richiesti per l'account utente locale e di dominio con cui viene eseguito il servizio Windows per un'installazione di IBM WebSphere MQ .

Accedi come lavoro batch	Abilita un servizio IBM WebSphere MQ Windows da eseguire con questo account utente.
Accedi come servizio	Consente agli utenti di impostare il servizio IBM WebSphere MQ Windows per accedere utilizzando l'account configurato.
Arresta il sistema	Consente al servizio IBM WebSphere MQ Windows di riavviare il server se configurato per farlo quando il ripristino di un servizio ha esito negativo.
Incremento quote	Richiesto per la chiamata del sistema operativo `CreateProcessAsUser` .
Azione come parte del sistema operativo	Richiesto per la chiamata del sistema operativo `LogonUser` .
Ignora controllo trasversale	Richiesto per la chiamata del sistema operativo `LogonUser` .
Sostituzione di un token livello elaborazione	Richiesto per la chiamata del sistema operativo `LogonUser` .

Nota: I diritti dei programmi di debug potrebbero essere richiesti in ambienti su cui sono in esecuzione applicazioni ASP e IIS.

L'account utente del dominio deve avere questi diritti utente Windows impostati come diritti utente effettivi, come elencato nell'applicazione Criteri di sicurezza locali. In caso contrario, impostarli utilizzando l'applicazione Local Security Policy localmente sul server o utilizzando l'applicazione Domain Security Application a livello di dominio.

Modifica del nome utente associato al servizio IBM WebSphere MQ

Potrebbe essere necessario modificare il nome utente associato al Servizio IBM WebSphere MQ da MUSR_MQADMIN a qualcos' altro. (Ad esempio, potrebbe essere necessario effettuare questa operazione se il gestore code è associato a DB2, che non accetta nomi utente di lunghezza superiore a 8 caratteri.)

Procedura

1. Creare un nuovo account utente (ad esempio **NEW_NAME**)
2. Utilizzare la procedura guidata Prepara IBM WebSphere MQ per immettere i dettagli del nuovo account utente.

Modifica della parola d'ordine dell'account utente del servizio IBM WebSphere MQ Windows

Informazioni su questa attività

Per modificare la password dell'account utente locale del servizioIBM WebSphere MQ Windows , effettuare le seguenti operazioni:

Procedura

1. Identifica l'utente con cui è in esecuzione il servizio.
2. Arrestare il servizio IBMIBM WebSphere MQ dal pannello Gestione computer.
3. Modificare la password richiesta nello stesso modo in cui si modificherebbe la password di un individuo.
4. Accedere alle proprietà per il servizio IBM WebSphere MQ dal pannello Gestione computer.
5. Selezionare la pagina **Accedi** .
6. Confermare che il nome account specificato corrisponda all'utente per cui è stata modificata la password.
7. Immettere la password nei campi **Password** e **Conferma password** e fare clic su **OK**.

Servizio IBM WebSphere MQ Windows per un'istallazione in esecuzione con un account utente del dominio

Informazioni su questa attività

Se il servizio IBM WebSphere MQ Windows per un'installazione è in esecuzione con un account utente del dominio, è anche possibile modificare la password per l'account nel modo seguente:

Procedura

1. Modificare la password per l'account di dominio sul controller di dominio. Potrebbe essere necessario chiedere all'amministratore del dominio di eseguire questa operazione.
2. Attieniti alla procedura per modificare la pagina **Log On** per il servizio IBMIBM WebSphere MQ .

   L'account utente con cui viene eseguito il servizio IBM WebSphere MQ Windows esegue tutti i comandi MQSC emessi dalle applicazioni dell'interfaccia utente o eseguiti automaticamente all'avvio del sistema, all'arresto o al ripristino del servizio. Questo account utente deve quindi disporre dei diritti di amministrazione IBM WebSphere MQ . Per impostazione predefinita, viene aggiunto al gruppo **mqm** locale sul server. Se questa appartenenza viene rimossa, il servizio IBM WebSphere MQ Windows non funziona. Per ulteriori informazioni sui diritti utente, consultare “Diritti utente richiesti per un servizio IBM WebSphere MQ Windows” a pagina 66

   Se si verifica un problema di sicurezza con l'account utente con cui viene eseguito il servizio IBM WebSphere MQ Windows , i messaggi di errore e le descrizioni vengono visualizzati nel log eventi di sistema.

Concetti correlati

“Utilizzo di Active Directory (soloWindows)” a pagina 65

In alcune configurazioni di rete, in cui gli account utente sono definiti su controller di dominio che utilizzano Active Directory, l'account utente locale IBM WebSphere MQ in esecuzione potrebbe non disporre dell'autorizzazione richiesta per eseguire la query dell'appartenenza al gruppo di altri account utente del dominio. La procedura guidata Prepara IBM WebSphere MQ identifica se questo è il caso eseguendo test e ponendo domande all'utente sulla configurazione della rete.

IBM WebSphere MQ coordinamento con Db2 come gestore risorse

Se si avviano i gestori code da IBM WebSphere MQ Explorero si utilizzano IBM WebSphere MQ V7e si verificano dei problemi durante il coordinamento di Db2, controllare i log degli errori del gestore code.

Controllare i log degli errori del gestore code per un errore simile al seguente:

```
23/09/2008 15:43:54 - Process(5508.1) User(MUSR_MQADMIN) Program(amqzxma0.exe)
Host(HOST_1) Installation(Installation1)
```

```
VMRF(7.1.0.0) QMgr(A.B.C)
AMQ7604: The XA resource manager 'DB2 MQBankDB database' was not available when called
    for xa_open. The queue manager is continuing without this resource manager.
```

Spiegazione: l'ID utente (il nome predefinito è MUSR_MQADMIN) che esegue ilIBM WebSphere MQ Processo servizio `amqsvc.exe` è ancora in esecuzione con un token di accesso che non contiene le informazioni di appartenenza al gruppo per il gruppo DB2USERS.

Risolvi: dopo aver verificato che l'ID utente del servizio IBM WebSphere MQ è un membro di DB2USERS, utilizzare la seguente sequenza di comandi:

- Arresta il servizio.
- arrestare tutti gli altri processi in esecuzione con lo stesso ID utente.
- riavviare questi processi.

Il riavvio della macchina garantirebbe i passaggi precedenti, ma non è necessario.

Estensione di IBM WebSphere MQ Explorer

IBM WebSphere MQ per Windowse IBM WebSphere MQ per Linux (piattaformex86 e x86-64) forniscono un'interfaccia di gestione denominata IBM WebSphere MQ Explorer per eseguire le attività di gestione come alternativa all'utilizzo dei comandi di controllo o MQSC.

Queste informazioni si applicano solo a WebSphere MQ per Windows e WebSphere MQ per Linux (piattaformex86 e x86-64).

IBM WebSphere MQ Explorer presenta le informazioni in uno stile coerente con quello del framework Eclipse e con le altre applicazioni plug-in supportate da Eclipse .

Mediante l'estensione di IBM WebSphere MQ Explorer, gli amministratori di sistema hanno la possibilità di personalizzare WebSphere MQ Explorer per migliorare il modo in cui amministrano WebSphere MQ.

Per ulteriori informazioni, consultare *Estensione di IBM WebSphere MQ Explorer* nella documentazione del prodotto IBM WebSphere MQ Explorer.

Utilizzo dell'applicazione della barra delle attività IBM WebSphere MQ (solo Windows)

L'applicazione della barra delle applicazioni IBM WebSphere MQ visualizza un'icona nella barra delle applicazioni Windows sul server. L'icona fornisce lo stato corrente di IBM WebSphere MQ e un menu da cui è possibile eseguire alcune semplici azioni.

In Windows, l'icona WebSphere MQ si trova nella barra delle applicazioni del server ed è sovrapposta ad un simbolo di stato con codifica a colori, che può avere uno dei seguenti significati:

Verde
Funzionamento corretto; nessun avviso al momento

Blu
Indeterminato; WebSphere MQ è in fase di avvio o di arresto

Giallo
Avviso; uno o più servizi sono in errore o hanno già avuto esito negativo

Per visualizzare il menu, fare clic con il pulsante destro del mouse sull'icona WebSphere MQ . Dal menu è possibile eseguire le seguenti operazioni:

- Fare clic su **Apri** per aprire WebSphere MQ Alert Monitor
- Fare clic su **Esci** per uscire dall'applicazione della barra delle applicazioni WebSphere MQ
- Fare clic su **WebSphere MQ Explorer** per avviare IBM WebSphere MQ Explorer
- Fare clic su **Arresta WebSphere MQ** per arrestare WebSphere MQ
- Fare clic su **Informazioni su WebSphere MQ** per visualizzare le informazioni sul Controllo segnalazioni WebSphere MQ

L'applicazione di controllo segnalazioni IBM WebSphere MQ (solo Windows)

Il controllo degli avvisi IBM WebSphere MQ è uno strumento di rilevamento degli errori che identifica e registra i problemi con IBM WebSphere MQ su una macchina locale.

Il controllo avvisi visualizza le informazioni sullo stato corrente dell'installazione locale di un server WebSphere MQ . Inoltre, monitora Windows Advanced Configuration and Power Interface (ACPI) e garantisce l'applicazione delle impostazioni ACPI.

Da WebSphere MQ Alert Monitor, è possibile:

- Accedere direttamente a Esplora risorse di IBM WebSphere MQ
- Visualizza le informazioni relative a tutti gli avvisi in sospeso
- Arrestare il servizio WebSphere MQ sulla macchina locale
- Instradare i messaggi di avviso sulla rete a un account utente configurabile o a una workstation o a un server Windows

Amministrazione di oggetti IBM WebSphere MQ locali

Questa sezione spiega come gestire gli oggetti IBM WebSphere MQ locali per supportare i programmi applicativi che utilizzano MQI (Message Queue Interface). In questo contesto, amministrazione locale significa creare, visualizzare, modificare, copiare ed eliminare oggetti IBM WebSphere MQ .

Oltre agli approcci descritti in questa sezione, è possibile utilizzare IBM WebSphere MQ Explorer per gestire gli oggetti WebSphere MQ locali; consultare “Amministrazione mediante IBM WebSphere MQ Explorer” a pagina 57.

Questa sezione contiene le seguenti informazioni:

Avvio e arresto di un gestore code

Utilizzare questo argomento come introduzione all'arresto e all'avvio di un gestore code.

Avvio di un gestore code

Per avviare un gestore code, utilizzare il comando `strmqm` nel modo seguente:

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

Su WebSphere MQ per sistemi Windows e WebSphere MQ per Linux (piattaformex86 e x86-64), è possibile avviare un gestore code nel modo seguente:

1. Aprire Esplora risorse di IBM WebSphere MQ .
2. Selezionare il gestore code dalla vista Navigator .
3. Fare clic su `Start`. Il gestore code viene avviato.

Se l'avvio del gestore code impiega più di pochi secondi, WebSphere MQ emette messaggi informativi che descrivono in modo intermittente l'avanzamento dell'avvio.

Il comando strmqm non restituisce il controllo fino a quando il gestore code non viene avviato ed è pronto ad accettare le richieste di connessione.

Avvio automatico di un gestore code

In WebSphere MQ for Windows , è possibile avviare automaticamente un gestore code quando il sistema viene avviato utilizzando WebSphere MQ Explorer. Per ulteriori informazioni, vedere “Amministrazione mediante IBM WebSphere MQ Explorer” a pagina 57.

Arresto di un gestore code

Utilizzare il comando **endmqm** per arrestare un gestore code.

Nota: È necessario utilizzare il comando **endmqm** dall'installazione associata al gestore code che si sta utilizzando. È possibile scoprire a quale installazione è associato un gestore code utilizzando il comando dspmq -o installation .

Ad esempio, per arrestare un gestore code denominato QMB, immettere il seguente comando:

```
endmqm QMB
```

Su WebSphere MQ per sistemi Windows e WebSphere MQ per sistemi Linux (x86 e x86-64), è possibile arrestare un gestore code nel modo seguente:

1. Aprire Esplora risorse di IBM WebSphere MQ .
2. Selezionare il gestore code dalla vista Navigator .
3. Fare clic su Stop.... Viene visualizzato il pannello Fine gestore code.
4. Selezionare Controllato o Immediato.
5. Fare clic su OK. Il gestore code viene arrestato.

arresto inattivo

Per impostazione predefinita, il comando **endmqm** esegue un arresto inattivo del gestore code specificato. Il completamento di questa operazione potrebbe richiedere un po' di tempo. Una chiusura sospesa attende che tutte le applicazioni connesse si disconnettano.

Utilizzare questo tipo di arresto per notificare le applicazioni da arrestare. Se si immette:

```
endmqm -c QMB
```

non viene indicato quando tutte le applicazioni sono state arrestate. Un comando endmqm -c QMB è equivalente a un comando endmqm QMB .

Tuttavia, se si immette:

```
endmqm -w QMB
```

il comando attende che tutte le applicazioni siano state arrestate e che il gestore code sia terminato.

arresto immediato

Per un arresto immediato, tutte le chiamate MQI correnti possono essere completate, ma le nuove chiamate non riescono. Questo tipo di arresto non attende la disconnessione delle applicazioni dal gestore code.

Per un arresto immediato, immettere:

```
endmqm -i QMB
```

arresto preventivo

Nota: Non utilizzare questo metodo a meno che tutti gli altri tentativi di arresto del gestore code tramite il comando **endmqm** non abbiano avuto esito negativo. Questo metodo può avere conseguenze imprevedibili per le applicazioni collegate.

Se un arresto immediato non funziona, è necessario ricorrere a un arresto *preventivo* , specificando l'indicatore `-p` . Ad esempio:

```
endmqm -p QMB
```

Questo arresta immediatamente il gestore code. Se questo metodo non funziona ancora, consultare “Arresto manuale di un gestore code” a pagina 71 per una soluzione alternativa.

Per una descrizione dettagliata del comando **endmqm** e delle relative opzioni, vedere endmqm.

In caso di problemi durante la chiusura di un gestore code

I problemi di chiusura di un gestore code sono spesso causati dalle applicazioni. Ad esempio, quando le applicazioni:

- Non controllare correttamente i codici di ritorno MQI
- Non richiedere la notifica di un quiesce
- Terminare senza disconnettersi dal gestore code (eseguendo una chiamata `MQDISC`)

Se si verifica un problema quando si arresta il gestore code, è possibile interrompere il comando **endmqm** utilizzando Ctrl-C. È quindi possibile immettere un altro comando **endmqm** , ma questa volta con un indicatore che specifica il tipo di arresto richiesto.

Arresto manuale di un gestore code

Se i metodi standard per arrestare i gestori code hanno esito negativo, provare i metodi descritti di seguito.

La modalità standard di arresto dei gestori code consiste nell'utilizzare il comando `endmqm` . Per arrestare manualmente un gestore code, utilizzare una delle procedure descritte in questa sezione. Per dettagli su come eseguire operazioni sui gestori code utilizzando i comandi di controllo, consultare Creazione e gestione dei gestori code.

Arresto dei gestori code su Windows

Come terminare i processi e il servizio IBM WebSphere MQ per arrestare i gestori code in IBM WebSphere MQ per Windows.

Per arrestare un gestore code in esecuzione in WebSphere MQ per Finestre:

1. Elencare i nomi (ID) dei processi in esecuzione, utilizzando Gestione attività di Windows .
2. Terminare i processi utilizzando Windows Task Manager o il comando **taskkill** , nel seguente ordine (se sono in esecuzione):

AMQZMUC0	Gestore processi critici
AMQZXMA0	Controllore di Esecuzione
AMQZFUMA	processo OAM
AMQZLAA0	Agent LQM
AMQZLSA0	Agent LQM
AMQZMUF0	Gestore utilità
AMQZMGR0	Controller processo
AMQZMUR0	Gestore processi riavviabile

AMQFQPUB	Processo di pubblicazione - sottoscrizione
AMQFCXBA	Processo di lavoro broker
AMQRMPPA	processo di pooling del processo
AMQCRSTA	Processo di lavoro del responder non sottoposto a thread
AMQCRS6B	Canale ricevente LU62 e connessione client
AMQRRMFA	Il processo del repository (per i cluster)
AMQZDMAA	Processore messaggi differiti
AMQPCSEA	Il server comandi
RUNMQTRM	Richiama un controllo trigger per un server
RUNMQDLQ	Richiama gestore code di messaggi non recapitabili
RUNMQCHI	Il processo iniziatore del canale
RUNMQLSR	Il processo del listener del canale
VNXMQS	Server di memoria condivisa
TRCADMQZ	Trace

3. Arrestare il servizio WebSphere MQ da **Strumenti di amministrazione** > **Servizi** nel Pannello di controllo di Windows .
4. Se sono stati tentati tutti i metodi e il gestore code non è stato arrestato, riavviare il sistema.

Il Task Manager Windows e il comando **tasklist** forniscono informazioni limitate sulle attività. Per ulteriori informazioni che consentono di determinare quali processi sono correlati a un determinato gestore code, utilizzare uno strumento come *Process Explorer* (procexp.exe), disponibile per il download dal sito Web Microsoft all'indirizzo https://www.microsoft.com

Arresto dei gestori code sui sistemi UNIX and Linux

Come terminare i processi e il servizio IBM WebSphere MQ , per arrestare i gestori code in IBM WebSphere MQ per UNIX and Linux. È possibile provare i metodi descritti di seguito se i metodi standard per l'arresto e la rimozione dei gestori code hanno esito negativo.

Per arrestare un gestore code in esecuzione in WebSphere MQ per sistemi UNIX and Linux :

1. Individuare l'ID processo dei programmi del gestore code ancora in esecuzione utilizzando il comando `ps` . Ad esempio, se il gestore code è denominato `QMNAME`, utilizzare il seguente comando:

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

2. Terminare tutti i processi del gestore code ancora in esecuzione. Utilizzare il comando `kill` , specificando gli ID processo rilevati utilizzando il comando `ps` .

   Terminare i processi nel seguente ordine:

amqzmuc0	Gestore processi critici
amqzxma0	Controllore di Esecuzione
amqzfuma	processo OAM
amqzlaa0	Agent LQM
amqzlsa0	Agent LQM
amqzmuf0	Gestore utilità
amqzmur0	Gestore processi riavviabile
amqzmgr0	Controller processo

amqfqpub	Processo di pubblicazione - sottoscrizione
amqfcxba	Processo di lavoro broker
amqrmppa	processo di pooling del processo
amqcrsta	Processo di lavoro del responder non sottoposto a thread
amqcrs6b	Canale ricevente LU62 e connessione client
amqrrmfa	Il processo del repository (per i cluster)
amqzdmaa	Processore messaggi differiti
amqpcsea	Il server comandi
runmqtrm	Richiama un controllo trigger per un server
runmqdlq	Richiama gestore code di messaggi non recapitabili
runmqchi	Il processo iniziatore del canale
runmqlsr	Il processo del listener del canale

Nota: È possibile utilizzare il comando **`kill -9`** per terminare i processi che non riescono ad arrestarsi.

Se si arresta il gestore code manualmente, è possibile che vengano utilizzati FFST e che i file FDC vengano inseriti in `/var/mqm/errors.` . Non considerare questo come un difetto nel gestore code.

Il gestore code verrà riavviato normalmente, anche dopo averlo arrestato utilizzando questo metodo.

Esecuzione di attività di amministrazione locale utilizzando i comandi MQSC

Questa sezione introduce i comandi MQSC e spiega come utilizzarli per alcune attività comuni.

Se si utilizza IBM WebSphere MQ per Windows o IBM WebSphere MQ per Linux (piattaformex86 e x86-64), è anche possibile eseguire le operazioni descritte in questa sezione utilizzando IBM WebSphere MQ Explorer. Per ulteriori informazioni, fare riferimento a “Amministrazione mediante IBM WebSphere MQ Explorer” a pagina 57.

È possibile utilizzare i comandi di MQSC per gestire gli oggetti del gestore code, inclusi il gestore code, le code, le definizioni di processo, canali, canali di connessione client, listener, servizi, elenchi nomi, cluster e oggetti delle informazioni di autenticazione. Questa sezione si occupa di gestori code, code e definizioni di processo; per informazioni sulla gestione del canale, del canale di connessione client e degli oggetti listener, consultare Oggetti . Per informazioni su tutti i comandi MQSC per la gestione di oggetti gestore code, consultare “Comandi script (MQSC)” a pagina 74.

Si immettono comandi MQSC a un gestore code utilizzando il comando `runmqsc` . Per i dettagli di questo comando, consultare runmqsc. È possibile eseguire questa operazione in modo interattivo, emettendo comandi da una tastiera oppure reindirizzare l'unità di immissione standard (`stdin`) per eseguire una sequenza di comandi da un file di testo ASCII. In entrambi i casi, il formato dei comandi è lo stesso. (Per informazioni sull'esecuzione dei comandi da un file di testo, vedere “Esecuzione dei comandi MQSC dai file di testo” a pagina 77.)

È possibile eseguire il comando `runmqsc` in tre modi, in base agli indicatori impostati sul comando:

- Verificare un comando senza eseguirlo, in cui i comandi MQSC vengono verificati su un gestore code locale, ma non vengono eseguiti.
- Eseguire un comando su un gestore code locale, in cui i comandi MQSC vengono eseguiti su un gestore code locale.
- Eseguire un comando su un gestore code remoto, in cui i comandi MQSC vengono eseguiti su un gestore code remoto.

È inoltre possibile eseguire il comando seguito da un punto interrogativo per visualizzare la sintassi.

Gli attributi dell'oggetto specificati nei comandi MQSC vengono mostrati in maiuscolo in questa sezione (ad esempio, RQMNAME), sebbene non siano sensibili al maiuscolo / minuscolo. I nomi attributo del

comando MQSC sono limitati a otto caratteri. I comandi MQSC sono disponibili su altre piattaforme, tra cui IBM i e z/OS.

I comandi MQSC vengono riepilogati nella raccolta di argomenti nella sezione Riferimento MQSC .

Comandi script (MQSC)

I comandi MQSC forniscono un metodo uniforme di emissione di comandi leggibili su piattaforme WebSphere MQ . Per informazioni sui comandi PCF (*programmable command format*), vedere “Introduzione ai formati di comando programmabili” a pagina 9.

Il formato generale dei comandi viene mostrato in Comandi MQSC.

È necessario osservare le seguenti regole quando si utilizzano i comandi MQSC:

- Ogni comando inizia con un parametro primario (un verbo) e questo è seguito da un parametro secondario (un nome). Questo è seguito dal nome o dal nome generico dell'oggetto (tra parentesi) se ne esiste uno, che è presente nella maggior parte dei comandi. Successivamente, i parametri possono di solito verificarsi in qualsiasi ordine; se un parametro ha un valore corrispondente, il valore deve essere immediatamente successivo al parametro a cui si riferisce.
- Parole chiave, parentesi e valori possono essere separati da qualsiasi numero di spazi e virgole. Una virgola mostrata nei diagrammi di sintassi può essere sempre sostituita da uno o più spazi vuoti. Deve essere presente almeno uno spazio vuoto immediatamente prima di ogni parametro (dopo il parametro primario).
- Qualsiasi numero di spazi vuoti può verificarsi all'inizio o alla fine del comando e tra parametri, punteggiatura e valori. Ad esempio, il seguente comando è valido:

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

  Gli spazi vuoti all'interno di una coppia di apici sono significativi.
- Le virgole aggiuntive possono essere visualizzate in qualsiasi punto in cui sono consentiti spazi e vengono trattate come se fossero spazi vuoti (a meno che, naturalmente, non siano all'interno di stringhe racchiuse tra virgolette).
- Non sono consentiti parametri ripetuti. La ripetizione di un parametro con la sua versione "NO", come in REPLACE NOREPLACE, non è consentita.
- Stringhe che contengono spazi, caratteri minuscoli o caratteri speciali diversi da:
  - Punto (.)
  - Barra (/)
  - Trattino basso (_)
  - Segno percentuale (%)

  deve essere racchiuso tra virgolette singole, a meno che non siano:
  - Valori generici che terminano con un asterisco
  - Un singolo asterisco (ad esempio, TRACE (*))
  - Una specifica di intervallo contenente due punti (ad esempio, CLASS (01:03))

  Se la stringa contiene una virgoletta singola, la virgoletta singola è rappresentata da due virgolette singole.I caratteri minuscoli non contenuti tra virgolette vengono ripiegati in maiuscolo.
- Su piattaforme diverse da z/OS, una stringa che non contiene caratteri (vale a dire, due virgolette singole senza spazi tra) viene interpretata come uno spazio vuoto racchiuso tra virgolette singole, vale a dire, interpretata allo stesso modo di (''). L'eccezione è se l'attributo utilizzato è uno dei seguenti:
  - TOPICSTR
  - SUB
  - USERDATA

– SELECTOR

due virgolette singole senza spazi vengono interpretate come una stringa di lunghezza zero.

- In v7.0, tutti gli spazi vuoti finali in tali attributi stringa che si basano sui tipi MQCHARV, come SELECTOR, dati utente secondari, vengono trattati come significativi, il che significa che 'abc' non è uguale a 'abc'.
- Una parentesi di apertura seguita da una parentesi di chiusura, senza alcuna informazione significativa nel mezzo, ad esempio

```
NAME ( )
```

non è valido tranne dove specificatamente indicato.

- Le parole chiave non sono sensibili al maiuscolo / minuscolo: AltER, alter e ALTER sono tutte accettabili.Tutto ciò che non è contenuto tra virgolette viene ripiegato in maiuscolo.
- I sinonimi sono definiti per alcuni parametri.Ad esempio, DEF è sempre un sinonimo di DEFINE, quindi DEF QLOCAL è valido. I sinonimi non sono, tuttavia, solo stringhe minime; DEFI non è un sinonimo valido per DEFINE.

  Nota: Non esiste alcun sinonimo per il parametro DELETE. Ciò per evitare l'eliminazione accidentale di oggetti quando si utilizza DEF, il sinonimo di DEFINE.

Per una panoramica sull'utilizzo dei comandi MQSC per la gestione di IBM WebSphere MQ, consultare “Esecuzione di attività di amministrazione locale utilizzando i comandi MQSC” a pagina 73.

I comandi MQSC utilizzano determinati caratteri speciali per avere determinati significati. Per ulteriori informazioni su questi caratteri speciali e su come utilizzarli, consultare Caratteri con significati speciali.

Per informazioni su come creare script utilizzando i comandi MQSC, consultare Creazione di script di comando.

Per un elenco completo dei comandi MQSC, consultare Comandi MQSC.

Attività correlate
Creazione di script di comando

WebSphere MQ nomi oggetto

Come utilizzare i nomi oggetto nei comandi MQSC.

Negli esempi, vengono utilizzati alcuni nomi lunghi per gli oggetti. Ciò consente di identificare il tipo di oggetto che si sta gestendo.

Quando si immettono comandi MQSC, è necessario specificare solo il nome locale della coda. Negli esempi, vengono utilizzati i seguenti nomi di coda:

```
ORANGE.LOCAL.QUEUE
```

La parte `LOCAL.QUEUE` del nome è per illustrare che questa coda è una coda locale. ***non*** è richiesto per i nomi delle code locali in generale.

Viene utilizzato anche il nome `saturn.queue.manager` come nome gestore code. La parte `queue.manager` del nome indica che questo oggetto è un gestore code. *Non* è richiesto per i nomi dei gestori code in generale.

Sensibilità al maiuscolo / minuscolo nei comandi MQSC

I comandi MQSC, inclusi i relativi attributi, possono essere scritti in maiuscolo o in minuscolo. I nomi degli oggetti nei comandi MQSC vengono ripiegati in maiuscolo (vale a dire, QUEUE e queue non sono differenziati), a meno che i nomi non siano racchiusi tra virgolette singole. Se non vengono utilizzate le virgolette, l'oggetto viene elaborato con un nome in maiuscolo. Per ulteriori informazioni, consultareGuida di riferimento a MQSC .

Il richiamo del comando `runmqsc` , in comune con tutti i comandi di controllo WebSphere MQ , è sensibile al maiuscolo / minuscolo in alcuni ambienti WebSphere MQ . Per ulteriori informazioni, consultare Utilizzo dei comandi di controllo .

Input e output standard

La *periferica di input standard*, indicata anche come `stdin`, è la periferica da cui viene preso l'input per il sistema. In genere questa è la tastiera, ma è possibile specificare che l'input deve provenire da una porta seriale o da un file disco, ad esempio. La *periferica di output standard*, nota anche come `stdout`, è la periferica a cui viene inviato l'output dal sistema. In genere si tratta di una visualizzazione, ma è possibile reindirizzare l'output a una porta seriale o a un file.

Nei comandi del sistema operativo e nei comandi del controllo WebSphere MQ , l'operatore < reindirizza l'immissione. Se questo operatore è seguito da un nome file, l'immissione viene presa dal file. Allo stesso modo, l'operatore> reindirizza l'output; se questo operatore è seguito da un nome file, l'output viene indirizzato a tale file.

Utilizzo interattivo dei comandi MQSC

È possibile utilizzare i comandi MQSC in modo interattivo utilizzando una finestra di comandi o una shell.

Per utilizzare i comandi MQSC in modo interattivo, aprire una finestra di comandi o una shell e immettere:

```
runmqsc
```

In questo comando, non è stato specificato un nome gestore code, pertanto i comandi MQSC vengono elaborati dal gestore code predefinito. Se si desidera utilizzare un gestore code differente, specificare il nome del gestore code nel comando **runmqsc** . Ad esempio, per eseguire comandi MQSC sul gestore code `jupiter.queue.manager`, utilizzare il comando:

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

Successivamente, tutti i comandi MQSC immessi vengono elaborati da questo gestore code, supponendo che si trovi sullo stesso nodo e che sia già in esecuzione.

Ora è possibile immettere qualsiasi comando MQSC, come richiesto. Ad esempio, prova questo:

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

Per i comandi che hanno troppi parametri per adattarsi su una riga, utilizzare i caratteri di continuazione per indicare che un comando continua sulla seguente riga:

- Un segno meno (-) indica che il comando deve continuare dall'inizio della seguente riga.
- Un segno più (+) indica che il comando deve continuare dal primo carattere non vuoto sulla seguente riga.

L'immissione del comando termina con il carattere finale di una riga non vuota che non è un carattere di continuazione. È anche possibile terminare esplicitamente l'input del comando immettendo un punto e virgola (;). Ciò è particolarmente utile se si immette accidentalmente un carattere di continuazione alla fine della riga finale di immissione del comando.

Feedback dai comandi MQSC

Quando si immettono i comandi MQSC, il gestore code restituisce i messaggi dell'operatore che confermano le azioni dell'utente o che indicano gli errori effettuati. Ad esempio:

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

Questo messaggio conferma che è stata creata una coda.

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

AMQ8426: Valid MQSC commands are:


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

Questo messaggio indica che è stato commesso un errore di sintassi.

Questi messaggi vengono inviati alla periferica di output standard. Se il comando non è stato immesso correttamente, fare riferimento a Guida di riferimento a MQSC per la sintassi corretta.

Fine dell'input interattivo dei comandi MQSC

Per interrompere l'utilizzo dei comandi MQSC, immettere il comando END.

In alternativa, è possibile utilizzare il carattere EOF per il proprio sistema operativo.

Esecuzione dei comandi MQSC dai file di testo

L'esecuzione interattiva dei comandi MQSC è adatta per test rapidi, ma se si dispone di comandi molto lunghi o si sta utilizzando una particolare sequenza di comandi ripetutamente, considerare il reindirizzamento di `stdin` da un file di testo.

“Input e output standard” a pagina 76 contiene informazioni su `stdin` e `stdout`. Per reindirizzare `stdin` da un file di testo, creare prima un file di testo contenente i comandi MQSC utilizzando il normale editor di testo. Quando si usa il comando `runmqsc` , utilizzare gli operatori di reindirizzamento. Ad esempio, il seguente comando esegue una serie di comandi contenuti nel file di testo `myprog.in`:

```
runmqsc < myprog.in
```

Allo stesso modo, è possibile reindirizzare l'output a un file. Un file contenente i comandi MQSC per l'input viene denominato *File di comandi MQSC*. Il file di output che contiene le risposte dal gestore code è denominato *file di output*.

Per reindirizzare `stdin` e `stdout` sul comando `runmqsc` , utilizzare il seguente formato del comando:

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

Questo comando richiama i comandi MQSC contenuti nel file di comando MQSC `myprog.in`. Poiché non è stato specificato un nome gestore code, i comandi MQSC vengono eseguiti sul gestore code predefinito. L'output viene inviato al file di testo `myprog.out`. Figura 12 a pagina 78 mostra un'estrazione dal file di comandi MQSC `myprog.in` e Figura 13 a pagina 79 mostra l'estrazione corrispondente dell'output in `myprog.out`.

Per reindirizzare `stdin` e `stdout` sul comando `runmqsc` , per un gestore code (`saturn.queue.manager`) che non è quello predefinito, utilizzare questo formato del comando:

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

File di comandi MQSC

I comandi MQSC sono scritti in formato leggibile, ossia in testo ASCII. Figura 12 a pagina 78 è un estratto da un file di comandi MQSC che visualizza un comando MQSC (DEFINE QLOCAL) con i relativi attributi. Guida di riferimento a MQSC contiene una descrizione di ciascun comando MQSC e la sua sintassi.

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

Figura 12. Estrai da un file di comandi MQSC

Per la portabilità tra ambienti WebSphere MQ , limitare la lunghezza della linea nei file di comandi MQSC a 72 caratteri. Il segno più indica che il comando continua sulla riga successiva.

Report di comandi MQSC

Il comando **runmqsc** restituisce un report , che viene inviato a `stdout`. Il report contiene:

- Un'intestazione che identifica i comandi MQSC come origine del report:

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

  Dove `jupiter.queue.manager` è il nome del gestore code.
- Un elenco numerato facoltativo dei comandi MQSC emessi. Per impostazione predefinita, il testo dell'input viene ripetuto all'output. All'interno di questo output, ogni comando è preceduto da un numero di sequenza, come mostrato in Figura 13 a pagina 79. Tuttavia, è possibile utilizzare l'indicatore -e nel comando `runmqsc` per eliminare l'output.
- Un messaggio di errore di sintassi per tutti i comandi trovati in errore.
- Un *messaggio operatore* che indica il risultato dell'esecuzione di ciascun comando. Ad esempio, il messaggio dell'operatore per il corretto completamento di un comando DEFINE QLOCAL è:

  ```
  AMQ8006: WebSphere MQ queue created.
  ```
- Altri messaggi risultanti da errori generali durante l'esecuzione del file script.
- Un breve riepilogo statistico del report che indica il numero di comandi letti, il numero di comandi con errori di sintassi e il numero di comandi che non è stato possibile elaborare.

Nota: Il gestore code tenta di elaborare solo i comandi senza errori di sintassi.

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

Figura 13. Estrai da un file di report di comandi MQSC

Esecuzione dei file di comandi MQSC forniti

I seguenti file di comandi MQSC vengono forniti con WebSphere WebSphere MQ:

amqscos0.tst
Definizioni di oggetti utilizzati da programmi di esempio.

amqscic0.tst
Definizioni di code per transazioni CICS .

In WebSphere MQ per Windows, questi file si trova nella directory *MQ_INSTALLATION_PATH*\tools\mqsc\samples. *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Sui sistemi UNIX and Linux questi file si trovano nella directory *MQ_INSTALLATION_PATH*/samp. *MQ_INSTALLATION_PATH* rappresenta la directory di alto livello in cui è installato WebSphere MQ .

Il comando che li esegue è:

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

Utilizzo di runmqsc per verificare i comandi

È possibile utilizzare il comando runmqsc per verificare i comandi MQSC su un gestore code locale senza eseguirli effettivamente. A tale scopo, impostare l'indicatore -v nel comando runmqsc , ad esempio:

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

Quando si richiama runmqsc rispetto a un file di comandi MQSC, il gestore code verifica ciascun comando e restituisce un report senza eseguire effettivamente i comandi MQSC. Ciò consente di controllare la sintassi dei comandi nel proprio file di comandi. Ciò è particolarmente importante se:

- Esecuzione di un numero elevato di comandi da un file di comandi.
- Utilizzo di un file di comandi MQSC più volte.

Il report restituito è simile a quello mostrato nella Figura 13 a pagina 79.

Non è possibile utilizzare questo metodo per verificare i comandi MQSC in remoto. Ad esempio, se si tenta questo comando:

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

l'indicatore -w, utilizzato per indicare che il gestore code è remoto, viene ignorato e il comando viene eseguito localmente in modalità di verifica. 30 è il numero di secondi che WebSphere MQ attende per ricevere risposte dal gestore code remoto.

Esecuzione di comandi MQSC da file batch

Se si dispone di comandi molto lunghi o si sta utilizzando una particolare sequenza di comandi ripetutamente, considerare il reindirizzamento di `stdin` da un file batch.

Per reindirizzare `stdin` da un file batch, creare prima un file batch contenente i comandi MQSC utilizzando il normale editor di testo. Quando si usa il comando `runmqsc` , utilizzare gli operatori di reindirizzamento. Il seguente esempio:

1. Crea un gestore code di prova, TESTQM
2. Crea una serie di listener e CLNTCONN corrispondente per utilizzare la porta TCP/IP 1600
3. Crea una coda di verifica, TESTQ
4. Inserisce un messaggio nella coda, utilizzando il programma di esempio amqsputc

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

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

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

amqsputc TESTQ $MYTEMPQM << EOF
hello world
EOF

endmqm -i $MYTEMPQM
```

Figura 14. Script di esempio per l'esecuzione di comandi MQSC da un file batch

Risoluzione dei problemi con i comandi MQSC

Se non è possibile richiamare i comandi MQSC da eseguire, utilizzare le informazioni contenute in questo argomento per verificare se uno di questi problemi comuni si applica all'utente. Non è sempre ovvio quale sia il problema quando si legge l'errore generato da un comando.

Se non è possibile richiamare i comandi MQSC da eseguire, utilizzare le seguenti informazioni per verificare se uno di questi problemi comuni si applica all'utente. Non è sempre ovvio qual è il problema quando si legge l'errore generato.

Quando si utilizza il comando `runmqsc` , tenere presente quanto segue:

- Utilizzare l'operatore **<** per reindirizzare l'input da un file. Se si omette questo operatore, il gestore code interpreta il nome file come un nome gestore code ed emette il seguente messaggio di errore:

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

- Se si reindirizza l'output a un file, utilizzare l'operatore di reindirizzamento **>** . Per impostazione predefinita, il file viene inserito nella directory di lavoro corrente al momento del richiamo di `runmqsc` . Specificare un nome file completo per inviare l'output a un file e directory specifici.
- Verificare di aver creato il gestore code che eseguirà i comandi, utilizzando il seguente comando per visualizzare tutti i gestori code:

```
dspmq
```

- Il gestore code deve essere in esecuzione. In caso contrario, avviarlo; (fare riferimento a Avvio di un gestore code). Se si tenta di avviare un gestore code già in esecuzione, viene visualizzato un messaggio di errore.
- Specificare un nome gestore code nel comando `runmqsc` se non è stato definito un gestore code predefinito o se si riceve questo errore:

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

- Non è possibile specificare un comando MQSC come parametro del comando `runmqsc` . Ad esempio, non è valido:

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

- Non è possibile immettere comandi MQSC prima di immettere il comando `runmqsc` .
- Non è possibile eseguire comandi di controllo da `runmqsc`. Ad esempio, non è possibile immettere il comando `strmqm` per avviare un gestore code mentre si stanno eseguendo i comandi MQSC in modo interattivo. Se si esegue questa operazione, si ricevono messaggi di errore simili ai seguenti:

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

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

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

Uso dei gestori code

Esempi di comandi MQSC che è possibile utilizzare per visualizzare o modificare gli attributi del gestore code.

Visualizzazione degli attributi del gestore code

Per visualizzare gli attributi del gestore code specificato sul comando **runmqsc** , utilizzare il seguente comando MQSC:

```
DISPLAY QMGR
```

L'output tipico di questo comando è mostrato in Figura 15 a pagina 82

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

Figura 15. Emissione tipica da un comando DISPLAY QMGR

Il parametro ALL è il valore predefinito sul comando DISPLAY QMGR . Visualizza tutti gli attributi del gestore code. In particolare, l'output indica il nome del gestore code predefinito, il nome della coda di messaggi non recapitabili e il nome della coda comandi.

È possibile confermare l'esistenza di queste code immettendo il seguente comando:

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

Visualizza un elenco di code che corrispondono alla radice SYSTEM.*. Le parentesi sono obbligatorie.

Modifica degli attributi del gestore code

Per modificare gli attributi del gestore code specificato sul comando **runmqsc** , utilizzare il comando MQSC ALTER QMGR, specificando gli attributi e valori che si desidera modificare. Ad esempio, utilizzare i comandi seguenti per modificare gli attributi di `jupiter.queue.manager`:

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

Il comando ALTER QMGR modifica la coda di messaggi non recapitabili utilizzata e abilita gli eventi di inibizione.

Riferimenti correlati

Attributi per il gestore code

Gestione delle code locali

Questa sezione contiene esempi di alcuni comandi MQSC che è possibile utilizzare per gestire code locali, modello e alias.

Consultare Guida di riferimento a MQSC per informazioni dettagliate su questi comandi.

Definizione di una coda locale

Per un'applicazione, il gestore code locale è il gestore code a cui è connessa l'applicazione. Le code gestite dal gestore code locale vengono dette locali per tale gestore code.

Utilizzare il comando MQSC DEFINE QLOCAL per creare una coda locale. È anche possibile utilizzare il valore predefinito definito nella definizione della coda locale predefinita oppure è possibile modificare le caratteristiche della coda da quelle della coda locale predefinita.

Nota: La coda locale predefinita è denominata SYSTEM.DEFAULT.LOCAL.QUEUE ed è stata creata sull'installazione del sistema.

Ad esempio, il comando DEFINE QLOCAL che segue definisce una coda denominata ORANGE.LOCAL.QUEUE con queste caratteristiche:

- È abilitato per le ricezioni, abilitato per gli inserimenti e opera su una base di ordine di priorità.
- Si tratta di una coda *normale* ; non è una coda di iniziazione o di trasmissione e non genera messaggi trigger.
- La grandezza massima della coda è 5000 messaggi; la lunghezza massima del messaggio è 4194304 byte.

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

Nota:

1. Ad eccezione del valore per la descrizione, tutti i valori di attributo visualizzati sono i valori predefiniti. Li abbiamo mostrati qui a scopo illustrativo. È possibile ometterli se si è certi che i valori predefiniti sono quelli desiderati o non sono stati modificati. Vedi anche “Visualizzazione degli attributi dell'oggetto predefiniti” a pagina 84.
2. USAGE (NORMAL) indica che questa coda non è una coda di trasmissione.
3. Se si dispone già di una coda locale sullo stesso gestore code con il nome ORANGE.LOCAL.QUEUE, questo comando ha esito negativo. Utilizzare l'attributo REPLACE se si desidera sovrascrivere la

definizione esistente di una coda, ma consultare anche “Modifica degli attributi della coda locale” a pagina 85.

Definizione di una coda di messaggi non recapitabili

Ogni gestore code deve avere una coda locale da utilizzare come coda di messaggi non recapitabili in modo che i messaggi che non possono essere consegnati alla destinazione corretta possano essere memorizzati per un successivo recupero. È necessario informare il gestore code della coda di messaggi non recapitabili.

Per informare il gestore code della coda di messaggi non recapitabili, specificare un nome di coda di messaggi non recapitabili nel comando `crtmqm` (`crtmqm -u DEAD.LETTER.QUEUE`, ad esempio) oppure utilizzando l'attributo DEADQ nel comando ALTER QMGR per specificarne uno in un secondo momento. È necessario definire la coda di messaggi non recapitabili prima di utilizzarla.

Una coda di messaggi non recapitabili di esempio denominata SYSTEM.DEAD.LETTER.QUEUE è disponibile con il prodotto. Questa coda viene creata automaticamente quando si crea il gestore code. Se necessario, è possibile modificare questa definizione e ridenominarla.

Una coda di messaggi non recapitabili non ha requisiti speciali tranne che:

- Deve essere una coda locale
- Il suo attributo MAXMSGL (lunghezza massima del messaggio) deve abilitare la coda ad accogliere i messaggi più grandi che il gestore code deve gestire **più** la dimensione dell'intestazione dei messaggi non instradabili (MQDLH)

WebSphere MQ fornisce un gestore code di messaggi non instradabili che consente di specificare il modo in cui i messaggi trovati in una coda di messaggi non instradabili devono essere elaborati o rimossi. Per ulteriori informazioni, consultare Gestione dei messaggi non recapitati con il gestore code di messaggi non recapitabili WebSphere MQ .

Visualizzazione degli attributi dell'oggetto predefiniti

È possibile utilizzare il comando DISPLAY QUEUE per visualizzare gli attributi che sono stati presi dall'oggetto predefinito quando è stato definito un oggetto WebSphere MQ .

Quando si definisce un oggetto WebSphere MQ , vengono utilizzati gli attributi non specificati dall'oggetto predefinito. Ad esempio, quando si definisce una coda locale, la coda eredita tutti gli attributi omessi nella definizione dalla coda locale predefinita, denominata SYSTEM.DEFAULT.LOCAL.QUEUE. Per vedere esattamente quali sono questi attributi, utilizzare il seguente comando:

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

La sintassi di questo comando è diversa da quella del comando DEFINE corrispondente. Nel comando DISPLAY è possibile fornire solo il nome della coda, mentre nel comando DEFINE è necessario specificare il tipo di coda, ovvero QLOCAL, QALIAS, QMODEL o QREMOTE.

È possibile visualizzare in modo selettivo gli attributi specificandoli singolarmente. Ad esempio:

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

Questo comando visualizza i tre attributi specificati nel modo seguente:

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

CURDEPTH è la profondità della coda corrente, ovvero il numero di messaggi sulla coda. Questo è un attributo utile da visualizzare, perché monitorando la profondità della coda è possibile assicurarsi che la coda non diventi piena.

Copia di una definizione di coda locale

È possibile copiare una definizione di coda utilizzando l'attributo LIKE nel comando DEFINE.

Ad esempio:

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

Questo comando crea una coda con gli stessi attributi della coda originale ORANGE.LOCAL.QUEUE, piuttosto che quelle della coda locale predefinita del sistema. Immettere il nome della coda da copiare ***esattamente*** come è stata immessa al momento della creazione della coda. Se il nome contiene caratteri minuscoli, racchiuderlo tra virgolette singole.

È anche possibile utilizzare questo formato del comando DEFINE per copiare una definizione di coda, ma sostituire una o più modifiche agli attributi dell'originale. Ad esempio:

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

Questo comando copia gli attributi della coda ORANGE.LOCAL.QUEUE alla coda THIRD.QUEUE, ma specifica che la lunghezza massima del messaggio nella nuova coda deve essere di 1024 byte, invece di 4194304.

Nota:

1. Quando si utilizza l'attributo LIKE in un comando DEFINE, si copiano solo gli attributi della coda. Non si stanno copiando i messaggi sulla coda.
2. Se si definisce una coda locale, senza specificare LIKE, è uguale a DEFINE LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE).

Modifica degli attributi della coda locale

È possibile modificare gli attributi della coda in due modi, utilizzando il comando ALTER QLOCAL o il comando DEFINE QLOCAL con l'attributo REPLACE.

In “Definizione di una coda locale” a pagina 83, la coda denominata ORANGE.LOCAL.QUEUE è stato definito. Si supponga, ad esempio, di voler diminuire la lunghezza massima del messaggio su questa coda a 10.000 byte.

- Utilizzando il comando ALTER:

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

  Questo comando modifica un singolo attributo, quello della lunghezza massima del messaggio; tutti gli altri attributi rimangono uguali.
- Utilizzando il comando DEFINE con l'opzione REPLACE, ad esempio:

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

  Questo comando modifica non solo la lunghezza massima del messaggio, ma anche tutti gli altri attributi, ai quali vengono assegnati valori predefiniti. La coda è ora abilitata all'inserimento, mentre in precedenza era inibita. L'inserimento abilitato è il valore predefinito, come specificato dalla coda SYSTEM.DEFAULT.LOCAL.QUEUE.

Se si ***riduce*** la lunghezza massima dei messaggi su una coda esistente, i messaggi esistenti non vengono influenzati. Qualsiasi nuovo messaggio, tuttavia, deve soddisfare i nuovi criteri.

Cancellazione di una coda locale

È possibile utilizzare il comando CLEAR per cancellare una coda locale.

Per eliminare tutti i messaggi da una coda locale denominata MAGENTA.QUEUE, utilizzare il seguente comando:

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

Nota: Non vi è alcuna richiesta che consente di cambiare idea; una volta premuto il tasto Invio, i messaggi vengono persi.

Non è possibile cancellare una coda se:

- Ci sono messaggi senza commit che sono stati inseriti nella coda nel punto di sincronizzazione.
- Un'applicazione ha attualmente la coda aperta.

Eliminazione di una coda locale

È possibile utilizzare il comando MQSC DELETE QLOCAL per eliminare una coda locale.

Non è possibile eliminare una coda se contiene messaggi di cui non è stato eseguito il commit. Tuttavia, se la coda ha uno o più messaggi di cui è stato eseguito il commit e nessun messaggio di cui non è stato eseguito il commit, può essere eliminata solo se si specifica l'opzione PURGE. Ad esempio:

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

Specificando NOPURGE invece di PURGE si garantisce che la coda non venga eliminata se contiene messaggi di cui è stato eseguito il commit.

Ricerca code

WebSphere MQ fornisce un browser della coda di esempio che è possibile utilizzare per esaminare il contenuto dei messaggi su una coda. Il browser viene fornito sia in formato origine che in formato eseguibile.

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

In WebSphere MQ per Windows, i percorsi e i nomi file del browser della coda di esempio sono i seguenti:

Sorgente
MQ_INSTALLATION_PATH\tools\c\samples\

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

In WebSphere MQ per UNIX and Linux, i percorsi e i nomi file sono i seguenti:

Sorgente
MQ_INSTALLATION_PATH/samp/amqsbcg0.c

Eseguibile
MQ_INSTALLATION_PATH/samp/bin/amqsbcg

L'esempio richiede due parametri di input, il nome coda e il nome gestore code. Ad esempio:

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

I risultati tipici di questo comando sono riportati in Figura 16 a pagina 87.

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

 MQOPEN - 'SYSTEM.ADMIN.QMGR.EVENT'


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

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

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

****   Message      ****

 length - 104 bytes

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

 No more messages
 MQCLOSE
 MQDISC
```

Figura 16. Risultati tipici dal browser della coda

Abilitazione di code grandi

IBM WebSphere MQ supporta code più grandi di 2 GB.

Sui sistemi Windows , il supporto per file di grandi dimensioni è disponibile senza alcuna ulteriore abilitazione. Sui sistemi AIX, HP-UX, Linuxe Solaris , è necessario abilitare esplicitamente il supporto file di grandi dimensioni prima di poter creare file di coda superiori a 2 GB. Per informazioni su come eseguire questa operazione, consultare la documentazione del sistema operativo.

Alcuni programmi di utilità, come tar, non sono in grado di gestire file superiori a 2 GB. Prima di attivare il supporto file di grandi dimensioni, consultare la documentazione del sistema operativo per informazioni sulle restrizioni relative ai programmi di utilità utilizzati.

Per informazioni sulla pianificazione della quantità di memoria necessaria per le code, consultare il sito Web IBM WebSphere MQ per i report delle prestazioni specifici della piattaforma:

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

Gestione delle code alias

È possibile definire una coda alias per fare riferimento indirettamente ad un'altra coda o argomento.

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.

La coda a cui fa riferimento una coda alias può essere una delle seguenti:

- Una coda locale (consultare “Definizione di una coda locale” a pagina 83).
- Una definizione locale di una coda remota (consultare “Creazione di una definizione locale di una coda remota” a pagina 112).
- Un argomento.

Una coda alias non è una coda reale, ma una definizione che si risolve in una coda reale (o di destinazione) al runtime. La definizione della coda alias specifica la coda di destinazione. Quando un'applicazione effettua una chiamata MQOPEN a una coda alias, il gestore code risolve l'alias nel nome coda di destinazione.

Una coda alias non può essere risolta in un'altra coda alias definita localmente. Tuttavia, una coda alias può risolversi in code alias definite altrove nei cluster di cui è membro il gestore code locale. Per ulteriori informazioni, consultare Risoluzione dei nomi .

Le code alias sono utili per:

- Fornire alle diverse applicazioni diversi livelli di autorizzazioni di accesso alla coda di destinazione.
- Consentire a diverse applicazioni di gestire la stessa coda in modi diversi. (È possibile che si desideri assegnare priorità predefinite differenti o valori di persistenza predefiniti differenti.)
- Semplificazione della manutenzione, della migrazione e del bilanciamento del carico di lavoro. (Forse si desidera modificare il nome della coda di destinazione senza dover modificare l'applicazione, che continua a utilizzare l'alias.)

Ad esempio, si supponga che un'applicazione sia stata sviluppata per inserire i messaggi su una coda denominata MY.ALIAS.QUEUE. Specifica il nome di questa coda quando effettua una richiesta MQOPEN e, indirettamente, se inserisce un messaggio in questa coda. L'applicazione non è consapevole che la coda è una coda alias. Per ogni chiamata MQI che utilizza questo alias, il gestore code risolve il nome della coda reale, che potrebbe essere una coda locale o una coda remota definita su questo gestore code.

Modificando il valore dell'attributo TARGET, è possibile reindirizzare chiamate MQI a un'altra coda, possibilmente su un altro gestore code. È utile per la manutenzione, la migrazione e il bilanciamento del carico.

Definizione di una coda alias

Il seguente comando crea una coda alias:

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

Questo comando reindirizza le chiamate MQI che specificano MY.ALIAS.QUEUE alla coda YELLOW.QUEUE. Il comando non crea la coda di destinazione; le chiamate MQI hanno esito negativo se la coda è YELLOW.QUEUE non esiste in fase di runtime.

Se si modifica la definizione dell'alias, è possibile reindirizzare le chiamate MQI a un'altra coda. Ad esempio:

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

Questo comando reindirizza le chiamate MQI a un'altra coda, MAGENTA.QUEUE.

È inoltre possibile utilizzare le code alias per fare in modo che una singola coda (la coda di destinazione) sembri avere attributi differenti per applicazioni differenti. Questa operazione viene eseguita definendo due alias, uno per ogni applicazione. Si supponga che vi siano due applicazioni:

- L'applicazione ALPHA può inserire messaggi su YELLOW.QUEUE, ma non è consentito richiamare messaggi da esso.
- L'applicazione BETA può ricevere messaggi da YELLOW.QUEUE, ma non è consentito inserire messaggi su di esso.

Il seguente comando definisce un alias che viene inserito abilitato e viene disabilitato per l'applicazione ALPHA:

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

Il seguente comando definisce un alias che viene inserito disabilitato e abilitato per l'applicazione BETA:

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

ALPHA utilizza il nome coda ALPHAS.ALIAS.QUEUE nelle sue chiamate MQI; BETA utilizza il nome della coda BETAS.ALIAS.QUEUE. Entrambi accedono alla stessa coda, ma in modi diversi.

È possibile utilizzare gli attributi LIKE e REPLACE quando si definiscono gli alias della coda, nello stesso modo in cui si utilizzano questi attributi con code locali.

Utilizzo di altri comandi con code alias

È possibile utilizzare i comandi MQSC appropriati per la visualizzazione o la modifica degli attributi della coda alias o per eliminare l'oggetto della coda alias. Ad esempio:

Utilizzare il seguente comando per visualizzare gli attributi della coda alias:

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

Utilizzare il seguente comando per modificare il nome della coda di base, in cui l'alias si risolve, in cui l'opzione `force` forza la modifica anche se la coda è aperta:

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

Utilizzare il comando riportato di seguito per eliminare questo alias della coda:

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

Non è possibile eliminare una coda alias se un'applicazione ha attualmente la coda aperta. Consultare Guida di riferimento a MQSC per ulteriori informazioni su questo e altri comandi della coda alias.

Utilizzo delle code modello

Un gestore code crea una *coda dinamica* se riceve una chiamata MQI da un'applicazione che specifica un nome coda definito come coda modello. Il nome della nuova coda dinamica viene generato dal gestore code quando viene creata la coda. Una *coda modello* è un modello che specifica gli attributi di tutte le code dinamiche da essa create. Le code modello forniscono un metodo conveniente per le applicazioni per creare le code come richiesto.

Definizione di una coda modello

Si definisce una coda modello con un insieme di attributi nello stesso modo in cui si definisce una coda locale. Le code modello e le code locali hanno la stessa serie di attributi, tranne per il fatto che sulle code modello è possibile specificare se le code dinamiche create sono temporanee o permanenti. Le code permanenti vengono gestite durante i riavvii del gestore code, mentre quelle temporanee non lo sono. Ad esempio:

```
DEFINE QMODEL (GREEN.MODEL.QUEUE) +
       DESCR('Queue for messages from application X') +
       PUT (DISABLED) +
       GET (ENABLED) +
       NOTRIGGER +
       MSGDLVSQ (FIFO) +
       MAXDEPTH (1000) +
       MAXMSGL (2000) +
       USAGE (NORMAL) +
       DEFTYPE (PERMDYN)
```

Questo comando crea una definizione di coda modello. Dall'attributo DEFTYPE, è possibile vedere che le code effettive create da questo modello sono code dinamiche permanenti. Gli attributi non specificati vengono automaticamente copiati da SYSYTEM SYSYTEM.DEFAULT.MODEL.QUEUE predefinita.

È possibile utilizzare gli attributi LIKE e REPLACE quando si definiscono le code modello, nello stesso modo in cui vengono utilizzate con le code locali.

Utilizzo di altri comandi con code modello

È possibile utilizzare i comandi MQSC appropriati per visualizzare o modificare gli attributi di una coda modello o per eliminare l'oggetto coda modello. Ad esempio:

Utilizzare il comando riportato di seguito per visualizzare gli attributi della coda modello:

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

Utilizzare il seguente comando per modificare il modello per abilitare gli inserimenti su qualsiasi coda dinamica creata da questo modello:

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

Utilizzare il seguente comando per eliminare questa coda modello:

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

Utilizzo degli argomenti di gestione

Utilizzare i comandi MQSC per gestire gli argomenti di amministrazione.

Consultare Guida di riferimento a MQSC per informazioni dettagliate su questi comandi.

Concetti correlati

Oggetti argomento di gestione

Definizione di un argomento di amministrazione

Utilizzare il comando MQSC **DEFINE TOPIC** per creare un argomento di gestione. Quando si definisce un argomento di gestione, è possibile impostare facoltativamente ogni attributo dell'argomento.

Qualsiasi attributo dell'argomento non esplicitamente impostato viene ereditato dall'argomento di gestione predefinito, SYSTEM.DEFAULT.TOPIC, che è stato creato quando è stata installata l'installazione del sistema.

Ad esempio, il comando **DEFINE TOPIC** che segue, definisce un argomento denominato **ORANGE.TOPIC** con queste caratteristiche:

- Si risolve nella stringa di argomenti ORANGE. Per informazioni su come utilizzare le stringhe argomento, consultare Combinazione di stringhe argomento.
- Qualsiasi attributo impostato su ASPARENT utilizza l'attributo come definito dall'argomento principale di questo argomento. Questa operazione viene ripetuta nella struttura ad albero degli argomenti fino all'argomento principale, SYSTEM.BASE.TOPIC trovato. Per ulteriori informazioni sulle strutture ad albero degli argomenti, consultare Alberi degli argomenti.

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

Nota:

- Tranne che per il valore della stringa di argomenti, tutti i valori di attributo visualizzati sono i valori predefiniti. Sono mostrati qui solo come un'illustrazione. È possibile ometterli se si è certi che i valori predefiniti sono quelli desiderati o non sono stati modificati. Vedi anche “Visualizzazione degli attributi dell'oggetto argomento di gestione” a pagina 91.
- Se si dispone già di un argomento di gestione sullo stesso gestore code con il nome ORANGE.TOPIC, questo comando non riesce. Utilizzare l'attributo REPLACE se si desidera sovrascrivere la definizione esistente di un argomento, ma consultare anche “Modifica degli attributi dell'argomento di amministrazione” a pagina 92

Visualizzazione degli attributi dell'oggetto argomento di gestione

Utilizzare il comando MQSC **DISPLAY TOPIC** per visualizzare un oggetto argomento di gestione.

Per visualizzare tutti gli argomenti, utilizzare:

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

È possibile visualizzare in modo selettivo gli attributi specificandoli singolarmente.Ad esempio:

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

```
        DEFPRTY +
        NPMSGDLV
```

Questo comando visualizza i tre attributi specificati nel modo seguente:

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

Per visualizzare i valori ASPARENT dell'argomento utilizzati al runtime, utilizzare DISPLAY TPSTATUS. Ad esempio, utilizzare:

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

Il comando visualizza i seguenti dettagli:

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

Quando si definisce un argomento di gestione, vengono utilizzati tutti gli attributi non specificati esplicitamente dall'argomento di gestione predefinito, denominato SYSTEM.DEFAULT.TOPIC. Per visualizzare questi attributi predefiniti, utilizzare il seguente comando:

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

Modifica degli attributi dell'argomento di amministrazione

È possibile modificare gli attributi dell'argomento in due modi, utilizzando il comando **ALTER TOPIC** o il comando **DEFINE TOPIC** con l'attributo **REPLACE** .

Se, ad esempio, si desidera modificare la priorità predefinita dei messaggi consegnati ad un argomento denominato ORANGE.TOPIC, deve essere 5, utilizzare uno dei seguenti comandi.

- Utilizzando il comando **ALTER** :

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

  Questo comando modifica un singolo attributo, quello della priorità predefinita del messaggio consegnato a questo argomento in 5; tutti gli altri attributi rimangono gli stessi.
- Utilizzando il comando **DEFINE** :

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

  Questo comando modifica la priorità predefinita dei messaggi consegnati a questo argomento. A tutti gli altri attributi vengono assegnati valori predefiniti.

Se si modifica la priorità dei messaggi inviati a questo argomento, i messaggi esistenti non vengono influenzati. Qualsiasi nuovo messaggio, tuttavia, utilizza la priorità specificata se non fornita dall'applicazione di pubblicazione.

Copia di una definizione di argomento di gestione

È possibile copiare una definizione di argomento utilizzando l'attributo LIKE sul comando **DEFINE** .

Ad esempio:

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

Questo comando crea un argomento, MAGENTA.TOPIC, con gli stessi attributi dell'argomento originale, ORANGE.TOPIC, piuttosto che quelli dell'argomento di gestione predefinito del sistema. Immettere il

nome dell'argomento da copiare esattamente come è stato immesso quando è stato creato l'argomento. Se il nome contiene caratteri minuscoli, racchiuderlo tra virgolette singole.

È anche possibile utilizzare questo modulo del comando **DEFINE** per copiare una definizione di argomento, ma apportare modifiche agli attributi dell'originale. Ad esempio:

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

È anche possibile copiare gli attributi dell'argomento BLUE.TOPIC all'argomento GREEN.TOPIC e specificare che quando le pubblicazioni non possono essere consegnate alla coda del sottoscrittore corretta, non vengono inserite nella coda di messaggi non recapitabili. Ad esempio:

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

Eliminazione di una definizione di argomento di gestione

È possibile utilizzare il comando MQSC **DELETE TOPIC** per eliminare un argomento di gestione.

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

Le applicazioni non saranno più in grado di aprire l'argomento per la pubblicazione o effettuare nuove sottoscrizioni utilizzando il nome oggetto, ORANGE.TOPIC. Le applicazioni di pubblicazione che hanno l'argomento aperto possono continuare a pubblicare la stringa di argomenti risolta. Le sottoscrizioni già effettuate a questo argomento continuano a ricevere pubblicazioni dopo che l'argomento è stato eliminato.

Le applicazioni che non fanno riferimento a questo oggetto argomento ma che utilizzano la stringa di argomenti risolta rappresentata da questo oggetto argomento, 'ORANGE' in questo esempio, continuano a funzionare. In questo caso, ereditano le proprietà da un oggetto argomento più in alto nella struttura ad albero degli argomenti. Per ulteriori informazioni sulle strutture ad albero degli argomenti, consultare Alberi degli argomenti.

Utilizzo delle sottoscrizioni

Utilizzare i comandi MQSC per gestire le sottoscrizioni.

Le sottoscrizioni possono essere di tre tipi, definiti nell'attributo **SUBTYPE** :

ADMIN
Definito amministrativamente da un utente.

Proxy
Una sottoscrizione creata internamente per instradare le pubblicazioni tra i gestori code.

API
Creato in modo programmatico, ad esempio, utilizzando la chiamata MQI MQSUB.

Consultare Guida di riferimento a MQSC per informazioni dettagliate su questi comandi.

Concetti correlati

“Definizione di una sottoscrizione di gestione” a pagina 94
Utilizzare il comando MQSC **DEFINE SUB** per creare una sottoscrizione di gestione. È anche possibile utilizzare il valore predefinito definito nella definizione della sottoscrizione locale predefinita. Oppure, è possibile modificare le caratteristiche della sottoscrizione da quelle della sottoscrizione locale predefinita, SYSTEM.DEFAULT.SUB che è stato creato quando il sistema è stato installato.

“Visualizzazione degli attributi delle sottoscrizioni” a pagina 94
È possibile utilizzare il comando **DISPLAY SUB** per visualizzare gli attributi configurati di qualsiasi sottoscrizione nota al gestore code.

“Modifica degli attributi della sottoscrizione locale” a pagina 95
È possibile modificare gli attributi di sottoscrizione in due modi, utilizzando il comando **ALTER SUB** o il comando **DEFINE SUB** con l'attributo **REPLACE** .

“Copia di una definizione di sottoscrizione locale” a pagina 96
È possibile copiare una definizione di sottoscrizione utilizzando l'attributo **LIKE** sul comando **DEFINE** .

“Eliminazione di una sottoscrizione” a pagina 96
È possibile utilizzare il comando MQSC **DELETE SUB** per eliminare una sottoscrizione locale.

Definizione di una sottoscrizione di gestione

Utilizzare il comando MQSC **DEFINE SUB** per creare una sottoscrizione di gestione. È anche possibile utilizzare il valore predefinito definito nella definizione della sottoscrizione locale predefinita. Oppure, è possibile modificare le caratteristiche della sottoscrizione da quelle della sottoscrizione locale predefinita, SYSTEM.DEFAULT.SUB che è stato creato quando il sistema è stato installato.

Ad esempio, il comando **DEFINE SUB** che segue definisce una sottoscrizione denominata ORANGE con le seguenti caratteristiche:

- Sottoscrizione durevole, che indica che persiste al riavvio del gestore code, con scadenza illimitata.
- Ricevere le pubblicazioni effettuate sulla stringa di argomenti ORANGE, con le priorità dei messaggi impostate dalle applicazioni di pubblicazione.
- Le pubblicazioni distribuite per questa sottoscrizione vengono inviate alla coda locale SUBQ, questa coda deve essere definita prima della definizione della sottoscrizione.

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

Nota:

- Il nome della sottoscrizione e della stringa di argomenti non devono corrispondere.
- Ad eccezione dei valori della descrizione e della stringa di argomenti, tutti i valori di attributo visualizzati sono i valori predefiniti. Sono mostrati qui solo come un'illustrazione. È possibile ometterli se si è certi che i valori predefiniti sono quelli desiderati o non sono stati modificati. Vedi anche “Visualizzazione degli attributi delle sottoscrizioni” a pagina 94.
- Se si dispone già di una sottoscrizione locale sullo stesso gestore code con il nome TEST, questo comando non riesce. Utilizzare l'attributo **REPLACE** se si desidera sovrascrivere la definizione esistente di una coda, ma consultare anche “Modifica degli attributi della sottoscrizione locale” a pagina 95.
- Se la coda SUBQ non esiste, questo comando ha esito negativo.

Visualizzazione degli attributi delle sottoscrizioni

È possibile utilizzare il comando **DISPLAY SUB** per visualizzare gli attributi configurati di qualsiasi sottoscrizione nota al gestore code.

Ad esempio, utilizzare:

```
DISPLAY SUB (ORANGE)
```

È possibile visualizzare in modo selettivo gli attributi specificandoli singolarmente. Ad esempio:

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

Questo comando visualizza i tre attributi specificati nel modo seguente:

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

TOPICSTR è la stringa argomento risolta su cui questo sottoscrittore è operativo. Quando una richiesta viene definita per utilizzare un oggetto argomento, la stringa di argomento da tale oggetto viene utilizzata come prefisso alla stringa argomento fornita durante l'esecuzione della richiesta. SUBID è un identificativo univoco assegnato dal gestore code quando viene creata una sottoscrizione. Questo è un attributo utile da visualizzare perché alcuni nomi di sottoscrizioni potrebbero essere lunghi o in una serie di caratteri diversa per cui potrebbero diventare impraticabili.

Un metodo alternativo per visualizzare le sottoscrizioni consiste nell'utilizzare il SUBID:

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

Questo comando fornisce lo stesso output di prima:

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

Le sottoscrizioni proxy su un gestore code non vengono visualizzate per default. Per visualizzarli specificare un **SUBTYPE** di PROXY o ALL.

È possibile utilizzare il comando DISPLAY SBSTATUS per visualizzare gli attributi Runtime. Ad esempio, utilizzare il comando:

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

Viene visualizzato il seguente output:

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

Quando si definisce una sottoscrizione di gestione, vengono utilizzati tutti gli attributi non specificati esplicitamente dalla sottoscrizione predefinita, denominata SYSTEM.DEFAULT.SUB. Per visualizzare questi attributi predefiniti, utilizzare il seguente comando:

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

Modifica degli attributi della sottoscrizione locale

È possibile modificare gli attributi di sottoscrizione in due modi, utilizzando il comando **ALTER SUB** o il comando **DEFINE SUB** con l'attributo **REPLACE** .

Se, ad esempio, si desidera modificare la priorità dei messaggi consegnati a una sottoscrizione denominata ORANGE in modo che sia 5, utilizzare uno dei seguenti comandi:

- Utilizzando il comando ALTER:

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

  Questo comando modifica un singolo attributo, quello della priorità dei messaggi consegnati a questa sottoscrizione a 5; tutti gli altri attributi rimangono gli stessi.
- Utilizzando il comando DEFINE:

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

Questo comando modifica non solo la priorità dei messaggi consegnati a questa sottoscrizione, ma anche tutti gli altri attributi a cui vengono assegnati valori predefiniti.

Se si modifica la priorità dei messaggi inviati a questa sottoscrizione, i messaggi esistenti non vengono influenzati. Tutti i nuovi messaggi, tuttavia, hanno la priorità specificata.

Copia di una definizione di sottoscrizione locale

È possibile copiare una definizione di sottoscrizione utilizzando l'attributo **LIKE** sul comando **DEFINE** .

Ad esempio:

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

È anche possibile copiare gli attributi del REAL secondario nel THIRD.SUBe specificare che il correlID delle pubblicazioni fornite è THIRD, piuttosto che il correlIDdei publisher. Ad esempio:

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

Eliminazione di una sottoscrizione

È possibile utilizzare il comando MQSC **DELETE SUB** per eliminare una sottoscrizione locale.

```
DELETE SUB(ORANGE)
```

È anche possibile eliminare una sottoscrizione utilizzando il SUBID:

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

Controllo dei messaggi su una sottoscrizione

Informazioni su questa attività

Quando una sottoscrizione viene definita, viene associata a una coda. I messaggi pubblicati corrispondenti a questa sottoscrizione vengono inseriti in questa coda.

Tenere presente che i seguenti comandi **runmqsc** mostrano solo le sottoscrizioni che hanno ricevuto i messaggi.

Per controllare i messaggi attualmente accodati per una sottoscrizione, effettuare le seguenti operazioni:

Procedura

1. Per controllare i messaggi accodati per un tipo sottoscrizione **DISPLAY SBSTATUS(<sub_name>) NUMMSGS**, consultare “Visualizzazione degli attributi delle sottoscrizioni” a pagina 94.
2. Se il valore di **NUMMSGS** è maggiore di zero, identificare la coda associata alla sottoscrizione immettendo **DISPLAY SUB(<sub_name>)DEST**.
3. Utilizzando il nome della coda restituita è possibile visualizzare i messaggi seguendo la tecnica descritta in “Ricerca code” a pagina 86.

Gestione dei servizi

Gli oggetti di servizio sono un mezzo con cui è possibile gestire ulteriori processi come parte di un gestore code. Con i servizi, è possibile definire programmi che vengono avviati e arrestati all'avvio e alla fine del gestore code. I servizi IBM WebSphere MQ vengono sempre avviati con l'ID utente dell'utente che ha avviato il gestore code.

Gli oggetti di servizio possono essere uno dei seguenti tipi:

Server
Un server è un oggetto di servizio con il parametro SERVTYPE specificato come SERVER. Un oggetto servizio server è la definizione di un programma eseguito quando viene avviato un gestore code specificato. Gli oggetti di servizio del server definiscono i programmi che generalmente vengono eseguiti per un lungo periodo di tempo. Ad esempio, un oggetto servizio server può essere utilizzato per eseguire un processo di controllo trigger, come `runmqtrm`.

Solo un'istanza di un oggetto servizio server può essere eseguita simultaneamente. Lo stato degli oggetti del servizio server in esecuzione può essere monitorato utilizzando il comando MQSC, DISPLAY SVSTATUS.

Comando
Un comando è un oggetto di servizio con il parametro SERVTYPE specificato come COMMAND. Gli oggetti servizio comandi sono simili agli oggetti servizio server, tuttavia più istanze di un oggetto servizio comandi possono essere eseguite contemporaneamente e il loro stato non può essere monitorato utilizzando il comando MQSC DISPLAY SVSTATUS.

Se il comando MQSC, STOP SERVICE, viene eseguito, non viene eseguito alcun controllo per stabilire se il programma avviato dal comando MQSC, START SERVICE, è ancora attivo prima di eseguire il programma di arresto.

Definizione di un oggetto servizio

Si definisce un oggetto servizio con vari attributi.

Gli attributi sono i seguenti:

SERVTYPE
Definisce il tipo di oggetto servizio. I valori possibili sono i seguenti:

SERVER
Un oggetto servizio server.

È possibile eseguire una sola istanza di un oggetto servizio server alla volta. Lo stato degli oggetti di servizio del server può essere monitorato utilizzando il comando MQSC, DISPLAY SVSTATUS.

COMANDO
Un oggetto servizio comandi.

È possibile eseguire contemporaneamente più istanze di un oggetto servizio comandi. Lo stato di un oggetto servizio comandi non può essere monitorato.

STARTCMD
Il programma eseguito per avviare il servizio. È necessario specificare un percorso completo per il programma.

STARTARG
Argomenti passati al programma di avvio.

STDERR
Specifica il percorso di un file a cui reindirizzare l'errore standard (stderr) del programma di servizio.

STDOUT
Specifica il percorso di un file a cui reindirizzare l'emissione standard (stdout) del programma di servizio.

STOPCMD
Il programma eseguito per arrestare il servizio. È necessario specificare un percorso completo per il programma.

STOPARG
Argomenti passati al programma di arresto.

CONTROL
Specifica il modo in cui il servizio deve essere avviato e arrestato:

MANUAL
Il servizio non deve essere avviato automaticamente o arrestato automaticamente. È controllato dall'utilizzo dei comandi START SERVICE e STOP SERVICE. Questo è il valore predefinito.

QMGR
Il servizio da definire deve essere avviato e arrestato contemporaneamente all'avvio e all'arresto del gestore code.

SOLO
Il servizio deve essere avviato contemporaneamente all'avvio del gestore code, ma non è richiesto l'arresto quando il gestore code viene arrestato.

Concetti correlati

"Gestione dei servizi" a pagina 98
Utilizzando il parametro CONTROL, un'istanza di un oggetto servizio può essere avviata e arrestata automaticamente dal gestore code oppure avviata e arrestata utilizzando i comandi MQSC START SERVICE e STOP SERVICE.

Gestione dei servizi

Utilizzando il parametro CONTROL, un'istanza di un oggetto servizio può essere avviata e arrestata automaticamente dal gestore code oppure avviata e arrestata utilizzando i comandi MQSC START SERVICE e STOP SERVICE.

Quando viene avviata un'istanza di un oggetto servizio, viene scritto un messaggio nel log degli errori del gestore code contenente il nome dell'oggetto servizio e l'ID processo del processo avviato. Di seguito è riportata una voce di log di esempio per un oggetto servizio del server in fase di avvio:

```
  02/15/2005 11:54:24 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5028: The Server 'S1' has started. ProcessId(13031).

  EXPLANATION:
  The Server process has started.
  ACTION:
  None.
```

Di seguito è riportata una voce di log di esempio per un oggetto servizio comandi in fase di avvio:

```
  02/15/2005 11:53:55 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5030: The Command 'C1' has started. ProcessId(13030).

  EXPLANATION:
  The Command has started.
  ACTION:
  None.
```

Quando un servizio del server delle istanze si arresta, viene scritto un messaggio nei log degli errori del gestore code contenenti il nome del servizio e l'ID processo del processo finale. Di seguito è riportata una voce di log di esempio per un oggetto servizio del server in fase di arresto:

```
  02/15/2005 11:54:54 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5029: The Server 'S1' has ended. ProcessId(13031).

  EXPLANATION:
  The Server process has ended.
  ACTION:
  None.
```

Riferimenti correlati

"Variabili di ambiente aggiuntive" a pagina 99

Quando un servizio viene avviato, l'ambiente in cui viene avviato il processo del servizio viene ereditato dall'ambiente del gestore code. È possibile definire ulteriori variabili di ambiente da impostare nell'ambiente del processo del servizio aggiungendo le variabili che si desidera definire a uno dei file di sovrascrittura dell'ambiente `service.env` .

Variabili di ambiente aggiuntive

Quando un servizio viene avviato, l'ambiente in cui viene avviato il processo del servizio viene ereditato dall'ambiente del gestore code. È possibile definire ulteriori variabili di ambiente da impostare nell'ambiente del processo del servizio aggiungendo le variabili che si desidera definire a uno dei file di sovrascrittura dell'ambiente `service.env` .

Nota:

Esistono due possibili file a cui è possibile aggiungere le variabili di ambiente:

- Il file `service.env` dell'ambito della macchina, che si trova in `/var/mqm` su sistemi UNIX and Linux o nella directory di dati selezionata durante l'installazione su sistemi Windows .
- Il file `service.env` dell'ambito gestore code, ubicato nella directory dei dati del gestore code. Ad esempio, l'ubicazione del file di sovrascrittura ambiente per un gestore code denominato QMNAME è:
  - Su sistemi UNIX and Linux , `/var/mqm/qmgrs/QMNAME/service.env`
  - Su sistemi Windows , `C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\service.env`

Entrambi i file vengono elaborati, se disponibili, con le definizioni nel file di ambito del gestore code che hanno la precedenza su quelle nel file di ambito della macchina.

Qualsiasi variabile di ambiente può essere specificata in `service.env`. Ad esempio, se il servizio IBM WebSphere MQ esegue un numero di comandi, potrebbe essere utile impostare la variabile utente PATH nel file `service.env` . I valori impostati sulla variabile non possono essere variabili di ambiente; ad esempio CLASSPATH=*%CLASSPATH%* non è corretto. Allo stesso modo, su Linux PATH=*$PATH*`:/opt/mqm/bin` si ottengono risultati non previsti.

CLASSPATH deve essere maiuscolo e l'istruzione del percorso classe può contenere solo valori letterali. Alcuni servizi (ad esempio Telemetria) impostano il proprio percorso di classe. Il CLASSPATH definito in `service.env` viene aggiunto ad esso.

Il formato delle variabili definite nel file,`service.env` è un elenco di coppie di variabili nome e valore. Ogni variabile deve essere definita su una nuova linea e ogni variabile viene presa come è esplicitamente definita, incluso lo spazio. Di seguito è riportato un esempio del file,`service.env` :

```
#********************************************************************#
#*                                                                  *#
#* <N_OCO_COPYRIGHT>                                                *#
#* Licensed Materials - Property of IBM                             *#
#*                                                                  *#
#* 63H9336                                                          *#
#* (C) Copyright IBM Corporation 2005, 2024.                        *#
#*                                                                  *#
#* <NOC_COPYRIGHT>                                                  *#
#*                                                                  *#
#********************************************************************#
#********************************************************************#
#* Module Name: service.env                                         *#
#* Type       : WebSphere MQ service environment file               *#
#* Function   : Define additional environment variables to be set   *#
#*              for SERVICE programs.                               *#
#* Usage      : <VARIABLE>=<VALUE>                                  *#
#*                                                                  *#
#********************************************************************#
MYLOC=/opt/myloc/bin
MYTMP=/tmp
TRACEDIR=/tmp/trace
MYINITQ=ACCOUNTS.INITIATION.QUEUE
```

Riferimenti correlati

“Inserimenti sostituibili sulle definizioni di servizio” a pagina 100

Nella definizione di un oggetto di servizio, è possibile sostituire i token. I token sostituiti vengono sostituiti automaticamente con il relativo testo espanso quando viene eseguito il programma di servizio. I token di sostituzione possono essere presi dal seguente elenco di token comuni o da qualsiasi variabile definita nel file, service.env.

Inserimenti sostituibili sulle definizioni di servizio

Nella definizione di un oggetto di servizio, è possibile sostituire i token. I token sostituiti vengono sostituiti automaticamente con il relativo testo espanso quando viene eseguito il programma di servizio. I token di sostituzione possono essere presi dal seguente elenco di token comuni o da qualsiasi variabile definita nel file, service.env.

I seguenti sono token comuni che possono essere utilizzati per sostituire i token nella definizione di un oggetto servizio:

PERCORSO MQ_INSTALL_
L'ubicazione in cui è installato WebSphere MQ .

MQ_DATA_PATH
L'ubicazione della directory di dati WebSphere MQ :

- Sui sistemi UNIX and Linux , l'ubicazione della directory di dati WebSphere MQ è `/var/mqm/`
- Su sistemi Windows , l'ubicazione della directory di dati WebSphere MQ è la directory di dati selezionata durante l'installazione di WebSphere MQ

QMNAME
Il nome del gestore code corrente.

NOME_SERVIZIO_MQ
Il nome del servizio.

PID MQ_SERVER_
Questo token può essere utilizzato solo dagli argomenti STOPARG e STOPCMD.

Per gli oggetti servizio del server, questo token viene sostituito con l'id processo del processo avviato dagli argomenti STARTCMD e STARTARG. Altrimenti, questo token viene sostituito con 0.

MQ_Q_MGR_DATA_PATH
L'ubicazione della directory dei dati del gestore code.

MQ_Q_MGR_DATA_NAME
Il nome trasformato del gestore code. Per ulteriori informazioni sulla trasformazione dei nomi, consultare Understanding WebSphere MQ file names.

Per utilizzare inserimenti sostituibili, inserire il token all'interno dei caratteri + in una delle stringhe STARTCMD, STARTARG, STOPCMD, STOPARG, STDOUT o STDERR. Per esempi, consultare “Esempi sull'utilizzo di oggetti di servizio” a pagina 100.

Esempi sull'utilizzo di oggetti di servizio

I servizi in questa sezione sono scritti con caratteri separatori di percorso in stile UNIX , tranne dove diversamente indicato.

Utilizzo di un oggetto servizio server

Questo esempio mostra come definire, utilizzare e modificare un oggetto servizio server per avviare un controllo trigger.

1. Un oggetto servizio server è definito, utilizzando il comando MQSC riportato di seguito:

```
DEFINE SERVICE(S1) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('+MQ_INSTALL_PATH+bin/runmqtrm') +
       STARTARG('-m +QMNAME+ -q ACCOUNTS.INITIATION.QUEUE') +
```

```
       STOPCMD('+MQ_INSTALL_PATH+bin/amqsstop') +
       STOPARG('-m +QMNAME+ -p +MQ_SERVER_PID+')
```

Dove:

+MQ_INSTALL_PATH+ è un token che rappresenta la directory di installazione.

+QMNAME+ è un token che rappresenta il nome del gestore code.

ACCOUNTS.INITIATION.QUEUE è la coda di avvio.

amqsstop è un programma di esempio fornito con WebSphere MQ che richiede al gestore code di interrompere tutte le connessioni per l'ID processo. amqsstop genera comandi PCF, quindi il server dei comandi deve essere in esecuzione.

+MQ_SERVER_PID+ è un token che indica l'ID processo passato al programma di arresto.

Consultare “Inserimenti sostituibili sulle definizioni di servizio” a pagina 100 per un elenco dei token comuni.

2. Un'istanza dell'oggetto servizio del server verrà eseguita al successivo avvio del gestore code. Tuttavia, verrà avviata un'istanza dell'oggetto servizio del server immediatamente con il seguente comando MQSC:

```
START SERVICE(S1)
```

3. Viene visualizzato lo stato del processo di servizio del server, utilizzando il seguente comando MQSC:

```
DISPLAY SVSTATUS(S1)
```

4. Questo esempio mostra ora come modificare l'oggetto servizio del server e fare in modo che gli aggiornamenti siano rilevati riavviando manualmente il processo servizio del server. L'oggetto servizio del server viene modificato in modo che la coda di iniziazione sia specificata come JUPITER.INITIATION.QUEUE . Viene utilizzato il seguente comando MQSC:

```
ALTER SERVICE(S1) +
      STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

Nota: Un servizio in esecuzione non ritira alcun aggiornamento alla propria definizione di servizio fino a quando non viene riavviato.

5. Il processo di servizio del server viene riavviato in modo che la modifica venga rilevata, utilizzando i seguenti comandi MQSC:

```
STOP SERVICE(S1)
```

seguito da:

```
START SERVICE(S1)
```

Il processo di servizio del server viene riavviato e prende le modifiche apportate in “4” a pagina 101.

Nota: Il comando MQSC, STOP SERVICE, può essere utilizzato solo se viene specificato un argomento STOPCMD nella definizione del servizio.

Utilizzo di un oggetto servizio comandi

Questo esempio mostra come definire un oggetto servizio comandi per avviare un programma che scrive le voci nel log di sistema del sistema operativo quando un gestore code viene avviato o arrestato.

1. L'oggetto del servizio comandi viene definito, utilizzando il seguente comando MQSC:

```
DEFINE SERVICE(S2) +
       CONTROL(QMGR) +
       SERVTYPE(COMMAND) +
       STARTCMD('/usr/bin/logger') +
```

```
       STARTARG('Queue manager +QMNAME+ starting') +
       STOPCMD('/usr/bin/logger') +
       STOPARG('Queue manager +QMNAME+ stopping')
```

Dove:

`logger` è il UNIX and Linux comando fornito dal sistema per scrivere nel log di sistema.

+QMNAME+ è un token che rappresenta il nome del gestore code.

Utilizzo di un oggetto servizio comandi solo quando un gestore code termina

Questo esempio mostra come definire un oggetto servizio comandi per avviare un programma che scrive le voci nel log di sistema del sistema operativo solo quando un gestore code viene arrestato.

1. L'oggetto del servizio comandi viene definito, utilizzando il seguente comando MQSC:

```
DEFINE SERVICE(S3) +
       CONTROL(QMGR) +
       SERVTYPE(COMMAND) +
       STOPCMD('/usr/bin/logger') +
       STOPARG('Queue manager +QMNAME+ stopping')
```

Dove:

`logger` è un programma di esempio fornito con WebSphere MQ che può scrivere voci nel log di sistema del sistema operativo.

+QMNAME+ è un token che rappresenta il nome del gestore code.

Ulteriori informazioni sul passaggio di argomenti

Questo esempio illustra come definire un oggetto servizio server per avviare un programma denominato `runserv` quando viene avviato un gestore code.

Questo esempio viene scritto con i caratteri di separazione del percorso di stile Windows .

Uno degli argomenti da passare al programma iniziale è una stringa contenente uno spazio. Questo argomento deve essere passato come una singola stringa. A tal fine, le virgolette doppie vengono utilizzate come mostrato nel seguente comando per definire l'oggetto servizio comando:

1. L'oggetto del servizio server è definito, utilizzando il comando MQSC riportato di seguito:

```
   DEFINE SERVICE(S1) SERVTYPE(SERVER) CONTROL(QMGR) +
     STARTCMD('C:\Program Files\Tools\runserv.exe') +
     STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
     STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')


DEFINE SERVICE(S4) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('C:\Program Files\Tools\runserv.exe') +
       STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
       STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')
```

Dove:

+QMNAME+ è un token che rappresenta il nome del gestore code.

`"C:\Program Files\Tools\"` è una stringa contenente uno spazio, che verrà passato come una singola stringa.

Avvio automatico di un servizio

Questo esempio mostra come definire un oggetto servizio del server che può essere utilizzato per avviare automaticamente il Controllo trigger all'avvio del gestore code.

1. L'oggetto del servizio server è definito, utilizzando il comando MQSC riportato di seguito:

```
DEFINE SERVICE(TRIG_MON_START) +
```

```
        CONTROL(QMGR) +
        SERVTYPE(SERVER) +
        STARTCMD('runmqtrm') +
        STARTARG('-m +QMNAME+ -q +IQNAME+')
```

Dove:

+QMNAME+ è un token che rappresenta il nome del gestore code.

+IQNAME+ è una variabile di ambiente definita dall'utente in uno dei file service.env che rappresenta il nome della coda di iniziazione.

Gestione degli oggetti per il trigger

WebSphere MQ consente di avviare un'applicazione automaticamente quando vengono soddisfatte determinate condizioni su una coda. Ad esempio, si potrebbe voler avviare un'applicazione quando il numero di messaggi su una coda raggiunge un numero specificato. Questa funzione è denominata *attivazione*. È necessario definire gli oggetti che supportano il trigger.

Trigger descritto in dettaglio in Avvio di applicazioni WebSphere MQ utilizzando i trigger.

Definizione di una coda dell'applicazione per il trigger

Una coda dell'applicazione è una coda locale utilizzata dalle applicazioni per la messaggistica, tramite MQI. L'attivazione richiede la definizione di un certo numero di attributi della coda sulla coda dell'applicazione.

L'attivazione è abilitata dall'attributo *Trigger* (TRIGGER nei comandi MQSC). In questo esempio, un evento trigger deve essere generato quando ci sono 100 messaggi di priorità 5 o superiore sulla coda locale MOTOR.INSURANCE.QUEUE, come segue:

```
DEFINE QLOCAL (MOTOR.INSURANCE.QUEUE) +
       PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
       MAXMSGL (2000) +
       DEFPSIST (YES) +
       INITQ (MOTOR.INS.INIT.QUEUE) +
       TRIGGER +
       TRIGTYPE (DEPTH) +
       TRIGDPTH (100)+
       TRIGMPRI (5)
```

dove:

QLOCAL (MOTOR.INSURANCE.QUEUE)
Indica il nome della coda dell'applicazione che si sta definendo.

PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
Indica il nome della definizione del processo che definisce l'applicazione che deve essere avviata da un programma di controllo trigger.

MAXMSGL (2000)
Indica la lunghezza massima dei messaggi sulla coda.

DEFPSIST (YES)
Specifica che i messaggi su questa coda sono persistenti per impostazione predefinita.

INITQ (MOTOR.INS.INIT.QUEUE)
Indica il nome della coda di iniziazione su cui il gestore code deve inserire il messaggio trigger.

TRIGGER
Indica il valore dell'attributo trigger.

TRIGTYPE (DEPTH)
Specifica che un evento trigger viene generato quando il numero di messaggi con la priorità richiesta (TRIGMPRI) raggiunge il numero specificato in TRIGDPTH.

TRIGDPTH (100)
Indica il numero di messaggi richiesti per generare un evento trigger.

TRIGMPRI (5)
Indica la priorità dei messaggi che devono essere conteggiati dal gestore code nel decidere se generare un evento trigger. Vengono conteggiati solo i messaggi con priorità 5 o superiore.

Definizione di una coda di iniziazione

Quando si verifica un evento trigger, il gestore code inserisce un messaggio trigger nella coda di avvio specificata nella definizione della coda dell'applicazione. Le code di iniziazione non dispongono di impostazioni speciali, ma è possibile utilizzare la definizione seguente della coda locale MOTOR.INS.INIT.QUEUE per istruzioni:

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
       GET (ENABLED) +
       NOSHARE +
       NOTRIGGER +
       MAXMSGL (2000) +
       MAXDEPTH (1000)
```

Definizione di un processo

Utilizzare il comando DEFINE PROCESS per creare un processo di definizione. Una definizione processo definisce l'applicazione da utilizzare per elaborare i messaggi dalla coda dell'applicazione. La definizione della coda dell'applicazione denomina il processo da utilizzare e quindi associa la coda dell'applicazione all'applicazione da utilizzare per elaborare i relativi messaggi. Questa operazione viene eseguita tramite l'attributo PROCESS sulla coda dell'applicazione MOTOR.INSURANCE.QUEUE. Il seguente comando MQSC definisce il processo richiesto, MOTOR.INSURANCE.QUOTE.PROCESS, identificato in questo esempio:

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
               DESCR ('Insurance request message processing') +
               APPLTYPE (UNIX) +
               APPLICID ('/u/admin/test/IRMP01') +
               USERDATA ('open, close, 235')
```

Dove:

MOTOR.INSURANCE.QUOTE.PROCESS
È il nome della definizione del processo.

DESCR ('Insurance request message processing')
Descrive il programma applicativo a cui si riferisce questa definizione. Questo testo viene visualizzato quando si utilizza il comando DISPLAY PROCESS. Questo può aiutare a identificare cosa fa il processo. Se si utilizzano spazi nella stringa, è necessario racchiudere la stringa tra virgolette singole.

APPLTYPE (UNIX)
Indica il tipo di applicazione da avviare.

APPLICID ('/u/admin/test/IRMP01')
È il nome del file eseguibile dell'applicazione, specificato come nome file completo. Nei sistemi Windows , un tipico valore APPLICID è `c:\appl\test\irmp01.exe`.

USERDATA ('open, close, 235')
Sono dati definiti dall'utente, che possono essere utilizzati dall'applicazione.

Visualizzazione degli attributi di una definizione di processo

Utilizzare il comando DISPLAY PROCESS per esaminare i risultati della definizione. Ad esempio:

```
DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)


    24 : DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) ALL
AMQ8407: Display Process details.
    DESCR ('Insurance request message processing')
    APPLICID ('/u/admin/test/IRMP01')
```

```
    USERDATA (open, close, 235)
    PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
    APPLTYPE (UNIX)
```

È inoltre possibile utilizzare il comando MQSC ALTER PROCESS per modificare una definizione di processo esistente e il comando DELETE PROCESS per eliminare una definizione di processo.

Gestione di oggetti IBM WebSphere MQ remoti

Questa sezione illustra come gestire gli oggetti IBM WebSphere MQ su un gestore code remoto utilizzando comandi MQSC e come utilizzare gli oggetti della coda remota per controllare la destinazione dei messaggi e dei messaggi di risposta.

Questa sezione descrive:

Canali, cluster e accodamento remoto

Un gestore code comunica con un altro gestore code inviando un messaggio e, se necessario, ricevendo una risposta. Il gestore code di ricezione potrebbe essere:

- Sulla stessa macchina
- Su un'altra macchina nella stessa posizione (o anche dall'altra parte del mondo)
- Esecuzione sulla stessa piattaforma del gestore code locale
- Esecuzione su un'altra piattaforma supportata da WebSphere MQ

Questi messaggi potrebbero provenire da:

- Programmi applicativi scritti dall'utente che trasferiscono i dati da un nodo ad un altro
- Applicazioni di gestione scritte dall'utente che utilizzano comandi PCF o MQAI
- IBM WebSphere MQ Explorer.
- Gestori code che inviano:
  - Messaggi di evento di strumentazione per un altro gestore code
  - I comandi MQSC immessi da un comando `runmqsc` in modalità indiretta (dove i comandi vengono eseguiti su un altro gestore code)

Prima che un messaggio possa essere inviato a un gestore code remoto, il gestore code locale ha bisogno di un meccanismo per rilevare l'arrivo dei messaggi e trasportarli, che consiste in:

- Almeno un canale
- Una coda di trasmissione
- Un iniziatore di canali

Perché un gestore code remoto riceva un messaggio, è richiesto un listener.

Un canale è un collegamento di comunicazione unidirezionale tra due gestori code e può trasportare messaggi destinati a qualsiasi numero di code sul gestore code remoto.

Ogni estremità del canale ha una definizione separata. Ad esempio, se un'estremità è un mittente o un server, l'altra estremità deve essere un destinatario o un richiedente. Un canale semplice è costituito da una *definizione di canale mittente* all'estremità del gestore code locale e da una *definizione di canale destinatario* all'estremità del gestore code remoto. Le due definizioni devono avere lo stesso nome e insieme costituiscono un singolo canale di messaggi.

Se si desidera che il gestore code remoto risponda ai messaggi inviati dal gestore code locale, impostare un secondo canale per inviare le risposte al gestore code locale.

Utilizzare il comando MQSC DEFINE CHANNEL per definire i canali. In questa sezione, gli esempi relativi ai canali utilizzano gli attributi di canale predefiniti, se non diversamente specificato.

Esiste un MCA (message channel agent) ad ogni estremità di un canale, che controlla l'invio e la ricezione dei messaggi. L'MCA prende i messaggi dalla coda di trasmissione e li inserisce nel collegamento di comunicazione tra i gestori code.

Una coda di trasmissione è una coda locale specializzata che conserva temporaneamente i messaggi prima che MCA li raccolga e li invii al gestore code remoto. Si specifica il nome della coda di trasmissione su una *definizione di coda remota*.

È possibile consentire a un MCA di trasferire i messaggi utilizzando più thread. Questo processo è noto come *pipelining*. Pipelining consente all'MCA di trasferire i messaggi in modo più efficiente, migliorando le prestazioni del canale. Consultare Attributi dei canali per i dettagli su come configurare un canale per utilizzare la pipeline.

"Preparazione di canali e code di trasmissione per la gestione remota" a pagina 107 indica come utilizzare queste definizioni per configurare la gestione remota.

Per ulteriori informazioni sull'impostazione dell'accodamento distribuito in generale, consultare Componente dell'accodamento distribuito.

Gestione remota mediante cluster

In una rete WebSphere MQ che utilizza l'accodamento distribuito, ogni gestore code è indipendente. Se un gestore code deve inviare messaggi a un altro gestore code, deve definire una coda di trasmissione, un canale per il gestore code remoto e una definizione di coda remota per ogni coda a cui desidera inviare i messaggi.

Un *cluster* è un gruppo di gestori code configurato in modo che i gestori code possano comunicare direttamente tra loro su una singola rete senza complesse definizioni di coda di trasmissione, canale e coda. I cluster possono essere configurati facilmente e in genere contengono gestori code che sono logicamente correlati in qualche modo e che devono condividere dati o applicazioni. Anche il cluster più piccolo riduce i costi di amministrazione del sistema.

La creazione di una rete di gestori code in un cluster implica meno definizioni rispetto alla creazione di un ambiente di accodamento distribuito tradizionale. Con un numero inferiore di definizioni da creare, è possibile impostare o modificare la rete in modo più rapido e semplice e ridurre il rischio di errori nelle definizioni.

Per impostare un cluster, è necessaria una definizione del mittente del cluster (CLUSSDR) e un destinatario del cluster (CLUSRCVR) per ogni gestore code. Non è necessaria alcuna definizione di coda di trasmissione o di coda remota. I principi dell'amministrazione remota sono gli stessi quando vengono utilizzati all'interno di un cluster, ma le definizioni stesse sono notevolmente semplificate.

Per ulteriori informazioni sui cluster, sui relativi attributi e su come impostarli, consultare Cluster di gestori code.

Gestione remota da un gestore code locale

Questa sezione spiega come gestire un gestore code remoto da un gestore code locale utilizzando comandi MQSC e PCF.

La preparazione di code e canali è essenzialmente la stessa per i comandi MQSC e PCF. In questa sezione, gli esempi mostrano i comandi MQSC, perché sono più semplici da comprendere. Per ulteriori informazioni sulla scrittura dei programmi di gestione utilizzando i comandi PCF, consultare "Utilizzo dei formati di comando programmabili" a pagina 10.

I comandi MQSC vengono inviati a un gestore code remoto in modo interattivo o da un file di testo contenente i comandi. Il gestore code remoto potrebbe trovarsi sulla stessa macchina o, più

generalmente, su una macchina diversa. È possibile gestire in remoto i gestori code in altri ambienti WebSphere MQ , inclusi i sistemi UNIX and Linux , Finestre , IBM ie z/OS.

per implementare la gestione remota, è necessario creare oggetti specifici. A meno che non si disponga di requisiti specifici, i valori predefiniti (ad esempio, per la lunghezza massima del messaggio) sono sufficienti.

Preparazione dei gestori code per la gestione remota

Come utilizzare i comandi MQSC per preparare i gestori code per la gestione remota.

Figura 17 a pagina 107 mostra la configurazione dei gestori code e dei canali necessari per l'amministrazione remota utilizzando il comando **runmqsc** . L'oggetto `source.queue.manager` è il gestore code di origine da cui è possibile immettere comandi MQSC e a cui vengono restituiti i risultati di questi comandi (messaggi operatore). L'oggetto `target.queue.manager` è il nome del gestore code di destinazione, che elabora i comandi e genera eventuali messaggi dell'operatore.

Nota: Se si utilizza **runmqsc** con l'opzione -w, `source.queue.manager` ***deve*** essere il gestore code predefinito. Per ulteriori informazioni sulla creazione di un gestore code, consultare crtmqm.

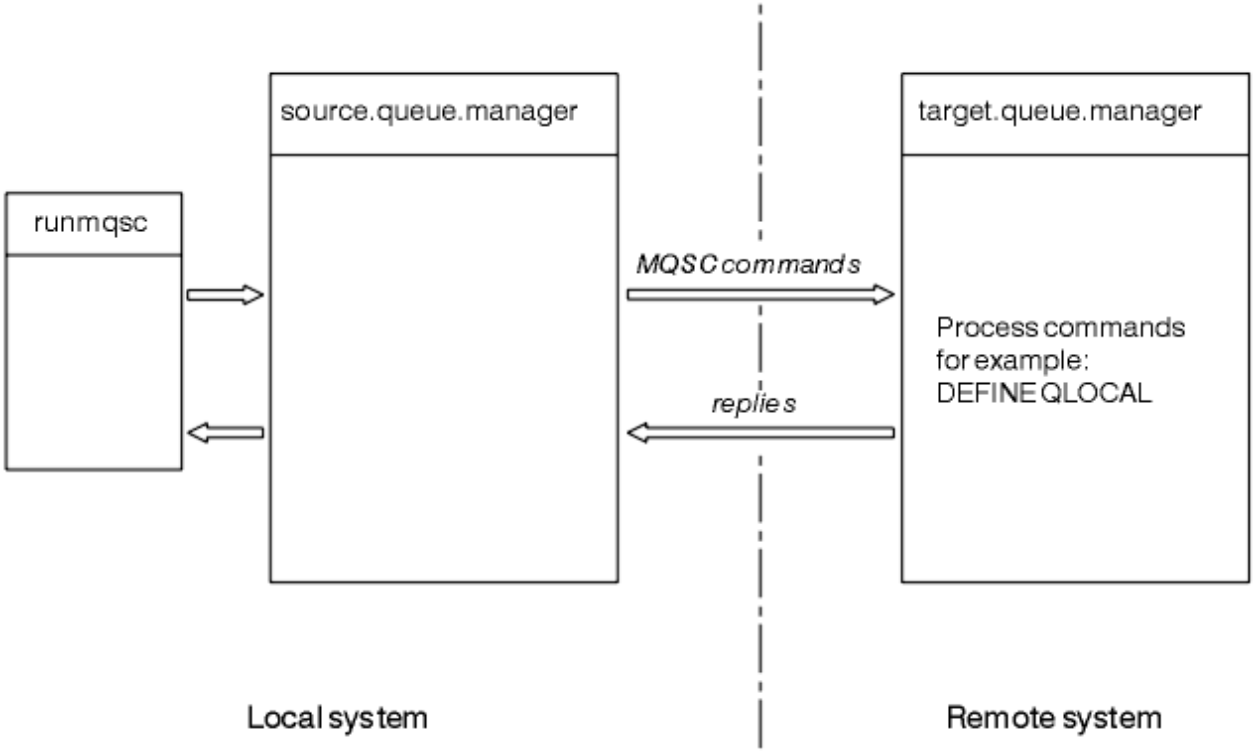


Figura 17. Gestione remota mediante i comandi MQSC

Su entrambi i sistemi, se non è già stato fatto:

- Creare il gestore code e gli oggetti predefiniti utilizzando il comando `crtmqm` .
- Avviare il gestore code utilizzando il comando `strmqm` .

Sul gestore code di destinazione:

- La coda comandi, SYSTEM.ADMIN.COMMAND.QUEUE, deve essere presente. Questa coda viene creata per impostazione predefinita quando viene creato il gestore code.

È necessario eseguire questi comandi localmente o su una funzione di rete come Telnet.

Preparazione di canali e code di trasmissione per la gestione remota

Come utilizzare i comandi MQSC per preparare i canali e le code di trasmissione per la gestione remota.

Per eseguire i comandi MQSC in remoto, impostare due canali, uno per ogni direzione e le relative code di trasmissione associate. Questo esempio presuppone che si stia utilizzando TCP/IP come tipo di trasporto e che si conosca l'indirizzo TCP/IP interessato.

Il canale `source.to.target` consente di inviare i comandi MQSC dal gestore code di origine al gestore code di destinazione. Il mittente è `source.queue.manager` e il destinatario è `target.queue.manager`. Il canale `target.to.source` viene utilizzato per restituire l'emissione dai comandi e da tutti i messaggi dell'operatore generati al gestore code di origine. È necessario definire anche una coda di trasmissione per ogni canale. Questa coda è una coda locale a cui viene assegnato il nome del gestore code di ricezione. Il nome XMITQ deve corrispondere al nome del gestore code remoto affinché la gestione remota funzioni, a meno che non si stia utilizzando un alias del gestore code. Figura 18 a pagina 108 riepiloga questa configurazione.

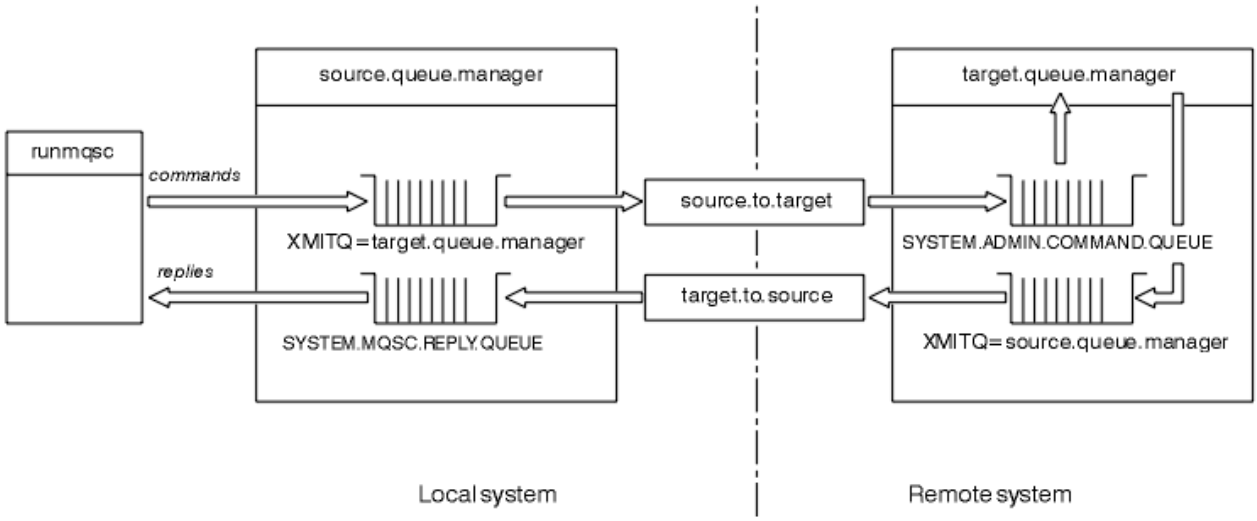


Figura 18. Impostazione di canali e code per l'amministrazione remota

Per ulteriori informazioni sull'impostazione dei canali, consultare Connessione delle applicazioni utilizzando l'accodamento distribuito .

Definizione di canali, listener e code di trasmissione

Sul gestore code di origine (`source.queue.manager`), immettere i seguenti comandi MQSC per definire i canali, il listener e la coda di trasmissione:

1. Definire il canale mittente nel gestore code di origine:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(SDR) +
       CONNAME (RHX5498) +
       XMITQ ('target.queue.manager') +
       TRPTYPE(TCP)
```

2. Definire il canale ricevente sul gestore code di origine:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Definire il listener sul gestore code di origine:

```
DEFINE LISTENER ('source.queue.manager') +
       TRPTYPE (TCP)
```

4. Definire la coda di trasmissione sul gestore code di origine:

```
DEFINE QLOCAL ('target.queue.manager') +
       USAGE (XMITQ)
```

Immettere i seguenti comandi sul gestore code di destinazione (`target.queue.manager`), per creare i canali, il listener e la coda di trasmissione:

1. Definire il canale mittente sul gestore code di destinazione:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(SDR) +
       CONNAME (RHX7721) +
       XMITQ ('source.queue.manager') +
       TRPTYPE(TCP)
```

2. Definire il canale ricevente sul gestore code di destinazione:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Definire il listener sul gestore code di destinazione:

```
DEFINE LISTENER ('target.queue.manager') +
       TRPTYPE (TCP)
```

4. Definire la coda di trasmissione sul gestore code di destinazione:

```
DEFINE QLOCAL ('source.queue.manager') +
       USAGE (XMITQ)
```

Nota: I nomi di connessione TCP/IP specificati per l'attributo CONNAME nelle definizioni del canale mittente sono solo a scopo illustrativo. Questo è il nome della rete della macchina all'estremità *altra* della connessione. Utilizzare i valori appropriati per la rete.

Avvio dei listener e canali

Come utilizzare i comandi MQSC per avviare listener e canali.

Avviare entrambi i listener utilizzando i seguenti comandi MQSC:

1. Avviare il listener sul gestore code di origine, `source.queue.manager`, immettendo il comando MQSC riportato di seguito:

```
START LISTENER ('source.queue.manager')
```

2. Avviare il listener sul gestore code di destinazione, `target.queue.manager`, emettendo il seguente comando MQSC:

```
START LISTENER ('target.queue.manager')
```

Avviare entrambi i canali del mittente utilizzando i seguenti comandi MQSC:

1. Avviare il canale mittente sul gestore code di origine, `source.queue.manager`, immettendo il seguente comando MQSC:

```
START CHANNEL ('source.to.target')
```

2. Avviare il canale mittente sul gestore code di destinazione, `target.queue.manager`, emettendo il seguente comando MQSC:

```
START CHANNEL ('target.to.source')
```

Definizione automatica dei canali
È possibile abilitare la definizione automatica delle definizioni di connessione server e ricevente aggiornando l'oggetto gestore code utilizzando il comando MQSC, ALTER QMGR (o il comando PCF Change Queue Manager).

Se WebSphere MQ riceve una richiesta di collegamento in entrata e non riesce a trovare un canale ricevente o di connessione server appropriato, crea automaticamente un canale. Le definizioni automatiche si basano su due definizioni predefinite fornite con WebSphere MQ: SYSTEM.AUTO.RECEIVER e SYSTEM.AUTO.SVRCONN.

Per ulteriori informazioni sulla creazione automatica delle definizioni di canale, consultare Preparazione dei canali. Per informazioni relative alla definizione automatica dei canali per i cluster, consultare Definizione automatica dei canali cluster.

Gestione del server dei comandi per la gestione remota

Come avviare, arrestare e visualizzare lo stato del server dei comandi. Un server dei comandi è obbligatorio per tutte le operazioni di gestione che utilizzano i comandi PCF, MQAI e anche per la gestione remota.

Ciascun gestore code può avere un server dei comandi associato. Un server dei comandi elabora i comandi in entrata dai gestori code remoti o i comandi PCF dalle applicazioni. Presenta i comandi al gestore code per l'elaborazione e restituisce un codice di completamento o un messaggio dell'operatore in base all'origine del comando.

Nota: Per la gestione remota, assicurarsi che il gestore code di destinazione sia in esecuzione. Altrimenti, i messaggi che contengono i comandi non possono lasciare il gestore code da cui vengono emessi. Invece, questi messaggi vengono accodati nella coda di trasmissione locale che serve il gestore code remoto. Evita questa situazione.

Esistono comandi di controllo separati per avviare e arrestare il server dei comandi. Se il server dei comandi è in esecuzione, gli utenti di WebSphere MQ per Windows o WebSphere MQ per Linux (piattaformex86 e x86-64) possono eseguire le operazioni descritte nelle seguenti sezioni utilizzando IBM WebSphere MQ Explorer. Per ulteriori informazioni, vedere “Amministrazione mediante IBM WebSphere MQ Explorer” a pagina 57.

Avvio del server dei comandi

A seconda del valore dell'attributo del gestore code, *SCMDSERV*, il server dei comandi viene avviato automaticamente all'avvio del gestore code oppure deve essere avviato manualmente. Il valore dell'attributo del gestore code può essere modificato utilizzando il comando MQSC `ALTER QMGR` specificando il parametro SCMDSERV. Per impostazione predefinita, il server dei comandi viene avviato automaticamente.

Se *SCMDSERV* è impostato su MANUAL, avviare il server dei comandi utilizzando il comando:

```
strmqcsv saturn.queue.manager
```

dove `saturn.queue.manager` è il gestore code per cui viene avviato il server dei comandi.

Visualizzazione dello stato del server dei comandi

Per la gestione remota, verificare che sia in esecuzione il server dei comandi sul gestore code di destinazione. Se non è in esecuzione, i comandi remoti non possono essere elaborati. Tutti i messaggi che contengono comandi vengono accodati nella coda comandi del gestore code di destinazione.

Per visualizzare lo stato del server dei comandi per un gestore code, immettere il seguente comando MQSC:

```
DISPLAY QMSTATUS CMDSERV
```

Arresto di un server dei comandi

Per terminare il server dei comandi avviato dall'esempio precedente, utilizzare il seguente comando:

```
endmqcsv saturn.queue.manager
```

È possibile arrestare il server dei comandi in due modi:

- Per un arresto controllato, utilizzare il comando `endmqcsv` con l'indicatore -c, che è il valore predefinito.
- Per un arresto immediato, utilizzare il comando `endmqcsv` con l'indicatore -i.

Nota: L'arresto di un gestore code termina anche il server dei comandi ad esso associato.

Immissione di comandi MQSC su un gestore code remoto

È possibile utilizzare un formato particolare del comando **runmqsc** per eseguire i comandi MQSC su un gestore code remoto.

Il server dei comandi ***deve*** essere in esecuzione sul gestore code di destinazione, se sta per elaborare i comandi MQSC in remoto. (Questo non è necessario sul gestore code di origine). Per informazioni su come avviare il server dei comandi su un gestore code, consultare “Gestione del server dei comandi per la gestione remota” a pagina 110.

Sul gestore code di origine, è possibile eseguire i comandi MQSC in modo interattivo in modalità indiretta immettendo:

```
runmqsc -w 30 target.queue.manager
```

Questo formato del comando `runmqsc` , con l'indicatore -w, esegue i comandi MQSC in modalità indiretta, in cui i comandi vengono inseriti (in forma modificata) nella coda di input del server dei comandi ed eseguiti in ordine.

Quando si immette un comando MQSC, viene reindirizzato al gestore code remoto, in questo caso, `target.queue.manager`. Il timeout è impostato su 30 secondi; se una risposta non viene ricevuta entro 30 secondi, viene generato il seguente messaggio sul gestore code locale (di origine):

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

Quando si smette di emettere comandi MQSC, il gestore code locale visualizza tutte le risposte in timeout che sono arrivate e scarta tutte le altre risposte.

Il gestore code di origine assume il valore predefinito del gestore code locale predefinito. Se si specifica l'opzione `-m` *LocalQmgrName* nel comando **runmqsc** , è possibile indirizzare i comandi da emettere tramite qualsiasi gestore code locale.

In modalità indiretta, è anche possibile eseguire un file di comandi MQSC su un gestore code remoto. Ad esempio:

```
runmqsc -w 60 target.queue.manager < mycomds.in > report.out
```

dove `mycomds.in` è un file contenente i comandi MQSC e `report.out` è il file di report.

Metodo consigliato per l'emissione di comandi in remoto

Quando si immettono comandi su un gestore code remoto, considerare l'utilizzo del seguente metodo:

1. Inserire i comandi MQSC da eseguire sul sistema remoto in un file di comandi.
2. Verificare i comandi MQSC localmente, specificando l'indicatore -v nel comando `runmqsc` .

   Non è possibile utilizzare `runmqsc` per verificare i comandi MQSC su un altro gestore code.
3. Verificare che il file di comandi venga eseguito localmente senza errori.
4. Eseguire il file di comandi sul sistema remoto.

Se si verificano problemi utilizzando i comandi MQSC in remoto

Se hai difficoltà ad eseguire i comandi MQSC in remoto, assicurati di avere:

- Il server dei comandi è stato avviato sul gestore code di destinazione.
- È stata definita una coda di trasmissione valida.
- Sono state definite le due estremità dei canali del messaggio per entrambi:
  - Il canale lungo il quale vengono inviati i comandi.
  - Il canale lungo il quale devono essere restituite le risposte.
- Specificare il nome di connessione corretto (CONNAME) nella definizione del canale.
- I listener sono stati avviati prima dei canali dei messaggi.
- Verificare che l'intervallo di disconnessione non sia scaduto, ad esempio, se un canale è stato avviato ma è stato chiuso dopo un certo periodo di tempo. Questo è particolarmente importante se si avviano i canali manualmente.
- Ha inviato richieste da un gestore code di origine che non hanno senso al gestore code di destinazione (ad esempio, richieste che includono parametri non supportati sul gestore code remoto).

Vedi anche “Risoluzione dei problemi con i comandi MQSC” a pagina 80.

Creazione di una definizione locale di una coda remota

Una definizione locale di una coda remota è una definizione su un gestore code locale che fa riferimento a una coda su un gestore code remoto.

Non è necessario definire una coda remota da una posizione locale, ma il vantaggio è che le applicazioni possono fare riferimento alla coda remota tramite il nome definito localmente invece di dover specificare un nome qualificato dall'ID del gestore code su cui si trova la coda remota.

Informazioni sul funzionamento delle definizioni locali delle code remote

Un'applicazione si connette a un gestore code locale ed emette una chiamata `MQOPEN` . Nella chiamata di apertura, il nome coda specificato è quello di una definizione di coda remota sul gestore code locale. La definizione della coda remota fornisce i nomi della coda di destinazione, del gestore code di destinazione e, facoltativamente, di una coda di trasmissione. Per inserire un messaggio sulla coda remota, l'applicazione emette una chiamata `MQPUT` , specificando l'handle restituito dal richiamo `MQOPEN` . Il gestore code utilizza il nome coda remota e il nome gestore code remoto in un'intestazione di trasmissione all'inizio del messaggio. Queste informazioni vengono utilizzate per instradare il messaggio alla destinazione corretta nella rete.

Come amministratore, è possibile controllare la destinazione del messaggio modificando la definizione della coda remota.

Il seguente esempio mostra come un'applicazione inserisce un messaggio in una coda di proprietà di un gestore code remoto. L'applicazione si connette a un gestore code, ad esempio `saturn.queue.manager`. La coda di destinazione è di proprietà di un altro gestore code.

Nella chiamata `MQOPEN` , l'applicazione specifica questi campi:

Valore del campo	Descrizione
ObjectName CYAN.REMOTE.QUEUE	Specifica il nome locale dell'oggetto coda remota. Definisce la coda di destinazione e il gestore code di destinazione.
ObjectType (coda)	Identifica questo oggetto come coda.
ObjectQmgrName Vuoto o `saturn.queue.manager`	Questo campo è facoltativo. Se vuoto, viene utilizzato il nome del gestore code locale. (Questo è il gestore code su cui esiste la definizione della coda remota).

Successivamente, l'applicazione emette una chiamata MQPUT per inserire un messaggio in questa coda.

Sul gestore code locale, è possibile creare una definizione locale di una coda remota utilizzando i seguenti comandi MQSC:

```
DEFINE QREMOTE (CYAN.REMOTE.QUEUE) +
       DESCR ('Queue for auto insurance requests from the branches') +
       RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE) +
       RQMNAME (jupiter.queue.manager) +
       XMITQ (INQUOTE.XMIT.QUEUE)
```

dove:

QREMOTE (CYAN.REMOTE.QUEUE)
Specifica il nome locale dell'oggetto coda remota. Questo è il nome che le applicazioni connesse a questo gestore code devono specificare nella chiamata MQOPEN per aprire la coda AUTOMOBILE.INSURANCE.QUOTE.QUEUE sul gestore code remoto `jupiter.queue.manager`.

DESCR ('Queue for auto insurance requests from the branches')
Fornisce ulteriore testo che descrive l'utilizzo della coda.

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)
Specifica il nome della coda di destinazione sul gestore code remoto. È la coda di destinazione reale per i messaggi inviati dalle applicazioni che specificano il nome della coda CYAN.REMOTE.QUEUE. La coda AUTOMOBILE.INSURANCE.QUOTE.QUEUE deve essere definito come coda locale sul Gestore code remoto.

RQMNAME (jupiter.queue.manager)
Specifica il nome del gestore code remoto che possiede la coda di destinazione AUTOMOBILE.INSURANCE.QUOTE.QUEUE.

XMITQ (INQUOTE.XMIT.QUEUE)
Specifica il nome della coda di trasmissione. Questa opzione è facoltativa; se il nome di una coda di trasmissione non viene specificato, viene utilizzata una coda con lo stesso nome del gestore code remoto.

In entrambi i casi, la coda di trasmissione appropriata deve essere definita come una coda locale con un attributo *Usage* che specifica che è una coda di trasmissione (USAGE (XMITQ) nei comandi MQSC).

Un modo alternativo di inserire messaggi in una coda remota

L'uso di una definizione locale di una coda remota non è l'unico modo per inserire messaggi in una coda remota. Le applicazioni possono specificare il nome completo della coda, incluso il nome del gestore code remoto, come parte della chiamata MQOPEN . In questo caso, non è necessaria una definizione locale di una coda remota. Tuttavia, ciò significa che le applicazioni devono conoscere o avere accesso al nome del gestore code remoto in fase di runtime.

Utilizzo di altri comandi con code remote

È possibile utilizzare comandi MQSC per visualizzare o modificare gli attributi di un oggetto coda remota oppure è possibile eliminare l'oggetto coda remota. Ad esempio:

- Per visualizzare gli attributi della coda remota:

```
DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
```

- Per modificare la coda remota per abilitare gli inserimenti. Ciò non influisce sulla coda di destinazione, ma solo sulle applicazioni che specificano questa coda remota:

```
ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)
```

- Per eliminare questa coda remota. Ciò non influisce sulla coda di destinazione, ma solo sulla sua definizione locale:

```
  DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

Nota: Quando si elimina una coda remota, si elimina solo la rappresentazione locale della coda remota. Non si elimina la coda remota stessa o alcun messaggio su di essa.

Definizione di una coda di trasmissione

Una coda di trasmissione è una coda locale utilizzata quando un gestore code inoltra i messaggi a un gestore code remoto tramite un canale di messaggi.

Il canale fornisce un link unidirezionale al gestore code remoto. I messaggi vengono accodati nella coda di trasmissione fino a quando il canale non li accetta. Quando si definisce un canale, è necessario specificare un nome coda di trasmissione all'estremità di invio del canale di messaggi.

L'attributo del comando MQSC USAGE definisce se una coda è una coda di trasmissione o una coda normale.

Code di trasmissione predefinite

Quando un gestore code invia messaggi a un gestore code remoto, identifica la coda di trasmissione utilizzando la seguente sequenza:

1. La coda di trasmissione denominata nell'attributo XMITQ della definizione locale di una coda remota.
2. Una coda di trasmissione con lo stesso nome del gestore code di destinazione. (Questo è il valore predefinito su XMITQ della definizione locale di una coda remota.)
3. La coda di trasmissione denominata nell'attributo DEFXMITQ del gestore code locale.

Ad esempio, il seguente comando MQSC crea una coda di trasmissione predefinita su `source.queue.manager` per i messaggi che vanno a `target.queue.manager`:

```
DEFINE QLOCAL ('target.queue.manager') +
       DESCR ('Default transmission queue for target qm') +
       USAGE (XMITQ)
```

Le applicazioni possono inserire i messaggi direttamente su una coda di trasmissione o indirettamente tramite una definizione di coda remota. Vedi anche “Creazione di una definizione locale di una coda remota” a pagina 112.

Utilizzo delle definizioni di coda remota come alias

Oltre a posizionare una coda su un altro gestore code, è anche possibile utilizzare una definizione locale di una coda remota per gli alias del gestore code e gli alias della coda di risposta. Entrambi i tipi di

alias vengono risolti tramite la definizione locale di una coda remota. È necessario impostare i canali appropriati affinché il messaggio arrivi a destinazione.

Alias del gestore code

Un alias è il processo con cui il nome del gestore code di destinazione, come specificato in un messaggio, viene modificato da un gestore code sull'instradamento del messaggio.Gli alias dei gestori code sono importanti perché è possibile utilizzarli per controllare la destinazione dei messaggi all'interno di una rete di gestori code.

A tale scopo, modificare la definizione della coda remota sul gestore code nel punto di controllo. L'applicazione mittente non è a conoscenza del fatto che il nome del gestore code specificato è un alias.

Per ulteriori informazioni sugli alias del gestore code, consultare Cosa sono gli alias?.

Alias coda di risposta

Facoltativamente, un'applicazione può specificare il nome di una coda di risposta quando inserisce un *messaggio di richiesta* in una coda.

Se l'applicazione che elabora il messaggio estrae il nome della coda di risposta, sa dove inviare il *messaggio di risposta*, se necessario.

Un alias della coda di risposta è il processo mediante il quale una coda di risposta, come specificato in un messaggio di richiesta, viene modificata da un gestore code sull'instradamento del messaggio. L'applicazione mittente non riconosce che il nome della coda di risposta specificato è un alias.

Un alias della coda di risposta consente di modificare il nome della coda di risposta e, facoltativamente, il gestore code. Questo a sua volta consente di controllare quale instradamento viene utilizzato per i messaggi di risposta.

Per ulteriori informazioni sui messaggi di richiesta, sui messaggi di risposta e sulle code di risposta, consultare Tipi di messaggio e Coda di risposta e gestore code.

Per ulteriori informazioni sugli alias della coda reply - to, consultare Cluster e alias della coda reply - to .

Conversione dati tra serie di caratteri codificati

I dati dei messaggi in WebSphere MQ formati definiti (noti anche come *formati integrati*) possono essere convertiti dal gestore code da una serie di caratteri codificati a un altro, a condizione che entrambe le serie di caratteri si riferiscano a una singola lingua o a un gruppo di lingue simili.

Ad esempio, è supportata la conversione tra serie di caratteri codificati con identificativi (CCSID) 850 e 500, poiché entrambe si applicano alle lingue dell'Europa occidentale.

Per le conversioni di caratteri NL (newline) EBCDIC in ASCII, consultare Tutti i gestori code .

Le conversioni supportate sono definite in Conversione dati.

Quando un gestore code non può convertire i messaggi in formati integrati

Il gestore code non può convertire automaticamente i messaggi in formati integrati se i relativi CCSID rappresentano gruppi di lingue nazionali differenti. Ad esempio, la conversione tra CCSID 850 e CCSID 1025 (che è una serie di caratteri codificati EBCDIC per le lingue che utilizzano lo script cirillico) non è supportata perché molti dei caratteri in una serie di caratteri codificati non possono essere rappresentati nell'altra. Se si dispone di una rete di gestori code che lavorano in lingue nazionali differenti e la conversione dei dati tra alcune serie di caratteri codificati non è supportata, è possibile abilitare una conversione predefinita. La conversione dati predefinita è descritta in “Conversione dati predefinita” a pagina 116.

File ccsid.tbl

Il file ccsid.tbl viene utilizzato per i seguenti scopi:

- In WebSphere MQ per Windows registra tutte le serie di codici supportate.
- Sulle piattaforme AIX e HP-UX , le serie di codici supportate sono gestite internamente dal sistema operativo.
- Per tutte le altre piattaforme UNIX and Linux , i set di codici supportati vengono conservati nelle tabelle di conversione fornite da WebSphere MQ.
- Specifica eventuali codeset aggiuntivi. Per specificare ulteriori codeset, è necessario modificare ccsid.tbl (nel file viene fornita una guida su come eseguire questa operazione).
- Specifica qualsiasi conversione dati predefinita.

È possibile aggiornare le informazioni registrate in ccsid.tbl; è possibile eseguire questa operazione se, ad esempio, una futura release del proprio sistema operativo supporta ulteriori serie di caratteri codificati.

In WebSphere MQ per Windows, ccsid.tbl si trova nella directory `C:\Program Files\IBM\WebSphere MQ\conv\table` per impostazione predefinita.

In WebSphere MQ per sistemi UNIX and Linux , ccsid.tbl si trova nella directory `/var/mqm/conv/table`.

Conversione dati predefinita

Se si impostano i canali tra due macchine su cui la conversione dati non è normalmente supportata, è necessario abilitare la conversione dati predefinita per il funzionamento dei canali.

Per abilitare la conversione dati predefinita, modificare il file ccsid.tbl per specificare un CCSID EBCDIC predefinito e un CCSID ASCII predefinito. Le istruzioni su come eseguire questa operazione sono incluse nel file. È necessario effettuare questa operazione su tutte le macchine che verranno connesse utilizzando i canali. Riavviare il gestore code per rendere effettive le modifiche.

Il processo di conversione dati predefinito è il seguente:

- Se la conversione tra CCSID di origine e di destinazione non è supportata, ma i CCSID degli ambienti di origine e di destinazione sono entrambi EBCDIC o ASCII, i dati carattere vengono passati all'applicazione di destinazione senza conversione.
- Se un CCSID rappresenta una serie di caratteri codificati ASCII, e l'altro rappresenta una serie di caratteri codificati EBCDIC, WebSphere MQ converte i dati utilizzando i CCSID di conversione dati predefiniti definiti in ccsid.tbl.

Nota: Tentare di limitare i caratteri che vengono convertiti a quelli che hanno gli stessi valori di codice nella serie di caratteri codificati specificata per il messaggio e nella serie di caratteri codificati predefinita. Se si utilizza solo la serie di caratteri validi per i nomi oggetto WebSphere MQ (come definito in Denominazione di oggetti IBM WebSphere MQ), si soddisferà questo requisito. Le eccezioni si verificano con CCSID EBCDIC 290, 930, 1279 e 5026 utilizzati in Giappone, dove i caratteri minuscoli hanno codici diversi da quelli utilizzati in altri CCSID EBCDIC.

Conversione dei messaggi in formati definiti dall'utente

Il gestore code non è in grado di convertire i messaggi in formati definiti dall'utente da una serie di caratteri codificati ad un'altra. Se è necessario convertire i dati in un formato definito dall'utente, è necessario fornire un'uscita di conversione dati per ciascun formato. Non utilizzare CCSID predefiniti per convertire i dati carattere in formati definiti dall'utente. Per ulteriori informazioni sulla conversione dei dati in formati definiti dall'utente e sulla scrittura delle uscite di conversione dati, consultare Scrittura delle uscite di conversione dati.

Modifica del CCSID del gestore code

Una volta utilizzato il CCSID del comando ALTER QMGR per modificare il CCSID del gestore code, arrestare e riavviare il gestore code per garantire che tutte le applicazioni in esecuzione, inclusi il server dei comandi e i programmi del canale, vengano arrestate e riavviate.

Ciò è necessario poiché tutte le applicazioni in esecuzione quando il CCSID del gestore code viene modificato continuano ad utilizzare il CCSID esistente.

AmministrazioneIBM WebSphere MQ Telemetry

IBM WebSphere MQ Telemetry viene gestito utilizzando IBM WebSphere MQ Explorer o su una riga comandi. Utilizzare Explorer per configurare i canali di telemetria, controllare il servizio di telemetria e monitorare i client MQTT connessi a IBM WebSphere MQ. Configurare la sicurezza di IBM WebSphere MQ Telemetry utilizzando JAAS, SSL e l'object authority manager IBM WebSphere MQ .

Amministrazione mediante IBM WebSphere MQ Explorer

Utilizzare Explorer per configurare i canali di telemetria, controllare il servizio di telemetria e monitorare i client MQTT connessi a IBM WebSphere MQ. Configurare la sicurezza di IBM WebSphere MQ Telemetry utilizzando JAAS, SSL e l'object authority manager IBM WebSphere MQ .

Gestione mediante la riga comandi

IBM WebSphere MQ Telemetry può essere completamente gestito dalla riga comandi utilizzando i comandi IBM WebSphere MQ MQSC .

La documentazione IBM WebSphere MQ Telemetry dispone inoltre di script di esempio che dimostrano l'utilizzo di base dell'applicazione client MQ Telemetry Transport v3 .

Leggere e comprendere gli esempi in IBM WebSphere MQ Telemetry programmi di esempio nella sezione Sviluppo di applicazioni per IBM WebSphere MQ Telemetry prima di utilizzarli.

Concetti correlati

WebSphere MQ Telemetry

“Configurare l'accodamento distribuito per inviare messaggi ai client MQTT” a pagina 121
Le applicazioni IBM WebSphere MQ possono inviare messaggi dei client MQTT v3 pubblicando su una sottoscrizione creata da un client o inviando direttamente un messaggio. Indipendentemente dal metodo utilizzato, il messaggio viene inserito in `SYSTEM.MQTT.TRANSMIT.QUEUE`e inviato al client dal servizio di telemetria (MQXR). Esistono diversi modi per inserire un messaggio su `SYSTEM.MQTT.TRANSMIT.QUEUE`.

“Identificazione, autorizzazione e autenticazione del client MQTT” a pagina 124

“Autenticazione del canale di telemetria mediante SSL” a pagina 130

“Privacy di pubblicazione sui canali di telemetria” a pagina 132

“Configurazione SSL di client e canali di telemetria MQTT” a pagina 133

“Configurazione del canale di telemetria JAAS” a pagina 138
Configurare JAAS per autenticare il `Nome utente` inviato dal client.

“IBM WebSphere MQ Daemon di telemetria per i concetti delle periferiche” a pagina 140
Il daemon IBM WebSphere MQ Telemetry per dispositivi è un'applicazione client MQTT V3 avanzata. Utilizzarlo per memorizzare e inoltrare messaggi da altri client MQTT. Si connette a IBM WebSphere MQ come un client MQTT, ma è anche possibile connettervi altri client MQTT.

Attività correlate

“Configurazione di un gestore code per la telemetria in Linux e AIX” a pagina 118
Seguire questi passi manuali per configurare un gestore code per l'esecuzione di IBM WebSphere MQ Telemetry. È possibile eseguire una procedura automatizzata per impostare una configurazione più semplice utilizzando il supporto IBM WebSphere MQ Telemetry per IBM WebSphere MQ Explorer.

“Configurazione di un gestore code per la telemetria su Windows” a pagina 119
Seguire questi passi manuali per configurare un gestore code per l'esecuzione di IBM WebSphere MQ Telemetry. È possibile eseguire una procedura automatizzata per impostare una configurazione più semplice utilizzando il supporto IBM WebSphere MQ Telemetry per IBM WebSphere MQ Explorer.

Riferimenti correlati

Proprietà MQXR

Configurazione di un gestore code per la telemetria in Linux e AIX

Seguire questi passi manuali per configurare un gestore code per l'esecuzione di IBM WebSphere MQ Telemetry. È possibile eseguire una procedura automatizzata per impostare una configurazione più semplice utilizzando il supporto IBM WebSphere MQ Telemetry per IBM WebSphere MQ Explorer.

Prima di iniziare

1. Consultare Installazione IBM WebSphere MQ Telemetry per informazioni su come installare IBM WebSphere MQe la funzione IBM WebSphere MQ Telemetry .
2. Creare e avviare un gestore code. Il gestore code viene indicato come *qMgr* in questa attività.
3. Come parte di questa attività, configurare il servizio di telemetria (MQXR). Le impostazioni della proprietà MQXR sono memorizzate in un file delle proprietà specifico della piattaforma: `mqxr_unix.properties`. Normalmente non è necessario modificare il file delle proprietà MQXR direttamente, poiché quasi tutte le impostazioni possono essere configurate tramite i comandi di amministrazione MQSC o MQ Explorer. Se si decide di modificare direttamente il file, arrestare il gestore code prima di apportare le modifiche. Vedere Proprietà MQXR.

Informazioni su questa attività

Il supporto IBM WebSphere MQ Telemetry per IBM WebSphere MQ Explorer include una procedura guidata e una procedura di comando di esempio `sampleMQM`. Impostano una configurazione iniziale utilizzando l'ID utente guest; consultare Verifica dell'installazione di IBM WebSphere MQ Telemetry utilizzando i programmi di esempio IBM WebSphere MQ Explorer e IBM WebSphere MQ Telemetry.

Seguire i passi in questa attività per configurare manualmente IBM WebSphere MQ Telemetry utilizzando diversi schemi di autorizzazione.

Procedura

1. Aprire una finestra comandi nella directory degli esempi di telemetria.

   La directory degli esempi di telemetria è `/opt/mqm/mqxr/samples`.
2. Creare la coda di trasmissione di telemetria.

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc
   qMgr
   ```

   Quando il servizio di telemetria (MQXR) viene avviato per la prima volta, crea `SYSTEM.MQTT.TRANSMIT.QUEUE`.

   Viene creato manualmente in questa attività, perché `SYSTEM.MQTT.TRANSMIT.QUEUE` deve esistere prima che venga avviato il servizio di telemetria (MQXR), per autorizzare l'accesso ad esso.
3. Imposta la coda di trasmissione predefinita

   Quando il servizio MQXR (telemetria) viene avviato per la prima volta, non modifica il gestore code per rendere `SYSTEM.MQTT.TRANSMIT.QUEUE` la coda di trasmissione predefinita.

   Per rendere `SYSTEM.MQTT.TRANSMIT.QUEUE` la coda di trasmissione predefinita, modificare la proprietà della coda di trasmissione predefinita. Modificare la proprietà utilizzando IBM WebSphere MQ Explorer o con il comando nel seguente esempio:

   ```
   echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
   ```

   La modifica della coda di trasmissione predefinita potrebbe interferire con la configurazione esistente. Il motivo per cui si modifica la coda di trasmissione predefinita in `SYSTEM.MQTT.TRANSMIT.QUEUE` è rendere più semplice l'invio di messaggi direttamente ai client MQTT. Senza modificare la coda di trasmissione predefinita, è necessario aggiungere una definizione di coda remota per ogni client che riceve messaggi IBM WebSphere MQ ; consultare “Invio diretto di un messaggio ad un cliente” a pagina 123.

4. Seguire una procedura in “Autorizzazione dei client MQTT ad accedere agli oggetti WebSphere MQ” a pagina 125 per creare uno o più ID utente. Gli ID utente hanno l'autorità di pubblicare, sottoscrivere e inviare pubblicazioni ai client MQTT.
5. Installa il servizio di telemetria (MQXR)

   ```
   cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
   ```

   Vedere anche il codice di esempio in Figura 19 a pagina 119.
6. Avvia servizio

   ```
   echo "START SERVICE(SYSTEM.MQXR.SERVICE)" | runmqsc qMgr
   ```

   Il servizio di telemetria (MQXR) viene avviato automaticamente all'avvio del gestore code.

   Viene avviato manualmente in questa attività perché il gestore code è già in esecuzione.
7. Utilizzando IBM WebSphere MQ Explorer, configurare i canali di telemetria per accettare le connessioni dai client MQTT.

   I canali di telemetria devono essere configurati in modo che le relative identità siano uno degli ID utente definiti al passo 4.

   Vedere anche DEFINE CHANNEL (MQTT).
8. Verificare la configurazione eseguendo il client di esempio.

   Per consentire al client di esempio di utilizzare il proprio canale di telemetria, il canale deve autorizzare il client a pubblicare, sottoscrivere e ricevere pubblicazioni. Il client di esempio si collega al canale di telemetria sulla porta 1883 per impostazione predefinita. Vedere inoltre IBM WebSphere MQ Telemetry programmi di esempio.

Esempio

Figura 19 a pagina 119 mostra il comando **runmqsc** per creare il SYSTEM.MQXR.SERVICE manualmente su Linux.

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
    CONTROL(QMGR) +
    DESCR('Manages clients using MQXR protocols such as MQTT') +
    SERVTYPE(SERVER) +
    STARTCMD('+MQ_INSTALL_PATH+/mqxr/bin/runMQXRService.sh') +
    STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+" -g "+MQ_DATA_PATH+"') +
    STOPCMD('+MQ_INSTALL_PATH+/mqxr/bin/endMQXRService.sh') +
    STOPARG('-m +QMNAME+') +
    STDOUT('+MQ_Q_MGR_DATA_PATH+/mqxr.stdout') +
    STDERR('+MQ_Q_MGR_DATA_PATH+/mqxr.stderr')
```

Figura 19. installMQXRService_unix.mqsc

Configurazione di un gestore code per la telemetria su Windows

Seguire questi passi manuali per configurare un gestore code per l'esecuzione di IBM WebSphere MQ Telemetry. È possibile eseguire una procedura automatizzata per impostare una configurazione più semplice utilizzando il supporto IBM WebSphere MQ Telemetry per IBM WebSphere MQ Explorer.

Prima di iniziare

1. Consultare Installazione IBM WebSphere MQ Telemetry per informazioni su come installare IBM WebSphere MQe la funzione IBM WebSphere MQ Telemetry .
2. Creare e avviare un gestore code. Il gestore code viene indicato come *qMgr* in questa attività.
3. Come parte di questa attività, configurare il servizio di telemetria (MQXR). Le impostazioni della proprietà MQXR sono memorizzate in un file delle proprietà specifico della piattaforma: mqxr_win.properties. Normalmente non è necessario modificare il file delle proprietà MQXR direttamente, poiché quasi tutte le impostazioni possono essere configurate tramite i comandi di

amministrazione MQSC o MQ Explorer. Se si decide di modificare direttamente il file, arrestare il gestore code prima di apportare le modifiche. Vedere Proprietà MQXR.

Informazioni su questa attività

Il supporto IBM WebSphere MQ Telemetry per IBM WebSphere MQ Explorer include una procedura guidata e una procedura di comando di esempio `sampleMQM`. Impostano una configurazione iniziale utilizzando l'ID utente guest; consultare Verifica dell'installazione di IBM WebSphere MQ Telemetry utilizzando i programmi di esempio IBM WebSphere MQ Explorer e IBM WebSphere MQ Telemetry.

Seguire i passi in questa attività per configurare manualmente IBM WebSphere MQ Telemetry utilizzando diversi schemi di autorizzazione.

Procedura

1. Aprire una finestra comandi nella directory degli esempi di telemetria.

   La directory degli esempi di telemetria è *WMQ program installation directory*\mqxr\samples.
2. Creare la coda di trasmissione di telemetria.

   ```
   echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
   ```

   Quando il servizio di telemetria (MQXR) viene avviato per la prima volta, crea SYSTEM.MQTT.TRANSMIT.QUEUE.

   Viene creato manualmente in questa attività, perché SYSTEM.MQTT.TRANSMIT.QUEUE deve esistere prima che venga avviato il servizio di telemetria (MQXR), per autorizzare l'accesso ad esso.
3. Imposta la coda di trasmissione predefinita

   ```
   echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
   ```

   Figura 20. Imposta coda di trasmissione predefinita

   Quando il servizio MQXR (telemetria) viene avviato per la prima volta, non modifica il gestore code per rendere SYSTEM.MQTT.TRANSMIT.QUEUE la coda di trasmissione predefinita.

   Per rendere SYSTEM.MQTT.TRANSMIT.QUEUE la coda di trasmissione predefinita, modificare la proprietà della coda di trasmissione predefinita. Modificare la proprietà utilizzando IBM WebSphere MQ Explorer o con il comando in Figura 20 a pagina 120.

   La modifica della coda di trasmissione predefinita potrebbe interferire con la configurazione esistente. Il motivo per cui si modifica la coda di trasmissione predefinita in SYSTEM.MQTT.TRANSMIT.QUEUE è rendere più semplice l'invio di messaggi direttamente ai client MQTT. Senza modificare la coda di trasmissione predefinita, è necessario aggiungere una definizione di coda remota per ogni client che riceve messaggi IBM WebSphere MQ ; consultare “Invio diretto di un messaggio ad un cliente” a pagina 123.
4. Seguire una procedura in “Autorizzazione dei client MQTT ad accedere agli oggetti WebSphere MQ” a pagina 125 per creare uno o più ID utente. Gli ID utente hanno l'autorità di pubblicare, sottoscrivere e inviare pubblicazioni ai client MQTT.
5. Installa il servizio di telemetria (MQXR)

   ```
   type
   installMQXRService_win.mqsc | runmqsc qMgr
   ```
6. Avvia servizio

   ```
   echo START SERVICE(SYSTEM.MQXR.SERVICE) | runmqsc qMgr
   ```

   Il servizio di telemetria (MQXR) viene avviato automaticamente all'avvio del gestore code.

   Viene avviato manualmente in questa attività perché il gestore code è già in esecuzione.

7. Utilizzando IBM WebSphere MQ Explorer, configurare i canali di telemetria per accettare le connessioni dai client MQTT.

   I canali di telemetria devono essere configurati in modo che le relative identità siano uno degli ID utente definiti al passo 4.

   Vedere anche DEFINE CHANNEL (MQTT).
8. Verificare la configurazione eseguendo il client di esempio.

   Per consentire al client di esempio di utilizzare il proprio canale di telemetria, il canale deve autorizzare il client a pubblicare, sottoscrivere e ricevere pubblicazioni. Il client di esempio si collega al canale di telemetria sulla porta 1883 per impostazione predefinita. Vedere inoltre IBM WebSphere MQ Telemetry programmi di esempio.

Creazione manuale di SYSTEM.MQXR.SERVICE

Figura 21 a pagina 121 mostra il comando **runmqsc** per creare manualmente il SYSTEM.MQXR.SERVICE su Windows.

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
    CONTROL(QMGR) +
    DESCR('Manages clients using MQXR protocols such as MQTT') +
    SERVTYPE(SERVER) +
    STARTCMD('+MQ_INSTALL_PATH+\mqxr\bin\runMQXRService.bat') +
    STARTARG('-m +QMNAME+  -d "+MQ_Q_MGR_DATA_PATH+\." -g "+MQ_DATA_PATH+\."') +
    STOPCMD('+MQ_INSTALL_PATH+\mqxr\bin\endMQXRService.bat') +
    STOPARG('-m +QMNAME+') +
    STDOUT('+MQ_Q_MGR_DATA_PATH+\mqxr.stdout') +
    STDERR('+MQ_Q_MGR_DATA_PATH+\mqxr.stderr')
```

Figura 21. installMQXRService_win.mqsc

Configurare l'accodamento distribuito per inviare messaggi ai client MQTT

Le applicazioni IBM WebSphere MQ possono inviare messaggi dei client MQTT v3 pubblicando su una sottoscrizione creata da un client o inviando direttamente un messaggio. Indipendentemente dal metodo utilizzato, il messaggio viene inserito in SYSTEM.MQTT.TRANSMIT.QUEUEe inviato al client dal servizio di telemetria (MQXR). Esistono diversi modi per inserire un messaggio su SYSTEM.MQTT.TRANSMIT.QUEUE.

Pubblicazione di un messaggio in risposta a una sottoscrizione client MQTT

Il servizio di telemetria (MQXR) crea una sottoscrizione per conto del client MQTT. Il client è la destinazione per tutte le pubblicazioni che corrispondono alla sottoscrizione inviata dal client. I servizi di telemetria inoltrano le pubblicazioni corrispondenti al client.

Un client MQTT è connesso a WebSphere MQ come gestore code, con il relativo nome gestore code impostato su ClientIdentifier. La destinazione delle pubblicazioni da inviare al client è una coda di trasmissione, SYSTEM.MQTT.TRANSMIT.QUEUE. Il servizio di telemetria inoltra i messaggi su SYSTEM.MQTT.TRANSMIT.QUEUE ai client MQTT, utilizzando il nome gestore code di destinazione come chiave per un client specifico.

Il servizio di telemetria (MQXR) apre la coda di trasmissione utilizzando ClientIdentifier come nome gestore code. Il servizio di telemetria (MQXR) passa l'handle dell'oggetto della coda alla chiamata MQSUB per inoltrare le pubblicazioni che corrispondono alla sottoscrizione client. Nella risoluzione del nome oggetto, il ClientIdentifier viene creato come nome del gestore code remoto e la coda di trasmissione deve risolversi in SYSTEM.MQTT.TRANSMIT.QUEUE. Utilizzando la risoluzione standard del nome oggetto WebSphere MQ , *ClientIdentifier* viene risolto come segue; consultare Tabella 6 a pagina 122.

1. *ClientIdentifier* non corrisponde a nulla.

   ClientIdentifier è un nome gestore code remoto. Non corrisponde al nome del gestore code locale, a un alias del gestore code o a un nome della coda di trasmissione.

Il nome della coda non è definito. Attualmente, il servizio di telemetria (MQXR) imposta SYSTEM.MQTT.PUBLICATION.QUEUE come nome della coda. Un client MQTT v3 non supporta le code, quindi il nome della coda risolta viene ignorato dal client.

La proprietà del gestore code locale, `Coda di trasmissione predefinita`, deve essere impostata su SYSTEM.MQTT.TRANSMIT.QUEUE, in modo che la pubblicazione venga inserita in SYSTEM.MQTT.TRANSMIT.QUEUE per essere inviata al client.

2. *ClientIdentifier* corrisponde a un alias del gestore code denominato *ClientIdentifier*.

   ClientIdentifier è un nome gestore code remoto. Corrisponde al nome di un alias del gestore code.

   L'alias del gestore code deve essere definito con *ClientIdentifier* come nome del gestore code remoto.

   Impostando il nome della coda di trasmissione nella definizione alias del gestore code, non è necessario che la trasmissione predefinita sia impostata su SYSTEM.MQTT.TRANSMIT.QUEUE.

Tabella 6. Risoluzione dei nomi di un alias del gestore code MQTT

	Immissione		**Output**		
ClientIdentifier	**Nome del gestore code**	**Nome coda**	**Nome del gestore code**	**Nome coda**	**Coda di trasmissione**
Non corrisponde a nulla	*ClientIdentifier*	*non definito*	*ClientIdentifier*	*non definito*	Coda di trasmissione predefinita. SYSTEM.MQTT.TRANSMIT.QUEUE
Corrisponde a un alias del gestore code denominato *ClientIdentifier*	*ClientIdentifier*	*non definito*	*ClientIdentifier*	*non definito*	SYSTEM.MQTT.TRANSMIT.QUEUE

Per ulteriori informazioni sulla risoluzione dei nomi, consultare Risoluzione dei nomi.

Qualsiasi programma WebSphere MQ può essere pubblicato nello stesso argomento. La pubblicazione viene inviata ai relativi sottoscrittori, inclusi i client MQTT v3 che hanno una sottoscrizione all'argomento.

Se un argomento di gestione viene creato in un cluster, con l'attributo CLUSTER(*clusterName*), qualsiasi applicazione nel cluster può pubblicare sul client; ad esempio:

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

Figura 22. Definizione di un argomento cluster in Windows

Nota: Non fornire a SYSTEM.MQTT.TRANSMIT.QUEUE un attributo cluster.

I sottoscrittori e i publisher del client MQTT possono connettersi a gestori code differenti. I sottoscrittori e i publisher possono far parte dello stesso cluster o essere collegati da una gerarchia di pubblicazione / sottoscrizione. La pubblicazione viene consegnata dal publisher al sottoscrittore utilizzando WebSphere MQ.

Invio diretto di un messaggio ad un cliente

Un'alternativa a un client che crea una sottoscrizione e riceve una pubblicazione che corrisponde all'argomento della sottoscrizione, invia un messaggio direttamente a un client MQTT v3 . Le applicazioni client MQTT V3 non possono inviare messaggi direttamente, ma altre applicazioni, come le applicazioni WebSphere MQ , possono farlo.

L'applicazione WebSphere MQ deve conoscere il `ClientIdentifier` del client MQTT v3 . Poiché i client MQTT v3 non hanno code, il nome della coda di destinazione viene passato al metodo MQTT v3 v3 `messageArrived` come nome argomento. Ad esempio, in un programma MQI, creare un descrittore oggetto con il client come `ObjectQmgrName`:

```
MQOD.ObjectQmgrName = ClientIdentifier;
MQOD.ObjectName = name;
```

Figura 23. Descrittore oggetto MQI per inviare un messaggio a una destinazione client MQTT v3

Se l'applicazione viene scritta utilizzando JMS, creare una destinazione point-to-point, ad esempio:

```
javax.jms.Destination jmsDestination =
                      (javax.jms.Destination)jmsFactory.createQueue
                      ("queue://ClientIdentifier/name");
```

Figura 24. Destinazione JMS per l'invio di un messaggio a un client MQTT v3

Per inviare un messaggio non richiesto a un client MQTT utilizzare una definizione di coda remota. Il nome del gestore code remoto deve essere risolto nel `ClientIdentifier` del client. La coda di trasmissione deve essere risolta in `SYSTEM.MQTT.TRANSMIT.QUEUE`; consultare Tabella 7 a pagina 123. Il nome della coda remota può essere qualsiasi cosa. Il client lo riceve come stringa di argomenti.

Tabella 7. Risoluzione del nome di una definizione di coda remota del client MQTT

Immissione		**Output**		
Nome coda	**Nome del gestore code**	**Nome coda**	**Nome del gestore code**	**Coda di trasmissione**
Nome della definizione della coda remota	Nome gestore code locale o vuoto	Nome coda remota utilizzato come stringa argomento	*ClientIdentifier*	SYSTEM.MQTT.TRANSMIT.QUEUE

Se il client è connesso, il messaggio viene inviato direttamente al client MQTT, che richiama il metodo `messageArrived` ; consultare MetodomessageArrived.

Se il client si è disconnesso con una sessione persistente, il messaggio viene memorizzato in `SYSTEM.MQTT.TRANSMIT.QUEUE`; consultare Sessioni stateless e stateful MQTT . Viene inoltrato al cliente quando il client si riconnette di nuovo alla sessione.

Se si invia un messaggio non persistente, questo viene inviato al client con QoS (quality of service) "al massimo una volta", `QoS=0`. Se si invia un messaggio permanente direttamente a un client, per impostazione predefinita viene inviato con QoS (quality of service) "esattamente una volta", `QoS=2`. Poiché il client potrebbe non avere un meccanismo di persistenza, può ridurre la qualità del servizio che accetta per i messaggi inviati direttamente. Per ridurre la qualità del servizio per i messaggi inviati direttamente a un client, sottoscrivere la sezione `DEFAULT.QoS`. Specificare la qualità massima del servizio che il client può supportare.

Identificazione, autorizzazione e autenticazione del client MQTT

Il servizio di telemetria (MQXR) pubblica o sottoscrive argomenti WebSphere MQ per conto dei client MQTT, utilizzando i canali MQTT. L'amministratore WebSphere MQ configura l'identità del canale MQTT utilizzata per l'autorizzazione WebSphere MQ . L'amministratore può definire un'identità comune per il canale oppure utilizzare il `Nome utente` o `ClientIdentifier` di un client connesso al canale.

Il servizio di telemetria (MQXR) può autenticare il client utilizzando il `Nome utente` fornito dal client o utilizzando un certificato client. Il `Nome utente` viene autenticato utilizzando una password fornita dal client.

Per riassumere: l'identificazione del client è la selezione dell'identità del client. In base al contesto, il client viene identificato da `ClientIdentifier`, `Username`, un'identità client comune creata dall'amministratore o un certificato client. L'identificativo client utilizzato per il controllo di autenticità non deve essere lo stesso identificativo utilizzato per l'autorizzazione.

I programmi client MQTT impostano il `Nome utente` e la `Password` inviati al server utilizzando un canale MQTT. Possono anche impostare le proprietà SSL richieste per codificare e autenticare la connessione. L'amministratore decide se autenticare il canale MQTT e come autenticare il canale.

Per autorizzare un client MQTT ad accedere agli oggetti WebSphere MQ , autorizzare il `ClientIdentifier`o il `Nome utente` del client oppure autorizzare un'identità client comune. Per consentire a un client di connettersi a WebSphere MQ, autenticare il `Nome utente`o utilizzare un certificato del client. Configurare JAAS per autenticare il `Nome utente`e configurare SSL per autenticare un certificato client.

Se imposti una `Password` sul client, crittografa la connessione utilizzando VPN o configura il canale MQTT per utilizzare SSL, per mantenere la password privata.

È difficile gestire i certificati client. Per questo motivo, se i rischi associati all'autenticazione della password sono accettabili, l'autenticazione della password viene spesso utilizzata per autenticare i client.

Se esiste un modo sicuro per gestire e memorizzare il certificato client, è possibile fare affidamento sull'autenticazione del certificato. Tuttavia, raramente i certificati possono essere gestiti in modo sicuro nei tipi di ambienti in cui viene utilizzata la telemetria. Invece, l'autenticazione delle unità che utilizzano i certificati client è completata dall'autenticazione delle password client sul server. A causa della complessità aggiuntiva, l'utilizzo dei certificati client è limitato alle applicazioni altamente sensibili. L'uso di due forme di autenticazione è chiamato autenticazione a due fattori. È necessario conoscere uno dei fattori, ad esempio una password, e l'altro, ad esempio un certificato.

In un'applicazione altamente sensibile, ad esempio un dispositivo con chip e pin, il dispositivo viene bloccato durante la fabbricazione per evitare la manomissione dell'hardware e del software interni. Un certificato client sicuro, limitato nel tempo, viene copiato sul dispositivo. Il dispositivo viene distribuito nell'ubicazione in cui deve essere utilizzato. Un'ulteriore autenticazione viene eseguita ogni volta che il dispositivo viene utilizzato, utilizzando una parola d'ordine o un altro certificato da una smart card.

Identità e autorizzazione client MQTT

Utilizzare `ClientIdentifier`, `Username`o un'identità client comune per l'autorizzazione ad accedere agli oggetti WebSphere MQ .

L'amministratore WebSphere MQ ha tre opzioni per selezionare l'identità del canale MQTT. L'amministratore effettua la scelta quando definisce o modifica il canale MQTT utilizzato dal client. L'identità viene utilizzata per autorizzare l'accesso agli argomenti WebSphere MQ . Le diverse opzioni sono:

1. L'identificativo del client.
2. Un'identità fornita dall'amministratore per il canale.
3. Il `Nome utente` passato dal client MQTT.

`Nome utente` è un attributo della classe di opzioni MqttConnect. Deve essere impostato prima che il client si colleghi al servizio. Il valore predefinito è null.

Utilizzare il comando WebSphere MQ **setmqaut** per selezionare quali oggetti e quali azioni sono autorizzati ad essere utilizzati dall'identit. associata al canale MQTT. Ad esempio, per autorizzare un'identità del canale, `MQTTClient`, fornita dall'amministratore del gestore code, `QM1`:

```
setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

Autorizzazione dei client MQTT ad accedere agli oggetti WebSphere MQ

Seguire questa procedura per autorizzare i client MQTT a pubblicare e sottoscrivere gli oggetti WebSphere MQ . La procedura segue quattro modelli di controllo accessi alternativi.

Prima di iniziare

I client MQTT sono autorizzati ad accedere agli oggetti in WebSphere MQ mediante l'assegnazione di un'identità quando si collegano a un canale di telemetria. L'amministratore di WebSphere MQ configura il canale di telemetria utilizzando WebSphere MQ Explorer per fornire a un client uno dei seguenti tre tipi di identità:

1. `ClientIdentifier`
2. `Nome utente`
3. Un nome che l'amministratore assegna al canale.

Indipendentemente dal tipo utilizzato, l'identit ... deve essere definita in WebSphere MQ come principal dal servizio di autorizzazione installato. Il servizio di autorizzazione predefinito in Windows o Linux è denominato OAM (Object Authority Manager). Se si utilizza OAM, l'identità deve essere definita come ID utente.

Utilizzare l'identità per fornire a un client, o a una raccolta di client, l'autorizzazione a pubblicare o sottoscrivere argomenti definiti in WebSphere MQ. Se un client MQTT ha sottoscritto un argomento, utilizzare l'identità per fornire l'autorizzazione a ricevere le pubblicazioni risultanti.

È difficile gestire un sistema con decine di migliaia di client MQTT, ognuno dei quali richiede autorizzazioni di accesso individuali. Una soluzione è definire identità comuni e associare singoli client MQTT a una delle identità comuni. Definire tutte le identità comuni necessarie per definire diverse combinazioni di autorizzazioni. Un'altra soluzione è scrivere il proprio servizio di autorizzazione che può trattare più facilmente con migliaia di utenti rispetto al sistema operativo.

È possibile combinare i client MQTT in identità comuni in due modi, utilizzando OAM:

1. Definire più canali di telemetria, ciascuno con un diverso ID utente assegnato dall'amministratore mediante WebSphere MQ Explorer. I client che si collegano utilizzando numeri di porta TCP/IP differenti sono associati a canali di telemetria differenti e sono assegnati a identità differenti.
2. Definire un singolo canale di telemetria, ma ciascun client deve selezionare un `Nome utente` da una piccola serie di ID utente. L'amministratore configura il canale di telemetria per selezionare il client `Nome utente` come sua identità.

In questa attività, l'identità del canale di telemetria viene denominata *`mqttUser`*, indipendentemente da come è impostata. Se le raccolte di client utilizzano identità differenti, utilizzare più *`mqttUsers`*, uno per ogni raccolta di client. Poiché l'attività utilizza OAM, ogni *`mqttUser`* deve essere un ID utente.

Informazioni su questa attività

In questa attività, è possibile scegliere tra quattro modelli di controllo accessi che è possibile personalizzare in base a requisiti specifici. I modelli differiscono nella loro granularità del controllo accessi.

- “Nessun controllo accessi” a pagina 126
- “Controllo degli accessi generici” a pagina 126
- “Controllo accessi a grana media” a pagina 126
- “Controllo dell'accesso dettagliato” a pagina 126

Il risultato dei modelli è quello di assegnare *mqttUsers* serie di autorizzazioni per la pubblicazione e la sottoscrizione a WebSphere MQe ricevere pubblicazioni da WebSphere MQ.

Nessun controllo accessi
Ai client MQTT viene fornita l'autorizzazione di gestione WebSphere MQ e può eseguire qualsiasi azione su qualsiasi oggetto.

Procedura

1. Creare un ID utente *mqttUser* che funga da identità di tutti i clienti MQTT.
2. Aggiungere *mqttUser* al gruppo mqm ; consultare Aggiunta di un utente a un gruppo in Windowsoppure Aggiunta di un utente a un gruppo in Linux

Controllo degli accessi generici
I client MQTT hanno l'autorità per pubblicare e sottoscrivere e per inviare messaggi ai client MQTT. Non dispongono dell'autorizzazione per eseguire altre azioni o per accedere ad altri oggetti.

Procedura

1. Creare un ID utente *mqttUser* che funga da identità di tutti i clienti MQTT.
2. Autorizzare *mqttUser* a pubblicare e sottoscrivere tutti gli argomenti e a inviare pubblicazioni ai client MQTT.

```
setmqaut -m qMgr -t topic -n SYSTEM.BASE.TOPIC -p mqttUser -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqttUser -all +put
```

Controllo accessi a grana media
I client MQTT sono divisi in diversi gruppi per pubblicare e sottoscrivere diverse serie di argomenti e per inviare messaggi a client MQTT.

Procedura

1. Creare più ID utente, *mqttUsers*e più argomenti di gestione nella struttura ad albero degli argomenti di pubblicazione / sottoscrizione.
2. Autorizzare *mqttUsers* differenti per argomenti differenti.

```
setmqaut -m qMgr -t topic -n topic1 -p mqttUserA -all +pub +sub
setmqaut -m qMgr -t topic -n topic2 -p mqttUserB -all +pub +sub
```

3. Creare un gruppo *mqtt*e aggiungere tutti i *mqttUsers* al gruppo.
4. Autorizzare *mqtt* a inviare argomenti ai client MQTT.

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Controllo dell'accesso dettagliato
I client MQTT vengono incorporati in un sistema di controllo accessi esistente, che autorizza i gruppi ad eseguire azioni sugli oggetti.

Informazioni su questa attività

Un ID utente viene assegnato ad uno o più gruppi del sistema operativo in base alle autorizzazioni richieste. Se le applicazioni WebSphere MQ stanno pubblicando e sottoscrivendo lo stesso spazio argomenti dei client MQTT, utilizzare questo modello. I gruppi vengono indicati come Publish*X*, Subscribe*Y*e mqtt

Publish*X*
I membri dei gruppi Publish*X* possono pubblicare in *topicX*.

Subscribe*Y*
I membri dei gruppi Subscribe*Y* possono sottoscrivere *topicY*.

mqtt
I membri del gruppo *mqtt* possono inviare pubblicazioni ai clienti MQTT.

Procedura

1. Creare più gruppi Publish*X* e Subscribe*Y* assegnati a più argomenti di gestione nella struttura ad albero degli argomenti di pubblicazione / sottoscrizione.
2. Creare un gruppo mqtt.
3. Creare più ID utente, *mqttUsers*, e aggiungere gli utenti a uno qualsiasi dei gruppi, a seconda di ciò che sono autorizzati a fare.
4. Autorizzare diversi gruppi Publish*X* e Subscribe*X* a diversi argomenti e autorizzare il gruppo *mqtt* a inviare messaggi a client MQTT.

```
setmqaut -m qMgr -t topic -n topic1 -p PublishX -all +pub
setmqaut -m qMgr -t topic -n topic1 -p SubscribeX -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Autenticazione client MQTT utilizzando una parola d'ordine

Autenticare il Nome utente utilizzando la password client. È possibile autenticare il client utilizzando un'identità diversa rispetto all'identità utilizzata per autorizzare il client a pubblicare e sottoscrivere argomenti.

Il servizio di telemetria (MQXR) utilizza JAAS per autenticare il client Username. JAAS utilizza la Password fornita dal client MQTT.

L'amministratore di WebSphere MQ decide se autenticare il Nome utenteo meno, configurando il canale MQTT a cui si connette un client. I client possono essere assegnati a canali diversi e ciascun canale può essere configurato per autenticare i propri client in modi differenti. Utilizzando JAAS, è possibile configurare quali metodi devono autenticare il client e quali possono facoltativamente autenticare il client.

La scelta dell'identità per l'autenticazione non influisce sulla selezione dell'identità per l'autorizzazione. È possibile impostare un'identità comune per l'autorizzazione per comodità di gestione, ma autenticare ciascun utente per utilizzare tale identità. La seguente procedura descrive la procedura per autenticare i singoli utenti per utilizzare un'identità comune:

1. L'amministratore di WebSphere MQ imposta l'identità del canale MQTT su qualsiasi nome, ad esempio MQTTClientUser, utilizzando WebSphere MQ Explorer.
2. L'amministratore di WebSphere MQ autorizza MQTTClient a pubblicare e sottoscrivere qualsiasi argomento:

```
setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

3. Lo sviluppatore dell'applicazione client MQTT crea un oggetto MqttConnectOptions e imposta Nome utente e Password prima di connettersi al server.
4. Lo sviluppatore della sicurezza crea un JAAS LoginModule per autenticare il nome utente con Password e lo include nel file di configurazione JAAS .
5. L'amministratore WebSphere MQ configura il canale MQTT per autenticare il UserName del client utilizzando JAAS.

Autenticazione client MQTT tramite SSL

Le connessioni tra il client MQTT e il gestore code vengono sempre avviate dal client MQTT. Il client MQTT è sempre il client SSL. L'autenticazione client del server e l'autenticazione server del client MQTT sono entrambe facoltative.

Fornire al client un certificato digitale firmato privato, è possibile autenticare il client MQTT su IBM WebSphere MQ. L'amministratore IBM WebSphere MQ può forzare i clienti MQTT ad autenticarsi

al gestore code utilizzando SSL. È possibile richiedere l'autenticazione client solo come parte dell'autenticazione reciproca.

Come alternativa all'utilizzo di SSL, alcuni tipi di VPN (Virtual Private Network), ad esempio IPsec, autenticano gli endpoint di una connessione TCP/IP. VPN codifica ogni pacchetto IP che passa attraverso la rete. Una volta che tale connessione VPN è stabilita, è stata creata una rete attendibile. È possibile connettere i client MQTT ai canali di telemetria utilizzando TCP/IP sulla rete VPN.

L'autenticazione client che utilizza SSL si basa sul fatto che il client abbia un segreto. Il segreto è la chiave privata del client nel caso di un certificato autofirmato o una chiave fornita da un'autorità di certificazione. La chiave viene utilizzata per firmare il certificato digitale del client. Chiunque sia in possesso della corrispondente chiave pubblica può verificare il certificato digitale. I certificati possono essere attendibili o, se sono concatenati, è possibile risalire attraverso una catena di certificati a un certificato root attendibile. La verifica del client invia al server tutti i certificati presenti nella catena di certificati fornita dal client. Il server controlla la catena di certificati finché non trova un certificato attendibile. Il certificato attendibile è il certificato pubblico generato da un certificato autofirmato oppure un certificato root in genere emesso da un'autorità di certificazione. Come ultimo passo facoltativo, il certificato attendibile può essere confrontato con un elenco di revoche di certificati in tempo reale.

Il certificato attendibile potrebbe essere emesso da un'autorità di certificazione ed essere già incluso nell'archivio certificati JRE. Potrebbe essere un certificato autofirmato oppure un certificato che è stato aggiunto al keystore del canale di telemetria come certificato attendibile.

Nota: Il canale di telemetria include un keystore/truststore combinato che contiene sia le chiavi private per uno o più canali di telemetria, sia eventuali certificati pubblici necessari per autenticare i client. Poiché un canale SSL deve disporre di un keystore ed è lo stesso file del truststore del canale, non viene mai fatto riferimento all'archivio certificati JRE. L'implicazione è che se l'autenticazione di un client richiede un certificato root CA, è necessario posizionare il certificato root nel keystore per il canale, anche se il certificato root CA si trova già nell'archivio certificati JRE. All'archivio certificati JRE non viene mai fatto riferimento.

Occorre considerare le minacce che l'autenticazione client deve contrastare e i ruoli che il client e il server ricoprono nel contrastare le minacce. L'autenticazione del certificato client da sola non è sufficiente a impedire l'accesso non autorizzato a un sistema. Se qualcun altro ha accesso al dispositivo client, tale dispositivo non effettua necessariamente azioni autorizzate dal titolare del certificato. Non fare mai affidamento su una singola difesa contro attacchi indesiderati. Utilizzare almeno un approccio di autenticazione a due fattori e integrare il possesso di un certificato con la conoscenza di informazioni private. Ad esempio, utilizzare JAAS e autenticare il client utilizzando una password emessa dal server.

La minaccia principale al certificato client è che esso cada nelle mani sbagliate. Il certificato è contenuto in un keystore protetto da password nel client. Come viene collocato nel keystore? In che modo il client MQTT ottiene la password sul keystore? Quanto è sicura la protezione con password? I dispositivi di telemetria sono spesso facili da rimuovere e possono essere oggetto di attacchi in privato. Il dispositivo deve essere a prova di manomissione? La distribuzione e la protezione dei certificati sul lato client è notoriamente difficile: si chiama problema di gestione chiavi.

Una minaccia secondaria è che il dispositivo venga utilizzato erroneamente per accedere ai server in modi non intenzionali. Ad esempio, se l'applicazione MQTT viene manomessa, potrebbe essere possibile utilizzare un punto debole della configurazione del server utilizzando l'identità del client autenticato.

Per autenticare un client MQTT tramite SSL, configurare il canale di telemetria e il client.

-
-

Configurazione del canale di telemetria per l'autenticazione del client MQTT mediante SSL

L'amministratore IBM WebSphere MQ configura i canali di telemetria sul server. Ogni canale è configurato per accettare una connessione TCP/IP su un numero di porta diverso. I canali SSL sono configurati con l'accesso protetto da passphrase ai file chiave. Se un canale SSL è definito senza passphrase o file di chiavi, il canale non accetta connessioni SSL.

Impostare la proprietà com.ibm.mq.MQTT.ClientAuth di un canale di telemetria SSL su REQUIRED per forzare tutti i client che si connettono su tale canale a fornire la prova di aver verificato i certificati digitali. I certificati client vengono autenticati utilizzando i certificati dalle autorità di certificazione, che portano a un certificato root attendibile. Se il certificato client è autofirmato o è firmato da un certificato che proviene da un'autorità di certificazione, i certificati firmati pubblicamente del client o dell'autorità di certificazione devono essere memorizzati in modo sicuro sul server.

Inserire il certificato client firmato pubblicamente o il certificato dall'autorità di certificazione nel keystore del canale di telemetria. Sul server, i certificati firmati pubblicamente vengono memorizzati nello stesso file chiave dei certificati firmati privatamente, piuttosto che in un truststore separato.

Il server verifica la firma di tutti i certificati client che vengono inviati utilizzando tutti i certificati pubblici e le suite di cifratura di cui dispone. Il server verifica la catena di chiavi. Il gestore code può essere configurato per verificare il certificato rispetto all'elenco di revoca certificati. La proprietà dell'elenco nomi di revoca del gestore code è SSLCRLNL.

Se uno qualsiasi dei certificati inviati da un client viene verificato da un certificato nel keystore del server, il client viene autenticato.

L'amministratore di WebSphere MQ può configurare lo stesso canale di telemetria per utilizzare JAAS per controllare il UserName o ClientIdentifier del client con la Passworddel client.

È possibile utilizzare lo stesso keystore per più canali di telemetria.

La verifica di almeno un certificato digitale nel keystore del client protetto da password sulla periferica autentica il client sul server. Il certificato digitale viene utilizzato solo per l'autenticazione da WebSphere MQ. Non viene utilizzato per verificare l'indirizzo TCP/IP del client o per impostare l'identità del client per l'autorizzazione o l'account. L'identità del client adottata dal server è il nome utente o ClientIdentifier del client oppure un'identità creata dall'amministratore di WebSphere MQ .

È inoltre possibile utilizzare le suite di cifratura SSL per l'autenticazione client. Questo è un elenco alfabetico delle suite di cifratura SSL attualmente supportate:

- SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- SSL_DH_anon_WITH_AES_128_CBC_SHA
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_DSS_WITH_AES_128_CBC_SHA
- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA
- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5

- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V 7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V 7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V 7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V 7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V 7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V 7.5.0.2 Se si prevede di utilizzare le suite di cifratura SHA-2 , consultare Requisiti di sistema per l'utilizzo di SHA-2 cifrate suite con canali MQTT.

Concetti correlati

“Configurazione del canale di telemetria per l'autenticazione del canale mediante SSL” a pagina 131
L'amministratore IBM WebSphere MQ configura i canali di telemetria sul server. Ogni canale è configurato per accettare una connessione TCP/IP su un numero di porta diverso. I canali SSL sono configurati con l'accesso protetto da passphrase ai file chiave. Se un canale SSL è definito senza passphrase o file di chiavi, il canale non accetta connessioni SSL.

CipherSpecs e CipherSuites

Riferimenti correlati

DEFINISCI CANALE (MQTT)

MODIFICA CANALE (MQTT)

Autenticazione del canale di telemetria mediante SSL

Le connessioni tra il client MQTT e il gestore code vengono sempre avviate dal client MQTT. Il client MQTT è sempre il client SSL. L'autenticazione client del server e l'autenticazione server del client MQTT sono entrambe facoltative.

Il client tenta sempre di autenticare il server, a meno che il client non sia configurato in modo da utilizzare una CipherSpec che supporta la connessione anonima. Se l'autenticazione non riesce, la connessione non viene stabilita.

Come alternativa all'utilizzo di SSL, alcuni tipi di VPN (Virtual Private Network), ad esempio IPsec, autenticano gli endpoint di una connessione TCP/IP. VPN codifica ogni pacchetto IP che passa attraverso la rete. Una volta che tale connessione VPN è stabilita, è stata creata una rete attendibile. È possibile connettere i client MQTT ai canali di telemetria utilizzando TCP/IP sulla rete VPN.

L'autenticazione server tramite SSL autentica il server al quale si sta per inviare informazioni riservate. Il client esegue i controlli che corrispondono ai certificati inviati dal server, rispetto ai certificati collocati nel relativo truststore o nel relativo `cacerts` archivio JRE.

L'archivio certificati JRE è un file JKS, `cacerts`. Si trova in `JRE InstallPath\lib\security\`. È installato con la password predefinita `changeit`. È possibile memorizzare certificati attendibili nell'archivio certificati JRE o nel truststore del client. Non è possibile utilizzare entrambi gli archivi. Utilizzare il truststore client se si desidera tenere i certificati pubblici che il client ritiene attendibili separati dai certificati utilizzati da altre applicazioni Java. Utilizzare l'archivio certificati JRE se si desidera utilizzare un archivio certificati comune per tutte le applicazioni Java in esecuzione sul client. Se si decide di utilizzare l'archivio certificati JRE, riesaminare i certificati in esso contenuti per assicurarsi che siano attendibili.

È possibile modificare la configurazione JSSE fornendo un diverso provider di trust. È possibile personalizzare un provider di trust per eseguire diversi controlli su un certificato. In alcuni ambienti OGSi che hanno utilizzato il client MQTT, l'ambiente fornisce un provider di attendibilità differente.

Per autenticare il canale di telemetria utilizzando SSL, configurare il server e il client

-
-

Configurazione del canale di telemetria per l'autenticazione del canale mediante SSL

L'amministratore IBM WebSphere MQ configura i canali di telemetria sul server. Ogni canale è configurato per accettare una connessione TCP/IP su un numero di porta diverso. I canali SSL sono configurati con l'accesso protetto da passphrase ai file chiave. Se un canale SSL è definito senza passphrase o file di chiavi, il canale non accetta connessioni SSL.

Memorizzare il certificato digitale del server, firmato con la chiave privata, nel keystore che il canale di telemetria utilizzerà sul server. Memorizzare i certificati nella relativa catena di chiavi nel keystore, se si desidera trasmettere la catena di chiavi al client. Configurare il canale di telemetria utilizzando WebSphere MQ explorer per utilizzare SSL. Fornire il percorso al keystore e la passphrase per accedere al keystore. Se non si imposta il numero di porta TCP/IP del canale, il numero di porta del canale di telemetria SSL assume il valore predefinito 8883.

È anche possibile utilizzare suite di cifratura SSL per l'autenticazione del canale. Questo è un elenco alfabetico delle suite di cifratura SSL attualmente supportate:

- `SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DH_anon_EXPORT_WITH_RC4_40_MD5`
- `SSL_DH_anon_WITH_3DES_EDE_CBC_SHA`
- `SSL_DH_anon_WITH_AES_128_CBC_SHA`
- `SSL_DH_anon_WITH_DES_CBC_SHA`
- `SSL_DH_anon_WITH_RC4_128_MD5`
- `SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA`
- `SSL_DHE_DSS_WITH_AES_128_CBC_SHA`
- `SSL_DHE_DSS_WITH_DES_CBC_SHA`
- `SSL_DHE_DSS_WITH_RC4_128_SHA`
- `SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA`
- `SSL_DHE_RSA_WITH_AES_128_CBC_SHA`
- `SSL_DHE_RSA_WITH_DES_CBC_SHA`
- `SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5`

- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 Se si prevede di utilizzare le suite di cifratura SHA-2 , consultare Requisiti di sistema per l'utilizzo di SHA-2 cifrate suite con canali MQTT.

Concetti correlati

“Configurazione del canale di telemetria per l'autenticazione del client MQTT mediante SSL” a pagina 128
L'amministratore IBM WebSphere MQ configura i canali di telemetria sul server. Ogni canale è configurato per accettare una connessione TCP/IP su un numero di porta diverso. I canali SSL sono configurati con l'accesso protetto da passphrase ai file chiave. Se un canale SSL è definito senza passphrase o file di chiavi, il canale non accetta connessioni SSL.

CipherSpecs e CipherSuites

Riferimenti correlati

DEFINISCI CANALE (MQTT)

MODIFICA CANALE (MQTT)

Privacy di pubblicazione sui canali di telemetria

La riservatezza delle pubblicazioni MQTT inviate in entrambe le direzioni attraverso i canali di telemetria è protetta utilizzando SSL per codificare le trasmissioni sulla connessione.

I client MQTT che si collegano ai canali di telemetria utilizzano SSL per proteggere la riservatezza delle pubblicazioni trasmesse sul canale utilizzando la crittografia a chiave simmetrica. Poiché gli endpoint non

sono autenticati, non puoi considerare attendibile la sola crittografia del canale. Combina la protezione della privacy con il server o l'autenticazione reciproca.

Come alternativa all'utilizzo di SSL, alcuni tipi di VPN (Virtual Private Network), ad esempio IPsec, autenticano gli endpoint di una connessione TCP/IP. VPN codifica ogni pacchetto IP che passa attraverso la rete. Una volta che tale connessione VPN è stabilita, è stata creata una rete attendibile. È possibile connettere i client MQTT ai canali di telemetria utilizzando TCP/IP sulla rete VPN.

Per una configurazione tipica, che codifica il canale e autentica il server, consultare “Autenticazione del canale di telemetria mediante SSL” a pagina 130.

La crittografia delle connessioni SSL senza autenticare il server espone la connessione ad attacchi man - in - the - middle. Anche se le informazioni scambiate sono protette contro le intercettazioni, non sai con chi le stai scambiando. A meno che tu non controlli la rete, sei esposto a qualcuno che intercetta le tue trasmissioni IP e si maschera come l'endpoint.

Puoi creare una connessione SSL crittografata, senza autenticare il server, utilizzando uno scambio di chiavi Diffie - Hellman CipherSpec che supporta SSL anonimo. Il segreto master, condiviso tra client e server e utilizzato per codificare le trasmissioni SSL, viene stabilito senza scambiare un certificato server firmato privatamente.

Poiché le connessioni anonime non sono sicure, la maggior parte delle implementazioni SSL non utilizzano per impostazione predefinita CipherSpecsanonime. Se una richiesta client per la connessione SSL viene accettata da un canale di telemetria, il canale deve avere un keystore protetto da una passphrase. Per impostazione predefinita, poiché le implementazioni SSL non utilizzano CipherSpecsanonime, il keystore deve contenere un certificato firmato privatamente che il client può autenticare.

Se si utilizza CipherSpecsanonimo, il keystore del server deve esistere, ma non deve contenere alcun certificato firmato privatamente.

Un altro modo per stabilire una connessione crittografata consiste nel sostituire il provider di attendibilità sul client con la propria implementazione. Il provider di attendibilità non autentica il certificato del server, ma la connessione viene codificata.

Configurazione SSL di client e canali di telemetria MQTT

I clienti MQTT e il servizio WebSphere MQ Telemetry (MQXR) utilizzano JSSE (Java Secure Socket Extension) per collegare i canali di telemetria utilizzando SSL. Il daemon IBM WebSphere MQ Telemetry per i dispositivi non supporta SSL.

Configurare SSL per autenticare il canale di telemetria, il client MQTT e codificare il trasferimento dei messaggi tra client e il canale di telemetria.

Come alternativa all'utilizzo di SSL, alcuni tipi di VPN (Virtual Private Network), ad esempio IPsec, autenticano gli endpoint di una connessione TCP/IP. VPN codifica ogni pacchetto IP che passa attraverso la rete. Una volta che tale connessione VPN è stabilita, è stata creata una rete attendibile. È possibile connettere i client MQTT ai canali di telemetria utilizzando TCP/IP sulla rete VPN.

È possibile configurare la connessione tra un client MQTT Java e un canale di telemetria per utilizzare il protocollo SSL su TCP/IP. Ciò che è protetto dipende dalla modalità di configurazione di SSL per l'utilizzo di JSSE. A partire dalla configurazione più sicura, è possibile configurare tre diversi livelli di sicurezza:

1. Consenti la connessione solo ai client MQTT attendibili. Connettere un solo client MQTT ad un canale di telemetria attendibile. Crittografare i messaggi tra il client e il gestore code; consultare “Autenticazione client MQTT tramite SSL” a pagina 127
2. Connettere un solo client MQTT ad un canale di telemetria attendibile. Crittografare i messaggi tra il client e il gestore code; consultare “Autenticazione del canale di telemetria mediante SSL” a pagina 130.
3. Crittografare i messaggi tra il client e il gestore code; consultare “Privacy di pubblicazione sui canali di telemetria” a pagina 132.

Parametri di configurazione JSSE

Modificare i parametri JSSE per modificare il modo in cui è configurata una connessione SSL. I parametri di configurazione JSSE sono disposti in tre serie:

1. IBM WebSphere MQ Canale di telemetria
2. Client MQTT Java
3. JRE

Configurare i parametri del canale di telemetria utilizzando IBM WebSphere MQ Explorer. Impostare i parametri del client Java MQTT nell'attributo `MqttConnectionOptions.SSLProperties` . Modificare i parametri di sicurezza JRE modificando i file nella directory di sicurezza JRE sul client e sul server.

IBM WebSphere MQ canale di telemetria

Impostare tutti i parametri SSL del canale di telemetria utilizzando WebSphere MQ Explorer.

`ChannelName`

`ChannelName` è un parametro obbligatorio su tutti i canali.

Il nome del canale identifica il canale associato ad un numero di porta particolare. Denominare i canali per gestire le serie di client MQTT.

`PortNumber`

`PortNumber` è un parametro facoltativo su tutti i canali. Il valore predefinito è 1883 per i canali TCP e 8883 per quelli SSL.

Il numero di porta TCP/IP associato a questo canale. I client MQTT sono connessi a un canale specificando la porta definita per il canale. Se il canale ha proprietà SSL, il client deve connettersi utilizzando il protocollo SSL; ad esempio:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

`NomeKeyFile`

`KeyFileKeyFile` è un parametro obbligatorio per i canali SSL. Deve essere omesso per i canali TCP.

`KeyFileNome` è il percorso del keystore Java contenente certificati digitali forniti. Utilizzare JKS, JCEKS o PKCS12 come tipo di keystore sul server.

Identificare il tipo di keystore utilizzando una delle seguenti estensioni file:

```
.jks
.jceks
.p12
.pkcs12
```

Si presume che un keystore con qualsiasi altra estensione file sia un keystore JKS.

È possibile combinare un tipo di keystore sul server con altri tipi di keystore sul client.

Inserire il certificato privato del server nel keystore. Il certificato è noto come certificato server. Il certificato può essere autofirmato o parte di una concatenazione di certificati firmata da un'autorità di firma.

Se si sta utilizzando una catena di certificati, inserire i certificati associati nel keystore del server.

Il certificato del server e tutti i certificati nella sua catena di certificati vengono inviati ai client per autenticare l'identità del server.

Se `ClientAuth` è stato impostato su `Required`, il keystore deve contenere i certificati necessari per autenticare il client. Il client invia un certificato autofirmato, o una catena di certificati, e il client viene autenticato dalla prima verifica di questo materiale rispetto a un certificato nel

keystore. Utilizzando una catena di certificati, un certificato può verificare molti client, anche se vengono emessi con certificati client differenti.

PassPhrase

`PassPhrase` è un parametro obbligatorio per i canali SSL. Deve essere omesso per i canali TCP.

La passphrase viene utilizzata per proteggere il keystore.

ClientAuth

`ClientAuth` è un parametro SSL facoltativo. L'impostazione predefinita è nessuna autenticazione client. Deve essere omesso per i canali TCP.

Impostare `ClientAuth` se si desidera che il servizio di telemetria (MQXR) autentichi il client, prima di consentire al client di connettersi al canale di telemetria.

Se si imposta `ClientAuth`, il client deve connettersi al server utilizzando SSL e autenticare il server. In risposta all'impostazione di `ClientAuth`, il client invia il proprio certificato digitale al server e qualsiasi altro certificato nel relativo keystore. Il suo certificato digitale è noto come certificato client. Questi certificati vengono autenticati rispetto a quelli contenuti nel keystore del canale e nell'archivio JRE `cacerts` .

CipherSuite

`CipherSuite` è un parametro SSL facoltativo. Viene utilizzato il valore predefinito per provare tutti i CipherSpecsabilitati. Deve essere omesso per i canali TCP.

Se si desidera utilizzare una particolare CipherSpec, impostare `CipherSuite` sul nome della CipherSpec che deve essere utilizzata per stabilire la connessione SSL.

Il servizio di telemetria e il client MQTT negoziano un CipherSpec comune da tutti i CipherSpecs abilitati ad ogni estremità. Se una specifica CipherSpec viene specificata in una o in entrambe le estremità della connessione, deve corrispondere alla CipherSpec nell'altra estremità.

Installare ulteriori cifrature aggiungendo ulteriori fornitori a JSSE.

FIPS (Federal Information Processing Standards)

FIPS è un'impostazione facoltativa. Per impostazione predefinita non è impostato.

Nel pannello delle proprietà del gestore code o utilizzando **runmqsc**, impostare SSLFIPS. SSLFIPS specifica se devono essere utilizzati solo algoritmi certificati FIPS.

Elenco nomi revoche

L'elenco nomi di revoca è un'impostazione facoltativa. Per impostazione predefinita non è impostato.

Nel pannello delle proprietà del gestore code o utilizzando **runmqsc**, impostare SSLCRLNL. SSLCRLNL specifica un elenco nomi degli oggetti delle informazioni di autenticazione utilizzati per fornire le posizioni di revoca dei certificati.

Non viene utilizzato nessun altro parametro del gestore code che imposta proprietà SSL.

Client MQTT Java

Impostare le proprietà SSL per il client Java in `MqttConnectionOptions.SSLProperties`; ad esempio:

```
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
com.ibm.micro.client.mqttv3.MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setSSLProperties(sslClientProperties);
```

I nomi e i valori di specifiche proprietà sono descritti nella classe `MqttConnectOptions` . Per i collegamenti alla documentazione API client per le librerie client MQTT, vedere Riferimento di programmazione client MQTT.

Protocollo

`Protocollo` è facoltativo.

Il protocollo viene selezionato in negoziazione con il server di telemetria. Se si richiede un protocollo specifico, è possibile selezionarne uno. Se il server di telemetria non supporta il protocollo, la connessione ha esito negativo.

ContextProvider

`ContextProvider` è facoltativo.

KeyStore

`KeyStore` è facoltativo. Configurarlo se `ClientAuth` è impostato sul server per forzare l'autenticazione del client.

Inserire il certificato digitale del client, firmato utilizzando la chiave privata, nel keystore. Specificare il percorso e la password del keystore. Il tipo e il fornitore sono facoltativi. JKS è il tipo predefinito e IBMJCE è il fornitore predefinito.

Specificare un provider keystore differente per fare riferimento a una classe che aggiunge un nuovo provider keystore. Inoltrare il nome dell'algoritmo utilizzato dal provider keystore per istanziare il `KeyManagerFactory` impostando il nome del gestore chiavi.

TrustStore

`TrustStore` è facoltativo. È possibile inserire tutti i certificati attendibili nell'archivio JRE `cacerts` .

Configurare il truststore se si desidera avere un truststore diverso per il client. È possibile che non si configuri il truststore se il server utilizza un certificato emesso da una CA nota che ha già il certificato root memorizzato in `cacerts`.

Aggiungere il certificato firmato pubblicamente del server o il certificato root al truststore e specificare il percorso del truststore e la password. JKS è il tipo predefinito e IBMJCE è il fornitore predefinito.

Specificare un provider truststore differente per fare riferimento a una classe che aggiunge un nuovo provider truststore. Inoltrare il nome dell'algoritmo utilizzato dal provider truststore per creare un'istanza di `TrustManagerFactory` impostando il nome del gestore sicuro.

JRE

Altri aspetti della protezione Java che influiscono sul comportamento di SSL sia sul client che sul server sono configurati nel JRE. I file di configurazione su Windows si trovano in *Java Installation Directory*`\jre\lib\security`. Se si sta utilizzando il JRE fornito con IBM WebSphere MQ , il percorso è come mostrato nella seguente tabella:

Tabella 8. Percorsi file per piattaforma per i file di configurazione SSL JRE

Piattaforma	**Percorso file**
Windows	*WMQ Installation Directory*`\java\jre\lib\security`
Altre piattaforme UNIX and Linux	*WMQ Installation Directory*`/java/jre64/jre/lib/security`

Autorità di certificazione note

Il file `cacerts` contiene i certificati root delle autorità di certificazione note. `cacerts` viene utilizzato per impostazione predefinita, a meno che non si specifichi un truststore. Se si utilizza l'archivio `cacerts` o non si fornisce un truststore, è necessario esaminare e modificare l'elenco di firmatari in `cacerts` per soddisfare i propri requisiti di sicurezza.

È possibile aprire `cacerts` utilizzando il comando WebSphere MQ `strmqikm`.che esegue il programma di utilità IBM Key Management. Aprire `cacerts` come file JKS, utilizzando la password `changeit`. Modificare la password per proteggere il file.

Configurazione delle classi di sicurezza

Utilizzare il file `java.security` per registrare ulteriori provider di sicurezza e altre proprietà di sicurezza predefinite.

Autorizzazioni

Utilizzare il file `java.policy` per modificare le autorizzazioni concesse alle risorse. `javaws.policy` concede le autorizzazioni a `javaws.jar`

Livello di crittografia

Alcuni JRE vengono forniti con una crittografia di potenza ridotta. Se non è possibile importare le chiavi nei keystore, la causa potrebbe essere una codifica di livello ridotto. Prova ad avviare **`ikeyman`** utilizzando il comando **`strmqikm`** oppure scarica i file con giurisdizione limitata da IBM developer kits, Security information.

Importante: Il tuo paese di origine potrebbe avere restrizioni sull'importazione, il possesso, l'uso o la riesportazione in un altro paese, del software di crittografia. Prima di scaricare o utilizzare i file delle politiche illimitate, è necessario controllare le leggi del proprio Paese. Controllare le relative normative e le relative politiche relative all'importazione, al possesso, all'utilizzo e alla riesportazione del software di crittografia, per determinare se è consentito.

Modificare il provider di attendibilità per consentire al client di connettersi a qualsiasi server

L'esempio illustra come aggiungere un provider di attendibilità e farvi riferimento dal codice client MQTT. L'esempio non esegue alcuna autenticazione del client o del server. La connessione SSL risultante viene codificata senza essere autenticata.

Il frammento di codice in Figura 25 a pagina 137 imposta il provider e il gestore sicuro `AcceptAllProviders` per il client MQTT.

```
java.security.Security.addProvider(new AcceptAllProvider());
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.trustManager","TrustAllCertificates");
sslClientProperties.setProperty("com.ibm.ssl.trustStoreProvider","AcceptAllProvider");
conOptions.setSSLProperties(sslClientProperties);
```

Figura 25. Frammento di codice client MQTT

```
package com.ibm.mq.id;
public class AcceptAllProvider extends java.security.Provider {
    private static final long serialVersionUID = 1L;
    public AcceptAllProvider() {
        super("AcceptAllProvider", 1.0, "Trust all X509 certificates");
        put("TrustManagerFactory.TrustAllCertificates",
                AcceptAllTrustManagerFactory.class.getName());
    }
```

Figura 26. AcceptAllProvider.java

```
    protected static class AcceptAllTrustManagerFactory extends
            javax.net.ssl.TrustManagerFactorySpi {
        public AcceptAllTrustManagerFactory() {}
        protected void engineInit(java.security.KeyStore keystore) {}
        protected void engineInit(
                javax.net.ssl.ManagerFactoryParameters parameters) {}
        protected javax.net.ssl.TrustManager[] engineGetTrustManagers() {
            return new javax.net.ssl.TrustManager[] { new AcceptAllX509TrustManager() };
        }
```

Figura 27. AcceptAllTrustManagerFactory.java

```
protected static class AcceptAllX509TrustManager implements
        javax.net.ssl.X509TrustManager {
    public void checkClientTrusted(
            java.security.cert.X509Certificate[] certificateChain,
            String authType) throws java.security.cert.CertificateException {
        report("Client authtype=" + authType);
        for (java.security.cert.X509Certificate certificate : certificateChain) {
            report("Accepting:" + certificate);
        }
    }
    public void checkServerTrusted(
            java.security.cert.X509Certificate[] certificateChain,
            String authType) throws java.security.cert.CertificateException {
        report("Server authtype=" + authType);
        for (java.security.cert.X509Certificate certificate : certificateChain) {
            report("Accepting:" + certificate);
        }
    }
    public java.security.cert.X509Certificate[] getAcceptedIssuers() {
        return new java.security.cert.X509Certificate[0];
    }
private static void report(String string) {
       System.out.println(string);
  }
}
```

Figura 28. AcceptAllX509TrustManager.java

Configurazione del canale di telemetria JAAS

Configurare JAAS per autenticare il `Nome utente` inviato dal client.

L'amministratore WebSphere MQ configura quali canali MQTT richiedono l'autenticazione client utilizzando JAAS. Specificare il nome di una configurazione JAAS per ogni canale che deve eseguire l'autenticazione JAAS . I canali possono utilizzare la stessa configurazione JAAS oppure possono utilizzare configurazioni JAAS differenti. Le configurazioni sono definite in *WMQData directory*`\qmgrs\`*qMgrName*`\mqxr\jaas.config`.

Il file `jaas.config` è organizzato per nome di configurazione JAAS . Sotto ogni nome di configurazione c'è un elenco di configurazioni di login; consultare Figura 29 a pagina 139.

JAAS fornisce quattro moduli di login standard. I moduli di login NT e UNIX standard hanno un valore limitato.

Modulo JndiLogin
Esegue l'autenticazione rispetto a un servizio directory configurato in JNDI (Java Naming and Directory Interface).

Krb5LoginModule
Autentica utilizzando i protocolli Kerberos .

NTLoginModule
Autentica utilizzando le informazioni di sicurezza NT per l'utente corrente.

Modulo UnixLogin
Autentica utilizzando le informazioni di sicurezza UNIX per l'utente attuale.

Il problema nell'utilizzare `NTLoginModule` o `UnixLoginModule` è che il servizio di telemetria (MQXR) viene eseguito con l'identità `mqm` e non con l'identità del canale MQTT. `mqm` è l'identità passata a `NTLoginModule` o `UnixLoginModule` per l'autenticazione e non l'identità del client.

Per risolvere questo problema, scrivere il proprio modulo di login o utilizzare gli altri moduli di login standard. Un esempio `JAASLoginModule.java` viene fornito con WebSphere MQ Telemetry. È un'implementazione dell'interfaccia `javax.security.auth.spi.LoginModule` . Utilizzarla per sviluppare il proprio metodo di autenticazione.

Tutte le nuove classi LoginModule fornite devono trovarsi nel percorso classi del servizio di telemetria (MQXR). Non posizionare le classi nelle directory WebSphere MQ che si trovano nel percorso di classe. Creare le proprie directory e definire l'intero percorso classe per il servizio di telemetria (MQXR).

È possibile aumentare il percorso classe utilizzato dal servizio di telemetria (MQXR) impostando il percorso classe nel file `service.env` . `CLASSPATH` deve essere in maiuscolo e l'istruzione del percorso classe può contenere solo valori letterali. Non è possibile utilizzare variabili in CLASSPATH; ad esempio, `CLASSPATH=%CLASSPATH%` non è corretto. Il servizio di telemetria (MQXR) imposta il proprio percorso classi. Il CLASSPATH definito in `service.env` viene aggiunto ad esso.

Il servizio di telemetria (MQXR) fornisce due callback che restituiscono `Username` e la `Password` per un client connesso al canale MQTT. `Nome utente` e `Password` sono impostati nell'oggetto `MqttConnectOptions` . Consultare Figura 30 a pagina 139 per un esempio di come accedere a `Nome utente` e `Password`.

Esempi

Un esempio di file di configurazione JAAS con una configurazione denominata, `MQXRConfig`.

```
MQXRConfig {
  samples.JAASLoginModule required debug=true;
  //com.ibm.security.auth.module.NTLoginModule required;
  //com.ibm.security.auth.module.Krb5LoginModule required
  //                 principal=principal@your_realm
  //                 useDefaultCcache=TRUE
  //                 renewTGT=true;
  //com.sun.security.auth.module.NTLoginModule required;
  //com.sun.security.auth.module.UnixLoginModule required;
  //com.sun.security.auth.module.Krb5LoginModule required
  //                 useTicketCache="true"
  //                 ticketCache="${user.home}${/}tickets";
};
```

Figura 29. File `jaas.config` di esempio

Un esempio di modulo di accesso JAAS codificato per ricevere il `Nome utente` e `Password` forniti da un client MQTT.

```
  public boolean login()
      throws javax.security.auth.login.LoginException {
    javax.security.auth.callback.Callback[] callbacks =
      new javax.security.auth.callback.Callback[2];
    callbacks[0] = new javax.security.auth.callback.NameCallback("NameCallback");
    callbacks[1] = new javax.security.auth.callback.PasswordCallback(
        "PasswordCallback", false);
    try {
      callbackHandler.handle(callbacks);
      String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
          .getName();
      char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
          .getPassword();
      // Accept everything.
      if (true) {
        loggedIn = true;
      } else
        throw new javax.security.auth.login.FailedLoginException("Login failed");

      principal= new JAASPrincipal(username);

    } catch (java.io.IOException exception) {
      throw new javax.security.auth.login.LoginException(exception.toString());
    } catch (javax.security.auth.callback.UnsupportedCallbackException exception) {
      throw new javax.security.auth.login.LoginException(exception.toString());
    }

    return loggedIn;
  }
```

Figura 30. Metodo `JAASLoginModule.Login()` di esempio

IBM WebSphere MQ Daemon di telemetria per i concetti delle periferiche

Il daemon IBM WebSphere MQ Telemetry per dispositivi è un'applicazione client MQTT V3 avanzata. Utilizzarlo per memorizzare e inoltrare messaggi da altri client MQTT. Si connette a IBM WebSphere MQ come un client MQTT, ma è anche possibile connettervi altri client MQTT.

Il daemon è un broker di pubblicazione / sottoscrizione. I client MQTT V3 si collegano ad esso per pubblicare e sottoscrivere argomenti, utilizzando stringhe di argomenti per la pubblicazione e filtri argomenti per la sottoscrizione. La stringa di argomenti è gerarchica, con livelli di argomenti divisi per /. I filtri argomento sono stringhe di argomento che possono includere caratteri jolly + di livello singolo e un carattere jolly # multilivello come ultima parte della stringa di argomento.

Nota: I caratteri jolly nel daemon seguono le regole più restrittive di WebSphere Message Broker, v6. IBM WebSphere MQ è diverso. Supporta più caratteri jolly multilivello; i caratteri jolly possono indicare qualsiasi numero di livelli della gerarchia, in qualsiasi punto della stringa di argomenti.

Più client MQTT v3 si collegano al daemon utilizzando una porta listener. La porta listener predefinita è modificabile. È possibile definire più porte listener e assegnare diversi spazi dei nomi ad esse, consultare “Daemon WebSphere MQ Telemetry per le porte listener dei dispositivi” a pagina 148. Il daemon è esso stesso un client MQTT v3 . Configurare una connessione bridge daemon per collegare il daemon alla porta listener di un altro daemon o a un servizio WebSphere MQ Telemetry (MQXR).

È possibile configurare più bridge per il daemon WebSphere MQ Telemetry per i dispositivi. Utilizzare i bridge per collegare insieme una rete di daemon che possono scambiare pubblicazioni.

Ogni bridge può pubblicare e sottoscrivere argomenti sul proprio daemon locale. Può anche pubblicare e sottoscrivere argomenti in un altro daemon, in un broker di pubblicazione / sottoscrizione WebSphere MQ o in qualsiasi altro broker MQTT v3 a cui è connesso. Utilizzando un filtraggio argomenti, è possibile selezionare le pubblicazioni da propagare da un broker all'altro. È possibile propagare le pubblicazioni in entrambe le direzioni. È possibile propagare le pubblicazioni dal daemon locale a ciascuno dei relativi broker remoti collegati o da uno qualsiasi dei broker collegati al daemon locale; consultare “IBM WebSphere MQ Daemon di telemetria per bridge di dispositivi” a pagina 140.

IBM WebSphere MQ Daemon di telemetria per bridge di dispositivi

Un daemon IBM WebSphere MQ Telemetry per il bridge dei dispositivi collega due broker di pubblicazione / sottoscrizione utilizzando il protocollo MQTT v3 . Il bridge propaga le pubblicazioni da un broker all'altro, in entrambe le direzioni. Da un lato è presente un daemon WebSphere MQ Telemetry per la connessione bridge dei dispositivi e dall'altro potrebbe essere un gestore code o un altro daemon. Un gestore code è connesso alla connessione bridge utilizzando un canale di telemetria. Un daemon è connesso alla connessione bridge utilizzando un listener daemon.

Il daemon IBM WebSphere MQ Telemetry per le periferiche supporta una o più connessioni simultanee ad altri broker. Le connessioni dal daemon sono chiamate bridge e sono definite da voci di connessione nel file di configurazione del daemon. Le connessioni a IBM WebSphere MQ vengono effettuate utilizzando i canali di telemetria IBM WebSphere MQ , come mostrato nella seguente figura:

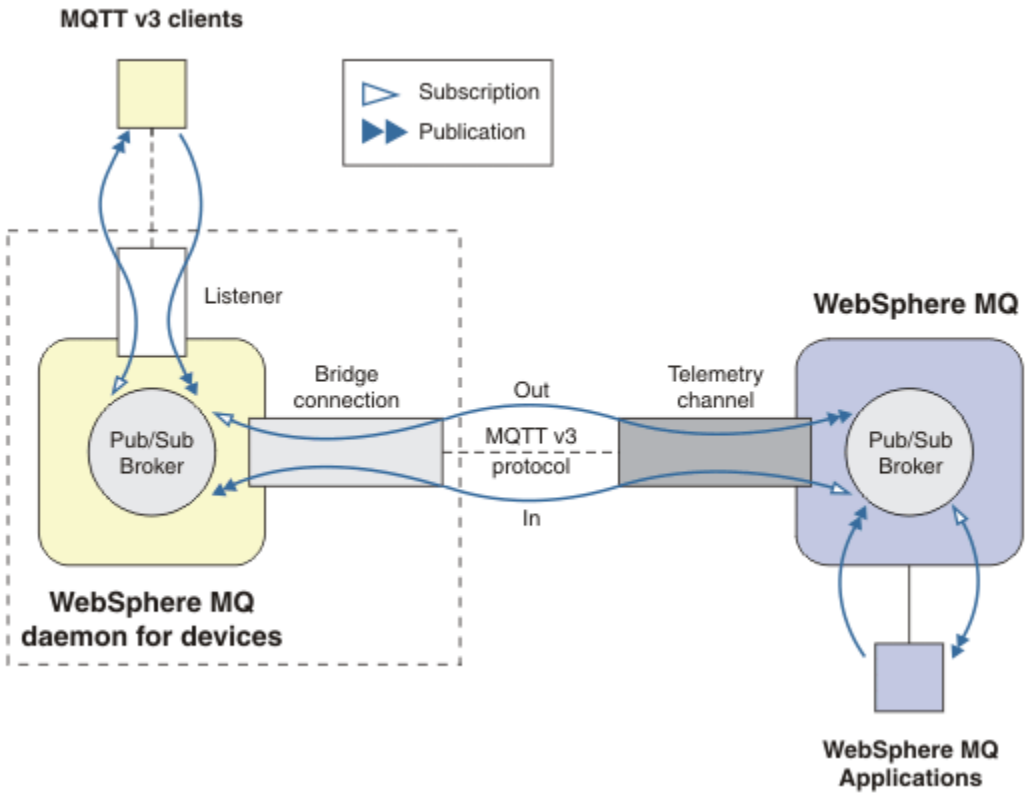


Figura 31. Connessione IBM WebSphere MQ Telemetry daemon for devices a IBM WebSphere MQ

Un bridge collega il daemon a un altro broker come client MQTT v3 . I parametri bridge riflettono gli attributi di un client MQTT v3 .

Un bridge è più di una connessione. Agisce come agente di pubblicazione e sottoscrizione situato tra due broker di pubblicazione / sottoscrizione. Il broker locale è il daemon IBM WebSphere MQ Telemetry per i dispositivi e il broker remoto è qualsiasi broker di pubblicazione / sottoscrizione che supporta il protocollo MQTT v3 . Di solito, il broker remoto è un altro deamon o IBM WebSphere MQ.

Il lavoro del bridge è quello di propagare le pubblicazioni tra i due broker. Il ponte è bidirezionale. Propaga le pubblicazioni in entrambe le direzioni. Figura 31 a pagina 141 illustra il modo in cui il bridge collega il daemon IBM WebSphere MQ Telemetry per i dispositivi a IBM WebSphere MQ. “Impostazioni dell'argomento di esempio per il ponte” a pagina 142 utilizza esempi per illustrare come utilizzare il parametro dell'argomento per configurare il bridge.

Le frecce In e Out in Figura 31 a pagina 141 indicano la bidirezionalità del bridge. Ad un'estremità della freccia, viene creata una sottoscrizione. Le pubblicazioni che corrispondono alla sottoscrizione vengono pubblicate sul broker all'estremità opposta della freccia. La freccia viene etichettata in base al flusso di pubblicazioni. Le pubblicazioni vengono portate In al daemon e Out dal daemon. L'importanza delle etichette è che vengono utilizzate nella sintassi del comando. Tenere presente che In e Out fanno riferimento al punto in cui vengono inviate le pubblicazioni e non al punto in cui viene inviata la sottoscrizione.

Altri client, applicazioni o broker potrebbero essere connessi a IBM WebSphere MQ o al daemon WebSphere MQ Telemetry per dispositivi. Pubblicano e sottoscrivono argomenti nel broker a cui sono connessi. Se il broker è IBM WebSphere MQ, gli argomenti potrebbero essere in cluster o distribuiti e non esplicitamente definiti nel gestore code locale.

Usi dei ponti

Collegare i daemon utilizzando le connessioni bridge e i listener. Connettere i daemon e i gestori code utilizzando le connessioni bridge e i canali di telemetria. Quando si connettono più broker insieme, è possibile creare dei loop. Attenzione: le pubblicazioni potrebbero circolare all'infinito intorno a un loop di broker, non rilevati.

Alcuni dei motivi per utilizzare i daemon collegati a IBM WebSphere MQ sono i seguenti:

Ridurre il numero di connessioni client MQTT a WebSphere MQ
Utilizzando una gerarchia di daemon, è possibile connettere molti client a WebSphere MQ; un numero di client superiore al numero di connessioni di un singolo gestore code alla volta.

Memorizza e inoltra i messaggi tra client MQTT e WebSphere MQ
È possibile utilizzare l'archiviazione e l'inoltro per evitare di mantenere connessioni continue tra client e IBM WebSphere MQ, se i client non dispongono di una propria memoria. È possibile utilizzare più tipi di connessione tra il client MQTT e WebSphere MQ; consultare Scenari e concetti di telemetria per il monitoraggio e il controllo.

Filtrare le pubblicazioni scambiate tra client MQTT e WebSphere MQ
In genere, le pubblicazioni si dividono in messaggi elaborati localmente e messaggi che coinvolgono altre applicazioni. Le pubblicazioni locali possono includere i flussi di controllo tra sensori e attuatori e le pubblicazioni remote includono richieste di letture, stato e comandi di configurazione.

Modificare gli spazi argomenti delle pubblicazioni
Evitare che le stringhe di argomenti dai client collegati a porte listener differenti entrino in conflitto tra loro. L'esempio utilizza il daemon per etichettare le letture contatore provenienti da edifici differenti; consultare Separazione degli spazi argomento di gruppi di client diversi.

Impostazioni dell'argomento di esempio per il ponte

Pubblica tutto nel broker remoto - utilizzando i valori predefiniti

La direzione predefinita viene denominata oute il bridge pubblica gli argomenti sul broker remoto. Il parametro topic controlla quali argomenti vengono propagati utilizzando filtri argomento.

Il bridge utilizza il parametro topic in Figura 32 a pagina 142 per sottoscrivere tutti gli elementi pubblicati sul daemon locale dai client MQTT o da altri broker. Il bridge pubblica gli argomenti sul broker remoto connesso dal bridge.

```
connection Daemon1
topic #
```

Figura 32. Pubblica tutto sul broker remoto

Pubblica tutto nel broker remoto - esplicito

L'impostazione topic nel seguente frammento di codice fornisce lo stesso risultato dell'utilizzo dei valori predefiniti. L'unica differenza è che il parametro **direction** è esplicito. Utilizzare la direzione `out` per sottoscrivere il broker locale, il daemon e pubblicare sul broker remoto. Le pubblicazioni create sul daemon locale a cui il bridge ha sottoscritto, vengono pubblicate sul broker remoto.

```
connection Daemon1
topic # out
```

Figura 33. Pubblica tutto nel broker remoto - esplicito

Pubblica tutto nel broker locale

Invece di utilizzare la direzione `out`, è possibile impostare la direzione opposta, `in`. Il seguente frammento di codice configura il bridge per sottoscrivere tutti gli elementi pubblicati sul broker remoto connesso dal bridge. Il bridge pubblica gli argomenti nel broker locale, il daemon.

```
connection Daemon1
topic # in
```

Figura 34. Pubblica tutto nel broker locale

Pubblicare tutto dall'argomento di esportazione nel Broker locale all'argomento di importazione nel Broker remoto

Utilizzare due parametri argomento aggiuntivi, **`local_prefix`** e **`remote_prefix`**, per modificare il filtro argomento, `#` negli esempi precedenti. Un parametro viene utilizzato per la modifica del filtro argomento utilizzato nella sottoscrizione e l'altro parametro viene utilizzato per modificare l'argomento in cui viene pubblicata la pubblicazione. L'effetto è quello di sostituire l'inizio della stringa di argomenti utilizzata in un broker con un'altra stringa di argomenti sull'altro broker.

In base alla direzione del comando dell'argomento, il significato di **`local_prefix`** e di **`remote_prefix`** viene invertire. Se la direzione è `out`, il valore predefinito, **`local_prefix`** viene utilizzato come parte della sottoscrizione argomento e **`remote_prefix`** sostituisce la parte **`local_prefix`** della stringa argomento nella pubblicazione remota. Se la direzione è `in`, **`remote_prefix`** diventa parte della sottoscrizione remota e **`local_prefix`** sostituisce la parte **`remote_prefix`** della stringa di argomenti.

La prima parte di una stringa di argomenti viene spesso considerata come una definizione di uno spazio argomenti. Utilizzare i parametri aggiuntivi per modificare lo spazio argomento in cui viene pubblicato un argomento. È possibile eseguire questa operazione per evitare che l'argomento venga propagato in conflitto con un altro argomento sul broker di destinazione o per rimuovere una stringa di argomento del punto di montaggio.

Ad esempio, nel frammento di codice riportato di seguito, tutte le pubblicazioni nella stringa di argomenti `export/#` sul daemon vengono ripubblicate in `import/#` sul broker remoto.

```
topic # out export/ import/
```

Figura 35. Pubblicare tutto dall'argomento di esportazione nel Broker locale all'argomento di importazione nel Broker remoto

Pubblicare tutto nell'argomento di importazione nel broker locale dall'argomento di esportazione nel broker remoto

Il seguente frammento di codice mostra la configurazione invertita; il bridge sottoscrive tutto ciò che è pubblicato con la stringa di argomenti `export/#` nel broker remoto e lo pubblica in `import/#` nel broker locale.

```
connection Daemon1
topic # in import/ export/
```

Figura 36. Pubblicare tutto nell'argomento di importazione nel broker locale dall'argomento di esportazione nel broker remoto

Pubblicare tutto dal punto di montaggio 1884/ al broker remoto con le stringhe di argomenti originali

Nel seguente frammento di codice, il bridge sottoscrive tutto ciò che viene pubblicato dai client connessi al punto di montaggio 1884/ sul daemon locale. Il bridge pubblica tutti gli elementi

pubblicati sul punto di montaggio sul broker remoto. La stringa del punto di montaggio 1884/ viene eliminata dagli argomenti pubblicati sul broker remoto. *local_prefix* è uguale alla stringa del punto di montaggio 1884/e *remote_prefix* è una stringa vuota.

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

Figura 37. Pubblicare tutto dal punto di montaggio 1884/ al broker remoto con le stringhe di argomenti originali.

Separazione degli spazi argomento di client differenti connessi a daemon differenti

Un'applicazione viene scritta per i contatori di potenza elettrica per pubblicare le letture contatore per un edificio. Le letture vengono pubblicate utilizzando i client MQTT per un daemon ospitato nello stesso edificio. L'argomento selezionato per le pubblicazioni è `power`. La stessa applicazione viene distribuita a un numero di edifici in un complesso. Per il monitoraggio del sito e l'archiviazione dei dati, le letture da tutti gli edifici vengono aggregate utilizzando connessioni bridge. Le connessioni collegano i daemon di creazione a WebSphere MQ in un'ubicazione centrale.

Le applicazioni client in ogni edificio sono identiche, ma i dati devono essere differenziati in base alla creazione. Ogni lettura ha un argomento `power` e deve essere preceduto dal numero di edificio per distinguerlo. Il ponte dal primo edificio nel complesso utilizza il prefisso `meters/building01/`, dal secondo il prefisso è `meters/building02/`. Le letture degli altri edifici seguono lo stesso schema. WebSphere MQ riceve le letture con argomenti come `meters/building01/power`.

L'esempio è inventato; in pratica lo spazio argomento su cui l'applicazione pubblica è probabilmente configurabile.

Il file di configurazione per ogni demone ha un'istruzione di argomento che segue il pattern nel seguente frammento di codice:

```
connection Daemon1
topic power out "" meters/building01/
```

Figura 38. Separare gli spazi argomenti dei client connessi a diversi daemon

Specificare una stringa vuota come segnaposto per il parametro local_prefix non utilizzato.

Separare gli spazi argomenti dei client connessi allo stesso daemon

Supponiamo che un singolo daemon venga utilizzato per collegare tutti i misuratori di potenza. Supponendo che nell'applicazione sia possibile configurare la connessione a porte differenti, è possibile distinguere gli edifici collegando i contatori da edifici differenti a porte listener differenti, come nel seguente frammenti di codice. Di nuovo, l'esempio è inventato; illustra come potrebbero essere utilizzati i punti di montaggio.

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

Figura 39. Separare gli spazi argomenti dei client connessi allo stesso daemon

Riassocia diversi argomenti per le pubblicazioni che fluiscono in entrambe le direzioni

Nella configurazione contenuta nel seguente frammento di codice, il bridge sottoscrive il singolo argomento b sul broker remoto e inoltra le pubblicazioni relative a b al daemon locale, modificando l'argomento in a. Il bridge inoltre sottoscrive il singolo argomento x nel broker locale e inoltra le pubblicazioni relative a x al broker remoto, modificando l'argomento in y.

```
connection Daemon1
topic "" in a b
topic "" out x y
```

Figura 40. Riassocia diversi argomenti per le pubblicazioni che fluiscono in entrambe le direzioni

Un punto importante su questo esempio è che diversi argomenti sono sottoscritti e pubblicati in entrambi i broker. Gli spazi argomenti in entrambi i broker sono disgiunti.

Riassocia gli stessi argomenti per le pubblicazioni che scorrono in entrambe le direzioni (loop)

A differenza dell'esempio precedente, la configurazione in Figura 41 a pagina 145, in genere, risulta in un loop. Nell'istruzione dell'argomento `topic "" in a b`, il bridge sottoscrive b in remoto e pubblica in a localmente. Nell'altra istruzione dell'argomento, il bridge sottoscrive a localmente e pubblica in b in remoto. La stessa configurazione può essere scritta come mostrato in Figura 42 a pagina 145.

Il risultato generale è che se un client pubblica in b in remoto, la pubblicazione viene trasferita al daemon locale come pubblicazione sull'argomento a. Tuttavia, quando viene pubblicato dal bridge al daemon locale sull'argomento a, la pubblicazione corrisponde alla sottoscrizione effettuata dal bridge all'argomento locale a. La sottoscrizione è `topic "" out a b`. Di conseguenza, la pubblicazione viene trasferita nuovamente al broker remoto come pubblicazione sull'argomento b. Il bridge è ora sottoscritto all'argomento remoto be il ciclo inizia nuovamente.

Alcuni broker implementano il rilevamento del loop per evitare che si verifichi il loop. Ma il meccanismo di rilevamento del loop deve funzionare quando diversi tipi di broker sono collegati tra loro. Il rilevamento dei loop non funziona se WebSphere MQ è collegato al daemon WebSphere MQ Telemetry per i dispositivi. Funziona se due daemon IBM WebSphere MQ Telemetry per dispositivi sono collegati tra loro. Per impostazione predefinita, il rilevamento loop è attivato; consultare try_private.

```
connection Daemon1
topic "" in a b
topic "" out a b
```

*Figura 41. **!**Riassocia gli stessi argomenti per le pubblicazioni che fluiscono in entrambe le direzioni*

```
connection Daemon1
topic "" both a b
```

*Figura 42. **!**Riassociare gli stessi argomenti per le pubblicazioni che fluiscono in entrambe le direzioni, utilizzando `both`.*

La configurazione in Figura 40 a pagina 145 è uguale a Figura 41 a pagina 145.

Disponibilità delle connessioni bridge IBM WebSphere MQ Telemetry daemon for devices

Configurare più indirizzi di connessione bridge IBM WebSphere MQ Telemetry daemon for devices per connettersi al primo broker remoto disponibile. Se il broker è un gestore code a più istanze, fornire

entrambi gli indirizzi TCP/IP. Configurare una connessione primaria per connettersi, o riconnettersi, al server primario, quando è disponibile.

Il parametro del bridge di connessione, indirizzi, è un elenco di indirizzi socket TCP/IP. Il bridge tenta di connettersi a ciascun indirizzo a turno, fino a quando non effettua una connessione corretta. I parametri di connessione round_robin e start_type controllano il modo in cui gli indirizzi vengono utilizzati una volta stabilita una corretta connessione.

Se start_type è auto, manualo lazy, se la connessione ha esito negativo, il bridge tenta di riconnettersi. Usa ogni indirizzo a turno, con un ritardo di circa 20 secondi tra ogni tentativo di connessione. Se `tipo_avvio` è una volta, se la connessione ha esito negativo, il bridge non tenta di riconnettersi automaticamente.

Se round_robin è true, i tentativi di connessione bridge iniziano dal primo indirizzo nell'elenco e tentano di volta in volta ogni indirizzo nell'elenco. Inizia di nuovo al primo indirizzo, quando l'elenco è esaurito. Se c'è solo un indirizzo nell'elenco, lo tenta di nuovo ogni 20 secondi.

Se round_robin è false, viene data la preferenza al primo indirizzo dell'elenco, denominato server primario. Se il primo tentativo di connessione al server primario non riesce, il bridge continua a provare a riconnettersi al server primario in background. Allo stesso tempo, il ponte tenta di connettersi utilizzando gli altri indirizzi nell'elenco. Quando i tentativi di connessione in background al server principale hanno esito positivo, il bridge si disconnette dalla connessione corrente e passa alla connessione del server principale.

Se una connessione viene disconnessa volontariamente, ad esempio immettendo un comando **`connection_stop`** , se la connessione viene riavviata, tenta di utilizzare nuovamente lo stesso indirizzo. Se la connessione è stata disconnessa a causa di un errore di connessione o se il broker remoto ha rilasciato la connessione, il bridge attende 20 secondi. Tenta quindi di collegarsi all'indirizzo successivo nell'elenco, o allo stesso indirizzo, se c'è un solo indirizzo nell'elenco.

Connessione a un gestore code a più istanze

In una configurazione del gestore code a più istanze, il gestore code viene eseguito su due server differenti con indirizzi IP diversi. In genere, i canali di telemetria vengono configurati senza un indirizzo IP specifico. Sono configurati solo con un numero di porta. Quando il canale di telemetria viene avviato, per impostazione predefinita seleziona il primo indirizzo di rete disponibile sul server locale.

Configurare il parametro indirizzi della connessione bridge con i due indirizzi IP utilizzati dal gestore code. Impostare round_robin su true.

Se l'istanza del gestore code attiva ha esito negativo, il gestore code passa all'istanza in standby. Il daemon rileva che la connessione all'istanza attiva è stata interrotta e tenta di riconnettersi all'istanza standby. Utilizza l'altro indirizzo IP nell'elenco di indirizzi configurati per la connessione bridge.

Il gestore code a cui si connette il bridge è ancora lo stesso gestore code. Il gestore code ripristina il proprio stato. Se cleansession è impostato su false, la sessione di connessione bridge viene ripristinata allo stesso stato di prima del failover. La connessione riprende dopo un ritardo. I messaggi con "almeno una volta" o "al massimo una volta" di QoS (quality of service) non vengono persi e le sottoscrizioni continuano a funzionare.

Il tempo di riconnessione dipende dal numero di canali e client che si riavviano all'avvio dell'istanza in standby e dal numero di messaggi in corso. La connessione bridge potrebbe tentare di riconnettersi a entrambi gli indirizzi IP un numero di volte prima che la connessione venga ristabilita.

Non configurare un canale di telemetria del gestore code a più istanze con un indirizzo IP specifico. L'indirizzo IP è valido solo su un server.

Se si utilizza una soluzione di alta disponibilità alternativa, che gestisce l'indirizzo IP, potrebbe essere corretto configurare un canale di telemetria con un indirizzo IP specifico.

pulisci sessione

Una connessione bridge è una sessione client MQTT v3 . È possibile controllare se una connessione avvia una nuova sessione o se ripristina una sessione esistente. Se ripristina una sessione esistente, la connessione bridge conserva le sottoscrizioni e le pubblicazioni conservate della sessione precedente.

Non impostare cleansession su false se address elenca più indirizzi IP e gli indirizzi IP si connettono a canali di telemetria ospitati da gestori code differenti o a daemon di telemetria diversi. Lo stato della sessione non viene trasferito tra gestori code o daemon. Il tentativo di riavviare una sessione esistente su un altro gestore code o daemon comporta l'avvio di una nuova sessione. I messaggi in dubbio vengono persi e le sottoscrizioni potrebbero non funzionare come previsto.

notifiche

Un'applicazione può tenere traccia se la connessione bridge è in esecuzione utilizzando le notifiche. Una notifica è una pubblicazione con il valore 1, connesso o 0, disconnesso. Viene pubblicato in *topicString* definito dal parametro `notification_topic` . Il valore predefinito di *topicString* Š $SYS/`broker`/ `connection`/ *clientIdentifier*/`state`. Il *topicString* predefinito contiene il prefisso $SYS. Sottoscrivi gli argomenti che iniziano con $SYS definendo un filtro argomenti che inizia con $SYS. Il filtro argomenti #, sottoscrive tutto, non sottoscrive gli argomenti che iniziano con $SYS sul daemon. Considerare $SYS come la definizione di uno spazio argomento di sistema speciale distinto dallo spazio argomento dell'applicazione.

Notifiche abilita IBM WebSphere MQ Telemetry daemon for devices a notificare ai client MQTT quando un bridge è connesso o disconnesso.

intervallo_keepaliv

Il parametro di connessione bridge keepalive_interval imposta l'intervallo tra il bridge che invia un ping TCP/IP al server remoto. L'intervallo predefinito è di 60 secondi. Il ping impedisce la chiusura della sessione TCP/IP da parte del server remoto o di un firewall, che rileva un periodo di inattività sulla connessione.

clientID

Una connessione bridge è una sessione client MQTT v3 e ha un `clientIdentifier` impostato dal parametro di connessione bridge clientid. Se si intende ripristinare una sessione precedente impostando il parametro cleansession su false, il `clientIdentifier` utilizzato in ciascuna sessione deve essere lo stesso. Il valore predefinito di clientid è *hostname.connectionName*, che rimane lo stesso.

Installazione, verifica, configurazione e controllo del daemon WebSphere MQ Telemetry per i dispositivi

L'installazione, la configurazione e il controllo del daemon sono basati su file.

Installare il daemon copiando il SDK (Software Development Kit) sul dispositivo su cui verrà eseguito il daemon .

Ad esempio, eseguire il programma di utilità del client MQTT e connettersi al daemon WebSphere MQ Telemetry per i dispositivi come broker di pubblicazione / sottoscrizione; consultare Pubblicazione di un messaggio su uno specifico client MQTT v3 .

Configurare il daemon creando un file di configurazione; consultare WebSphere MQ Telemetry daemon for devices configuration file.

Controllare un daemon in esecuzione creando comandi nel file, `amqtdd.upd`. Ogni 5 secondi il daemon legge il file, esegue i comandi ed elimina il file; consultare il daemon WebSphere MQ Telemetry per il file di comando dei dispositivi.

Daemon WebSphere MQ Telemetry per le porte listener dei dispositivi

Connettere i client MQTT V3 al daemon WebSphere MQ Telemetry per i dispositivi che utilizzano le porte listener. È possibile qualificare una porta listener con un punto di montaggio e un numero massimo di connessioni.

Una porta del listener deve corrispondere al numero di porta specificato sul metodo `connect(serverURI)` del client MQTT di un client che si connette a questa porta. Il valore predefinito è 1883sia sul client che sul daemon.

È possibile modificare la porta predefinita per il daemon impostando la definizione globale `port` nel file di configurazione del daemon. È possibile impostare porte specifiche aggiungendo una definizione `listener` al file di configurazione del daemon.

Per ogni porta listener, diversa da quella predefinita, è possibile specificare un punto di montaggio per isolare i client. I client connessi a una porta con un punto di montaggio sono isolati dagli altri client; consultare "Daemon WebSphere MQ Telemetry per i punti di montaggio dei dispositivi" a pagina 148.

È possibile limitare il numero di client che possono connettersi a qualsiasi porta. Impostare la definizione globale `max_connections` per limitare le connessioni alla porta predefinita o qualificare ciascuna porta listener con `max_connections`.

Esempio

Un esempio di un file di configurazione che modifica la porta predefinita da 1883 a 1880e limita la connessione alla porta 1880 a 10000. Le connessioni alla porta 1884 si limitano a 1000. I client collegati alla porta 1884 sono isolati dai client collegati ad altre porte.

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

Daemon WebSphere MQ Telemetry per i punti di montaggio dei dispositivi

È possibile associare un punto di montaggio a una porta listener utilizzata dai client MQTT per connettersi a un daemon WebSphere MQ Telemetry per dispositivi. Un punto di montaggio isola le pubblicazioni e le sottoscrizioni scambiate dai clienti MQTT utilizzando una porta listener dai client MQTT connessi a una porta listener differente.

I client collegati ad una porta listener con un punto di montaggio non possono mai scambiare direttamente argomenti con i client collegati ad altre porte listener. I client collegati a una porta listener senza un punto di montaggio possono pubblicare o sottoscrivere argomenti di qualsiasi client. I client non sanno se sono collegati tramite un punto di montaggio o meno; non fa alcuna differenza per le stringhe di argomenti create dai client.

Un punto di montaggio è una stringa di testo che ha come prefisso la stringa di argomenti di pubblicazioni e sottoscrizioni. Ha come prefisso tutte le stringhe di argomento create dai client collegati alla porta del listener con un punto di montaggio. La stringa di testo viene eliminata da tutte le stringhe di argomenti inviate a client collegati alla porta listener.

Se una porta listener non ha un punto di montaggio, le stringhe di argomenti delle pubblicazioni e delle sottoscrizioni create e ricevute dai client collegati alla porta non vengono modificate.

Creare stringhe di punto di montaggio con un /finale. In questo modo il punto di montaggio è l'argomento principale della struttura ad albero degli argomenti per il punto di montaggio.

Esempio

Un file di configurazione contiene le seguenti porte listener:

```
listener 1883
mount_point 1883/
```

```
listener 1884 127.0.0.1
mount_point 1884/
listener 1885
```

Un client, collegato alla porta 1883, crea una sottoscrizione a `MyTopic`. Il daemon registra la sottoscrizione come `1883/MyTopic`. Un altro client collegato alla porta 1883 pubblica un messaggio sull'argomento, `MyTopic`. Il daemon modifica la stringa argomento in `1883/MyTopic` e ricerca le sottoscrizioni corrispondenti. Il sottoscrittore sulla porta 1883 riceve la pubblicazione con la stringa di argomenti originale `MyTopic`. Il daemon ha rimosso il prefisso del punto di montaggio dalla stringa argomento.

Un altro client, collegato alla porta 1884, pubblica anche l'argomento `MyTopic`. Questa volta il daemon registra l'argomento come `1884/MyTopic`. Il sottoscrittore (subscriber) sulla porta 1883 non riceve la pubblicazione perché il punto di montaggio differente risulta in una sottoscrizione con una stringa di argomenti differente.

Un client, collegato alla porta 1885, pubblica sull'argomento, `1883/MyTopic`. Il daemon non modifica la stringa argomento. Il sottoscrittore sulla porta 1883 riceve la pubblicazione in `MyTopic`.

Daemon WebSphere MQ Telemetry per la qualità del servizio dei dispositivi, le sottoscrizioni durevoli e le pubblicazioni conservate

Le impostazioni QoS (Quality of Service) si applicano solo a un daemon in esecuzione. Se un daemon si arresta, in modo controllato o a causa di un errore, lo stato dei messaggi in corso viene perso. La consegna di un messaggio almeno una volta, o al massimo una volta, non può essere garantita se il daemon si arresta. Il daemon WebSphere MQ Telemetry per dispositivi supporta la persistenza limitata. Impostare il parametro di configurazione **`retained_persistence`** per salvare le pubblicazioni e le sottoscrizioni conservate quando il daemon è arrestato.

A differenza di WebSphere MQ, il daemon WebSphere MQ Telemetry per dispositivi non registra i dati persistenti. Lo stato della sessione, lo stato del messaggio e le pubblicazioni conservate non vengono salvate in modo transazionale. Per impostazione predefinita, il daemon elimina tutti i dati quando si arresta. È possibile impostare un'opzione per controllare periodicamente le sottoscrizioni e le pubblicazioni conservate. Lo stato del messaggio viene sempre perso quando il daemon viene arrestato. Tutte le pubblicazioni non conservate vengono perse.

Impostare l'opzione di configurazione del daemon, `Retained_persistenza` su `true`, per salvare periodicamente le pubblicazioni conservate in un file. Quando il daemon viene riavviato, le pubblicazioni conservate che sono state salvate automaticamente per l'ultima volta vengono ripristinate. Per impostazione predefinita, i messaggi conservati creati dai client non vengono ripristinati al riavvio del daemon.

Impostare l'opzione di configurazione daemon, `Retained_persistence` su `true`, per salvare le sottoscrizioni create periodicamente in una sessione persistente in un file. Se `Retained_persistence` è impostato su `true`, le sottoscrizioni che i client creano in una sessione con `CleanSession` impostato su `false`, una "sessione persistente", vengono ripristinate. Il daemon ripristina le sottoscrizioni al riavvio, che iniziano a ricevere le pubblicazioni. Il client riceve le pubblicazioni quando viene riavviato con `CleanSession` in `false`. Per impostazione predefinita, lo stato della sessione client non viene salvato quando un daemon viene arrestato e quindi le sottoscrizioni non vengono ripristinate, anche se il client imposta `CleanSession` su `false`.

`Retained_persistence` è un meccanismo di salvataggio automatico. Potrebbe non salvare le pubblicazioni o le sottoscrizioni conservate più recenti. È possibile modificare la frequenza con cui vengono salvate le pubblicazioni e le sottoscrizioni conservate. Impostare l'intervallo tra i salvataggi o il numero di modifiche tra salvataggi, utilizzando le opzioni di configurazione `autosave_on_changes` e `autosave_interval`.

Configurazione di esempio per l'impostazione della persistenza

```
# Sample configuration
# Daemon listens on port 1882 with persistence in /tmp
# Autosave every minute
```

```
port 1882
persistence_location /tmp/
retained_persistence true
autosave_on_changes false
autosave_interval 60
```

Daemon WebSphere MQ Telemetry per la sicurezza dei dispositivi

Il daemon WebSphere MQ Telemetry per i dispositivi può autenticare i client che si collegano ad esso, utilizzare credenziali per connettersi ad altri broker e controllare l'accesso agli argomenti. La sicurezza fornita dal daemon è limitata dall'utilizzo del client C WebSphere MQ Telemetry , che non fornisce il supporto SSL. Di conseguenza, le connessioni al e dal daemon non vengono codificate e non possono essere autenticate utilizzando i certificati.

Per impostazione predefinita, non è attivata alcuna sicurezza.

Autenticazione dei client

I clienti MQTT possono impostare un nome utente e una password utilizzando i metodi `MqttConnectOptions.setUserName` e `MqttConnectOptions.setPassword`.

Autenticare un cliente che si connette al daemon controllando il nome utente e la password forniti da un cliente rispetto alle voci nel file delle password. Per abilitare l'autenticazione, creare un file di password e impostare il parametro `password_file` nel file di configurazione daemon; consultare password_file.

Impostare il parametro `allow_anonymous` nel file di configurazione daemon per consentire ai client che si collegano senza nomi utente o password di connettersi a un daemon che sta controllando l'autenticazione; consultare allow_anonymous. Se un client non fornisce un nome utente o una password, viene sempre controllato rispetto al file di password, se il parametro `password_file` è impostato.

Impostare il parametro `clientid_prefixes` nel file di configurazione daemon per limitare le connessioni a client specifici. I client devono avere `clientIdentifiers` che iniziano con uno dei prefissi elencati nel parametro `clientid_prefixes` ; consultare clientid_prefixes.

Sicurezza della connessione bridge

Ogni daemon WebSphere MQ Telemetry per la connessione bridge dei dispositivi è un client MQTT V3 . È possibile impostare il nome utente e la password per ogni connessione bridge come parametro di connessione bridge nel file di configurazione daemon; consultare username e password. Un bridge può quindi autenticarsi su un broker.

Controllo accessi degli argomenti

Se i client sono in fase di autenticazione, il daemon può anche fornire l'accesso di controllo agli argomenti per ciascun utente. Il daemon concede il controllo dell'accesso in base alla corrispondenza dell'argomento su cui un client sta effettuando la pubblicazione o la sottoscrizione con una stringa di argomenti di accesso nel file di controllo dell'accesso; consultare acl_file.

L'elenco di controllo accessi è diviso in due parti. La prima parte controlla l'accesso per tutti i client, inclusi quelli anonimi. La seconda parte contiene una sezione per qualsiasi utente nel file di password. Elenca il controllo accessi specifico per ciascun utente.

Esempio

I parametri di sicurezza sono riportati nel seguente esempio.

```
acl_file c:\WMQTDaemon\config\acl.txt
password_file c:\WMQTDaemon\config\passwords.txt
allow_anonymous true
connection Daemon1
username daemon1
password deamonpassword
```

Figura 43. File di configurazione daemon

```
Fred:Fredpassword
Barney:Barneypassword
```

Figura 44. File di password, passwords.txt

```
topic home/public/#
topic read meters/#
user Fred
topic write meters/fred
topic home/fred/#
user Barney
topic write meters/barney
topic home/barney/#
```

Figura 45. File di controllo accessi, acl.txt

Gestione multicast

Utilizzare queste informazioni per informazioni sulle attività di gestione di WebSphere MQ Multicast, come la riduzione della dimensione dei messaggi multicast e l'abilitazione della conversione dei dati.

Introduzione a multicast

Utilizzare queste informazioni per iniziare a utilizzare gli argomenti WebSphere MQ Multicast e gli oggetti delle informazioni di comunicazione.

Informazioni su questa attività

WebSphere MQ La messaggistica multicast utilizza la rete per consegnare i messaggi associando gli argomenti agli indirizzi dei gruppi. Le seguenti attività sono un modo rapido per verificare se la porta e l'indirizzo IP richiesti sono configurati correttamente per la messaggistica multicast.

Creazione di un oggetto COMMINFO per multicast

L'oggetto informazioni di comunicazione (COMMINFO) contiene gli attributi associati alla trasmissione multicast. Per ulteriori informazioni sui parametri oggetto COMMINFO, consultare DEFINE COMMINFO.

Utilizzare il seguente esempio di riga comandi per definire un oggetto COMMINFO per multicast:

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

dove *MC1* è il nome dell'oggetto COMMINFO, *indirizzo gruppo* è l'indirizzo IP o il nome DNS multicast del gruppo e *numero porta* è la porta su cui trasmettere (il valore predefinito è 1414).

Viene creato un nuovo oggetto COMMINFO denominato *MC1* ; questo nome è il nome che è necessario specificare quando si definisce un oggetto TOPIC nell'esempio successivo.

Creazione di un oggetto TOPIC per multicast

Un argomento è l'oggetto delle informazioni pubblicate in un messaggio di pubblicazione / sottoscrizione e un argomento viene definito creando un oggetto TOPIC. Gli oggetti TOPIC hanno due parametri che definiscono se possono essere utilizzati o meno con multicast. Questi parametri sono **COMMINFO** e **MCAST**.

- **COMMINFO** Questo parametro specifica il nome dell'oggetto delle informazioni di comunicazione multicast. Per ulteriori informazioni sui parametri oggetto COMMINFO, consultare DEFINE COMMINFO.
- **MCAST** Questo parametro specifica se il multicast è consentito in questa posizione nella struttura ad albero degli argomenti.

Utilizzare il seguente esempio di riga comandi per definire un oggetto TOPIC per multicast:

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

Viene creato un nuovo oggetto TOPIC denominato *ALLSPORTS* . Ha una stringa di argomenti *Sports*, il relativo oggetto delle informazioni di comunicazione è denominato *MC1* (che è il nome specificato quando si definisce un oggetto COMMINFO nell'esempio precedente) e multicast è abilitato.

Verifica della pubblicazione / sottoscrizione multicast

Una volta creati gli oggetti TOPIC e COMMINFO, è possibile verificarli utilizzando l'esempio `amqspubc` e `amqssubc` . Per ulteriori informazioni su questi esempi, consultare Programmi di esempio di pubblicazione / sottoscrizione.

1. Aprire due finestre della riga comandi; la prima riga comandi è per l'esempio di pubblicazione `amqspubc` e la seconda riga comandi è per l'esempio di sottoscrizione `amqssubc` .
2. Immettere il seguente comando nella riga comandi 1:

   ```
   amqspubc Sports QM1
   ```

   dove *Sports* è la stringa argomento dell'oggetto TOPIC definito in un esempio precedente e *QM1* è il nome del gestore code.
3. Immettere il seguente comando nella riga comandi 2:

   ```
   amqssubc Sports QM1
   ```

   dove *Sports* e *QM1* sono gli stessi utilizzati nel passaggio “2” a pagina 152.
4. Immettere `Hello world` alla riga comandi 1. Se la porta e l'indirizzo IP specificati nell'oggetto COMMINFO sono configurati correttamente; l'esempio `amqssubc` , che è in ascolto sulla porta per le pubblicazioni dall'indirizzo specificato, emette `Hello world` sulla riga comandi 2.

Topologia argomento IBM WebSphere MQ Multicast

Utilizzare questo esempio per comprendere la topologia dell'argomento IBM WebSphere MQ Multicast.

IBM WebSphere MQ Il supporto multicast richiede che ogni struttura ad albero secondaria abbia il proprio gruppo multicast e flusso di dati all'interno della gerarchia totale.

Lo schema di reindirizzamento IP *rete classful* ha uno spazio degli indirizzi designato per l'indirizzo multicast. L'intervallo multicast completo dell'indirizzo IP è compreso tra `224.0.0.0` e `239.255.255.255`, ma alcuni di questi indirizzi sono riservati. Per un elenco di indirizzi riservati, contattare l'amministratore di sistema o consultare IPv4 Multicast Address Space Registry per ulteriori informazioni. Si consiglia di utilizzare l'indirizzo multicast di ambito locale nell'intervallo compreso tra `239.0.0.0` e `239.255.255.255`.

Nel seguente diagramma, sono disponibili due possibili flussi di dati multicast:

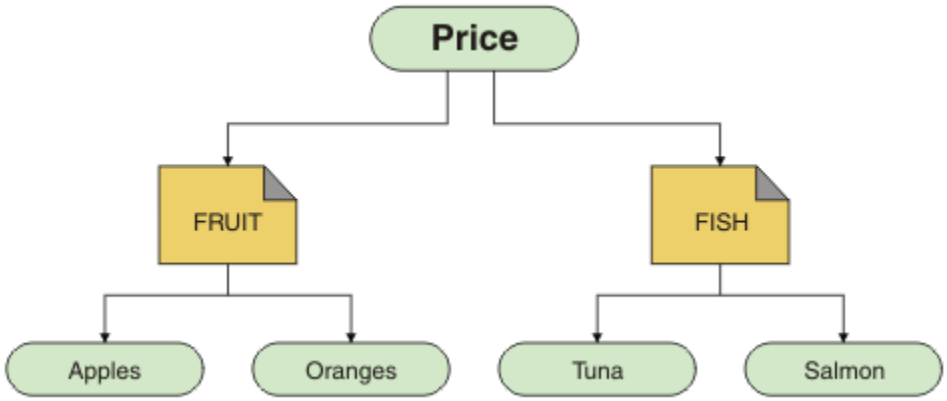
```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX
)

DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

dove *239.XXX.XXX.XXX* e *239.YYY.YYY.YYY* sono indirizzi multicast validi.

Queste definizioni degli argomenti vengono utilizzate per creare una struttura ad albero degli argomenti come mostrato nel seguente diagramma:

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)
DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



Ciascun oggetto informazioni di comunicazione multicast (COMMINFO) rappresenta un flusso di dati differente poiché i rispettivi indirizzi di gruppo sono diversi. In questo esempio, l'argomento FRUIT è definito per utilizzare l'oggetto COMMINFO MC1, l'argomento FISH è definito per utilizzare l'oggetto COMMINFO MC2e il nodo Price non ha definizioni multicast.

WebSphere MQ Multicast ha un limite di 255 caratteri per le stringhe argomento. Questa limitazione significa che è necessario prestare attenzione ai nomi dei nodi e dei nodi foglia all'interno della struttura ad albero; se i nomi dei nodi e dei nodi foglia sono troppo lunghi, la stringa dell'argomento potrebbe superare 255 caratteri e restituire il codice motivo 2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR . Si consiglia di rendere le stringhe di argomenti il più brevi possibile perché le stringhe di argomenti più lunghe potrebbero avere un effetto negativo sulle prestazioni.

Controllo della dimensione dei messaggi multicast

Utilizzare queste informazioni per informazioni sul formato del messaggio WebSphere MQ e ridurre la dimensione dei messaggi WebSphere MQ .

WebSphere MQ i messaggi hanno un numero di attributi associati che sono contenuti nel descrittore del messaggio. Per i messaggi di piccole dimensioni, questi attributi potrebbero rappresentare la maggior parte del traffico dati e possono avere un effetto negativo significativo sulla velocità di trasmissione. WebSphere MQ Multicast consente all'utente di configurare quali attributi, se presenti, vengono trasmessi insieme al messaggio.

La presenza di attributi del messaggio, diversi dalla stringa dell'argomento, dipende dal fatto che l'oggetto COMMINFO indichi che devono essere inviati o meno. Se un attributo non viene trasmesso, l'applicazione di ricezione applica un valore predefinito. I valori MQMD predefiniti non sono necessariamente uguali al valore MQMD_DEFAULT e sono descritti in Tabella 9 a pagina 154.

L'oggetto COMMINFO contiene l'attributo MCPROP che controlla il numero di campi MQMD e il flusso di proprietà utente con il messaggio. Impostando il valore di questo attributo su un livello appropriato, è possibile controllare la dimensione dei messaggi WebSphere MQ Multicast:

MCPROP
Il controllo proprietà multicast verifica quante proprietà utente e MQMD vengono trasmesse insieme al messaggio.

TUTTO
Vengono trasmesse tutte le proprietà utente e tutti i campi di MQMD.

Rispondi
Solo le proprietà utente e i campi MQMD che si occupano delle risposte ai messaggi vengono trasmessi. Queste proprietà sono:

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- ReplyToQmgr

USER
Solo le proprietà utente vengono trasmesse.

NESSUNO
Non vengono trasmessi né le proprietà utente, né i campi MQMD.

COMPAT
Questo valore fa in modo che la trasmissione del messaggio venga eseguita in una modalità compatibile a RMM, che consente alcune interoperazioni con le attuali applicazioni XMS e WebSphere Message Broker RMM .

Attributi messaggio multicast

Gli attributi dei messaggi possono provenire da diverse posizioni, ad esempio MQMD, i campi in MQRFH2e le proprietà dei messaggi.

La seguente tabella mostra cosa succede quando i messaggi vengono inviati in base al valore di MCPROP e il valore predefinito utilizzato quando non viene inviato alcun attributo.

Tabella 9. Attributi di messaggistica e modalità di relazione con multicast

Attributo	Azione quando si utilizza multicast	Valore predefinito se non trasmesso
TopicString	Sempre incluso	Non applicabile
StrucId MQMQ	Non trasmesso	Non applicabile
Versione MQMD	Non trasmesso	Non applicabile
Prospetto	Incluso se non predefinito	0
MsgType	Incluso se non predefinito	MQMT_DATAGRAM
Scadenza	Incluso se non predefinito	0
Feedback	Incluso se non predefinito	0
Codifica	Incluso se non predefinito	MQENC_NORMAL (equivalente)
CodedCharSetId	Incluso se non predefinito	1208
Formato	Incluso se non predefinito	MQRFH2
Priorit...	Incluso se non predefinito	4
Persistenza	Incluso se non predefinito	MQPER_NOT_PERSISTENT
MsgId	Incluso se non predefinito	Nullo
CorrelId	Incluso se non predefinito	Nullo
BackoutCount	Incluso se non predefinito	0
ReplyToQ	Incluso se non predefinito	Spazio
ReplyToQMgr	Incluso se non predefinito	Spazio

Tabella 9. Attributi di messaggistica e modalità di relazione con multicast (Continua)		
Attributo	**Azione quando si utilizza multicast**	**Valore predefinito se non trasmesso**
UserIdentifier	Incluso se non predefinito	Spazio
AccountingToken	Incluso se non predefinito	Nullo
Tipo IT PutApp	Incluso se non predefinito	JAVA MQAT
PutAppIName	Incluso se non predefinito	Spazio
PutDate	Incluso se non predefinito	Spazio
PutTime	Incluso se non predefinito	Spazio
ApplOriginData	Incluso se non predefinito	Spazio
GroupID	Esclusi	Non applicabile
MsgSeqNumber	Esclusi	Non applicabile
Offset	Esclusi	Non applicabile
MsgFlags	Esclusi	Non applicabile
OriginalLength	Esclusi	Non applicabile
UserProperties	Inclusi	Non applicabile

Riferimenti correlati
COMMINFO ALTER
DEFINE COMMINFO

Abilitazione della conversione dati 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.

In un'installazione con piattaforma mista, è possibile che la maggior parte dei client richieda i dati in un formato che non sia il formato nativo dell'applicazione di trasmissione. In questa situazione, i valori **CCSID** e **ENCODING** dell'oggetto COMMINFO multicast possono essere utilizzati per definire la codifica della trasmissione del messaggio per una maggiore efficienza.

WebSphere MQ Multicast supporta la conversione dei dati del payload del messaggio per i seguenti formati integrati:

- MQADMIN
- MQEVENT
- MQPCF
- MQRFH
- MQRFH2
- MQSTR

Oltre a questi formati, è anche possibile definire i propri formati e utilizzare un'uscita di conversione dati MQDXP - parametro di conversione dati .

Per informazioni sulla programmazione delle conversione dei dati, consultare Conversione dei dati in MQI 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.

Monitoraggio applicazione multicast

Utilizzare queste informazioni per informazioni sulla gestione e il controllo di WebSphere MQ Multicast.

Lo stato dei publisher e dei sottoscrittori correnti per il traffico multicast (ad esempio, il numero di messaggi inviati e ricevuti o il numero di messaggi persi) viene periodicamente trasmesso al server dal client. Quando si riceve lo stato, l'attributo COMMEV dell'oggetto COMMINFO specifica se il gestore code inserisce o meno un messaggio di evento nel SISTEMA SYSTEM.ADMIN.PUBSUB.EVENT. Il messaggio dell'evento contiene le informazioni sullo stato ricevute. Queste informazioni sono un aiuto diagnostico prezioso per individuare l'origine di un problema.

Utilizzare il comando MQSC **DISPLAY CONN** per visualizzare le informazioni di connessione relative alle applicazioni connesse al gestore code. Per ulteriori informazioni sul comando **DISPLAY CONN** , consultare DISPLAY CONN.

Utilizzare il comando MQSC **DISPLAY TPSTATUS** per visualizzare lo stato dei publisher e dei sottoscrittori. Per ulteriori informazioni sul comando **DISPLAY TPSTATUS** , consultare DISPLAY TPSTATUS.

COMMEV e l'indicatore di affidabilità del messaggio multicast

L' *indicatore di affidabilità*, utilizzato insieme all'attributo **COMMEV** dell'oggetto COMMINFO, è un elemento chiave nel monitoraggio dei publisher e dei sottoscrittori WebSphere MQ Multicast. L'indicatore di affidabilità (il campo **MSGREL** che viene restituito nei comandi di stato di pubblicazione o sottoscrizione) è un indicatore WebSphere MQ che illustra la percentuale di trasmissioni che non hanno errori. A volte, i messaggi devono essere ritrasmessi a causa di un errore di trasmissione, che si riflette nel valore di **MSGREL**. Le cause potenziali degli errori di trasmissione includono sottoscrittori lenti, reti occupate e interruzioni di rete. **COMMEV** controlla se i messaggi di eventi vengono generati per handle multicast creati utilizzando l'oggetto COMMINFO ed è impostato su uno dei tre valori possibili:

DISABILITATO
I messaggi di evento non vengono scritti.

Abilitato
I messaggi di evento vengono sempre scritti, con una frequenza definita nel parametro COMMINFO **MONINT** .

ECCEZIONE
I messaggi di evento vengono scritti se l'affidabilità del messaggio è inferiore alla soglia di affidabilità. Un livello di affidabilità del messaggio del 90% o inferiore indica che potrebbe esserci un problema con la configurazione di rete o che una o più applicazioni di pubblicazione / sottoscrizione sono in esecuzione troppo lentamente:

- Il valore **MSGREL(100,100)** indica che non si sono verificati problemi a breve termine o a lungo termine.
- Il valore **MSGREL(80,60)** indica che il 20% dei messaggi sta attualmente avendo problemi, ma che si tratta anche di un miglioramento rispetto al valore a lungo termine di 60.

I client potrebbero continuare a trasmettere e ricevere traffico multicast anche quando la connessione unicast al gestore code è interrotta, pertanto i dati potrebbero non essere aggiornati.

Affidabilità dei messaggi multicast

Utilizzare queste informazioni per informazioni su come impostare la sottoscrizione WebSphere MQ Multicast e la cronologia dei messaggi.

Un elemento chiave per il superamento dell'errore di trasmissione con multicast è il buffer di dati trasmessi di WebSphere MQ(una cronologia di messaggi da conservare all'estremità di trasmissione del collegamento). Questo processo significa che non è richiesto alcun buffer di messaggi nel processo di inserimento dell'applicazione perché WebSphere MQ fornisce l'affidabilità. La dimensione di questa cronologia viene configurata mediante l'oggetto informazioni di comunicazione (COMMINFO), come descritto nelle seguenti informazioni. Un buffer di trasmissione più grande significa che c'è più cronologia di trasmissione da ritrasmettere se necessario, ma a causa della natura del multicast, la consegna garantita al 100% non può essere supportata.

La cronologia dei messaggi multicast WebSphere MQ è controllata nell'oggetto COMMINFO (communication information) dall'attributo **MSGHIST** :

MSGHIST
Questo valore è la quantità di cronologia dei messaggi in kilobyte conservata dal sistema per gestire le ritrasmissioni nel caso di NACK (riconoscimenti negativi).

Il valore 0 fornisce il livello minimo di affidabilità. Il valore predefinito è 100 KB.

La cronologia delle nuove sottoscrizioni WebSphere MQ Multicast è controllata nell'oggetto informazioni di comunicazione (COMMINFO) dall'attributo **NSUBHIST** :

NSUBHIST
La cronologia nuovo sottoscrittore verifica se un sottoscrittore che si iscrive a un flusso di pubblicazioni riceve tutti dati attualmente disponibili o solo le pubblicazioni disponibili dal momento della sottoscrizione.

NESSUNO
Un valore di NONE fa sì che il trasmettitore trasmetta solo la pubblicazione effettuata dal momento della sottoscrizione. NONE è il valore predefinito.

TUTTO
Un valore ALL fa sì che il trasmettitore ritrasmetta la quantità di cronologia dell'argomento nota. In alcune circostanze, questa situazione può fornire un comportamento simile alle pubblicazioni conservate.

Nota: L'utilizzo del valore di ALL potrebbe avere un effetto negativo sulle prestazioni se esiste una cronologia di argomenti di grandi dimensioni poiché tutta la cronologia degli argomenti viene ritrasmessa.

Riferimenti correlati
DEFINE COMMINFO
COMMINFO ALTER

Attività multicast avanzate

Utilizzare queste informazioni per informazioni sulle attività di gestione avanzate di WebSphere MQ Multicast, quali la configurazione dei file `.ini` e l'interoperabilità con WebSphere MQ LLM.

Per considerazioni sulla sicurezza in un'installazione multicast, consultare Sicurezza multicast .

Collegamento tra domini di pubblicazione / sottoscrizione multicast e non multicast

Utilizzare queste informazioni per comprendere cosa accade quando un publisher non multicast pubblica in un argomento abilitato WebSphere MQ Multicast.

Se un publisher non multicast pubblica un argomento definito come **MCAST** abilitato e **BRIDGE** abilitato, il gestore code trasmette il messaggio tramite multicast direttamente a tutti i sottoscrittori che potrebbero essere in ascolto. Un publisher multicast non può eseguire la pubblicazione su argomenti che non sono abilitati multicast.

Gli argomenti esistenti possono essere abilitati multicast impostando i parametri **MCAST** e **COMMINFO** di un oggetto argomento. Consultare Initial multicast concepts per ulteriori informazioni su questi parametri.

L'attributo oggetto COMMINFO **BRIDGE** controlla le pubblicazioni dalle applicazioni che non utilizzano multicast. Se **BRIDGE** è impostato su ENABLED e il parametro **MCAST** dell'argomento è impostato anche su ENABLED, le pubblicazioni delle applicazioni che non utilizzano il multicast vengono collegate tramite bridge alle applicazioni che lo utilizzano. Per ulteriori informazioni sul parametro **BRIDGE** , consultare DEFINE COMMINFO.

Configurazione dei file .ini per Multicast

Utilizzare queste informazioni per comprendere i campi WebSphere MQ Multicast nei file `.ini` .

È possibile eseguire una ulteriore configurazione WebSphere MQ Multicast in un file `ini` . Il file `ini` specifico che è necessario utilizzare dipende dal tipo di applicazioni:

- Client: configurare il file *MQ_DATA_PATH*`/mqclient.ini` .
- Gestore code: configurare il file *MQ_DATA_PATH*`/qmgrs/`*QMNAME*`/qm.ini` .

dove *MQ_DATA_PATH* è l'ubicazione della directory di dati WebSphere MQ (`/var/mqm/mqclient.ini`) e *QMNAME* è il nome del gestore code a cui si applica il file `.ini` .

Il file `.ini` contiene campi utilizzati per ottimizzare il funzionamento di WebSphere MQ Multicast:

```
Multicast:
  Protocol              = IP | UDP
  IPVersion             = IPV4 | IPV6 | ANY | BOTH
  LimitTransRate        = DISABLED | STATIC | DYNAMIC
  TransRateLimit        = 100000
  SocketTTL             = 1
  Batch                 = NO
  Loop                  = 1
  Interface             = <IPaddress>
  FeedbackMode          = ACK | NACK | WAIT1
  HeartbeatTimeout      = 20000
  HeartbeatInterval     = 2000
```

Protocollo

UDP
In questa modalità, i package vengono inviati utilizzando il protocollo UDP. Tuttavia, gli elementi di rete non possono fornire assistenza nella distribuzione multicast come avviene in modalità IP. Il formato del pacchetto rimane compatibile con PGM. Questo è il valore predefinito.

IP
In questa modalità, il trasmettitore invia pacchetti IP non elaborati. Gli elementi di rete con supporto PGM assistono nella distribuzione di pacchetti multicast affidabili. Questa modalità è completamente compatibile con lo standard PGM.

IPVERSION

IPV4
Comunicare utilizzando solo protocollo IPv4 . Questo è il valore predefinito.

IPV6
Comunicare utilizzando solo protocollo IPv6 .

ANY
Comunicare utilizzando IPv4, IPv6o entrambi, a seconda del protocollo disponibile.

ENTRAMBI
Supporta la comunicazione utilizzando IPv4 e IPv6.

Frequenza LimitTrans

DISABILITATO
Non esiste alcun controllo della velocità di trasmissione. Questo è il valore predefinito.

STATICO
Implementa il controllo della velocità di trasmissione statica. Il trasmettitore non trasmetterà ad una velocità superiore a quella specificata dal parametro TransRateLimit.

DINAMICO
Il trasmettitore adatta la sua velocità di trasmissione in base al feedback che riceve dai ricevitori. In questo caso, il limite della velocità di trasmissione non può essere superiore al valore specificato dal parametro TransRateLimit. Il trasmettitore cerca di raggiungere una velocità di trasmissione ottimale.

Limite TransRate
Il limite della velocità di trasmissione in Kbps.

SocketTTL
Il valore di SocketTTL determina se il traffico multicast può passare attraverso un router o il numero di router che può passare attraverso.

Batch
Controlla se i messaggi vengono batch o inviati immediatamente Ci sono 2 valori possibili:

- *NO* I messaggi non sono in batch, vengono inviati immediatamente.
- *Sì* I messaggi vengono sottoposti a batch.

Loop
Impostare il valore su 1 per abilitare il loop multicast. Il loop multicast definisce se i dati inviati vengono inviati in loop o meno all'host.

Interfaccia
L'indirizzo IP dell'interfaccia su cui fluisce il traffico multicast. Per ulteriori informazioni e la risoluzione dei problemi, consultare: Test delle applicazioni multicast su una rete non multicast e Impostazione della rete appropriata per il traffico multicast

FeedbackMode

NACK
Feedback da riconoscimenti negativi. Questo è il valore predefinito.

ACK
Feedback da riconoscimenti positivi.

WAIT1
Feedback da parte di riconoscimenti positivi in cui il trasmettitore attende solo 1 ACK da uno qualsiasi dei ricevitori.

HeartbeatTimeout
Il timeout heartbeat in millisecondi. Il valore 0 indica che gli eventi di timeout heartbeat non vengono generati dal destinatario o dai destinatari dell'argomento. Il valore predefinito è 20000.

HeartbeatInterval
L'intervallo di heartbeat in millisecondi. Il valore 0 indica che non viene inviato alcun heartbeat. L'intervallo di heartbeat deve essere notevolmente inferiore al valore **`HeartbeatTimeout`** per evitare falsi eventi di timeout di heartbeat. Il valore predefinito è 2000.

Interoperabilità multicast con WebSphere MQ Low Latency Messaging

Utilizzare queste informazioni per comprendere l'interoperabilità tra WebSphere MQ Multicast e WebSphere MQ Low Latency Messaging (LLM).

Il trasferimento payload di base è possibile per un'applicazione che utilizza LLM, con un'altra applicazione che utilizza multicast per scambiare messaggi in entrambe le direzioni. Anche se multicast utilizza la tecnologia LLM, il prodotto LLM stesso non è incorporato. Pertanto, è possibile installare LLM e WebSphere MQ Multicast e gestire e gestire i due prodotti separatamente.

Le applicazioni LLM che comunicano con multicast potrebbero dover inviare e ricevere le proprietà del messaggio. Le proprietà del messaggio WebSphere MQ e i campi MQMD vengono trasmessi come proprietà del messaggio LLM con codici proprietà specifici del messaggio LLM, come mostrato nella tabella seguente:

Tabella 10. Associazioni delle proprietà LLM WebSphere MQ a WebSphere MQ			
proprietà WebSphere MQ	**WebSphere MQ Tipo di proprietà LLM**	**Tipo di proprietà LLM**	**Codice proprietà LLM**
MQMD.Report	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1001
MQMD.MsgType	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1002
MQMD.Expiry	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1003
MQMD.Feedback	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1004
MQMD.Encoding	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1005
MQMD.CodedCharSetId	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1006
MQMD.Format	RMM_MSG_PROP_BYTES	Stringa LLM_PROP_KIND_	-1007
MQMD.Priority	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1008
MQMD.Persistence	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1009
MQMD.MsgId	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_ByteArray	-1010
MQMD.BackoutCount	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1012
MQMD.ReplyToQ	RMM_MSG_PROP_BYTES	Stringa LLM_PROP_KIND_	-1013
MQMD.ReplyToQMger	RMM_MSG_PROP_BYTES	Stringa LLM_PROP_KIND_	-1014
MQMD.PutDate	RMM_MSG_PROP_BYTES	Stringa LLM_PROP_KIND_	-1020
MQMD.PutTime	RMM_MSG_PROP_BYTES	Stringa LLM_PROP_KIND_	-1021
MQMD.ApplOriginData	RMM_MSG_PROP_BYTES	Stringa LLM_PROP_KIND_	-1022
MQPubOptions	RMM_MSG_PROP_INT32	LLM_PROP_KIND_int32	-1053

Per ulteriori informazioni su LLM, consultare la documentazione del prodotto LLM: WebSphere MQ Low Latency Messaging.

AmministrazioneHP Integrity NonStop Server

Utilizzare queste informazioni per informazioni sulle attività di amministrazione per il client IBM WebSphere MQ per HP Integrity NonStop Server.

Sono disponibili due attività di amministrazione:

1. Avviare manualmente TMF/Gateway da Pathway.
2. Arresto di TMF/Gateway da Pathway.

Avvio manuale di TMF/Gateway da Pathway

Puoi consentire a Pathway di avviare automaticamente il TMF/Gateway alla prima richiesta di arruolamento oppure puoi avviare manualmente il TMF/Gateway da Pathway.

Procedura

Per avviare manualmente il TMF/Gateway da Pathway, immettere il seguente comando PATHCOM:

```
START SERVER <server_class_name>
```

Se un'applicazione client effettua una richiesta di iscrizione prima che il TMF/Gateway completi il ripristino delle transazioni in dubbio, la richiesta viene trattenuta per un massimo di 1 secondo. Se il

ripristino non viene completato entro tale periodo di tempo, l'arruolamento viene rifiutato. Il client riceve quindi un errore MQRC_UOW_ENLISTMENT_ERROR dall'utilizzo di una MQI transazionale.

Arresto di TMF/Gateway da Pathway

Questa attività descrive come arrestare TMF/Gateway da Pathway e come riavviare TMF/Gateway dopo averlo arrestato.

Procedura

1. Per impedire che vengano effettuate nuove richieste di arruolamento a TMF/Gateway, immettere il seguente comando:

   ```
   FREEZE SERVER <server_class_name>
   ```

2. Per attivare il TMF/Gateway per completare tutte le operazioni in corso e per terminare, immettere il seguente comando:

   ```
   STOP SERVER <server_class_name>
   ```

3. Per consentire al TMF/Gateway di riavviarsi automaticamente al primo inserimento o manualmente, seguendo i passi 1 e 2, immetti il seguente comando:

   ```
   THAW SERVER <server_class_name>
   ```

   Alle applicazioni viene impedito di effettuare nuove richieste di arruolamento e non è possibile immettere il comando **START** fino a quando non si immette il comando **THAW** .

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. Tuttavia, è responsabilità dell'utente valutare e verificare il funzionamento di qualsiasi prodotto, programma o servizio nonIBM .

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:

IBM Director of Commercial Relations IBM Corporation North Castle Drive Armonk, NY 10504-1785 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:

Licenza di proprietà intellettuale e legge sulla proprietà intellettuale 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 in queste informazioni a siti Web nonIBM viene fornito solo per comodità e non serve in alcun modo da approvazione di tali 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 Software Interoperability Coordinator, Department 49XA 3605 Highway 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 sono state ottenute dai fornitori di tali prodotti. IBM non ha testato quei prodotti e non può confermarne l'accuratezza delle prestazioni, la compatibilità o qualsiasi altro reclamo relativo ai prodotti non IBM. Commenti relativi alle prestazioni di prodotti non IBM, dovrebbero essere indirizzati ai fornitori di questi prodotti.

Tutte le dichiarazioni relative all'orientamento o alle intenzioni future di IBMsono soggette a modifica o a 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: